

Тодор Димов



ИНФОРМАТИКА за II (втора) година гимназиско образование

Со Одлука за одобрување и употреба на учебник по наставниот предмет Информатика за II (втора) година во средно гимназиско образование бр. 26-843/1 од 22.07.2026 година, донесена од Националната комисија за учебници.

Скопје, 2026

Информатика

за II (втора) година гимназиско образование

Издавач:

ДПТУ „СТУДЕНТСКИ СЕРВИС“ ДОО - Скопје

Автор:

Тодор Димов

Рецензенти

Проф. д-р Марија Календар

Проф д-р Горан Јакимовски

Жаклина Милошева

Уредник:

Симона Ристовска

Лектор:

Проф. д-р Томислав Треневски

Дизајн и техничка подготовка:

ДПТУ „СТУДЕНТСКИ СЕРВИС“ ДОО - Скопје

Печати:

ДПТУ „СТУДЕНТСКИ СЕРВИС“ ДОО

Ул. „Перо Наков“ бр.112 - Скопје

Тираж:

5000 примероци

Скопје, 2026

CIP - Каталогизација во публикација

Национална и универзитетска библиотека «Св. Климент Охридски», Скопје

004(075.3)

ДИМОВ, Тодор

Информатика за II (втора) година гимназиско образование /

Тодор Димов. - Скопје : Студентски сервис, 2026. - 330 стр. : илустр. ; 28 см

ISBN 978-608-4777-47-2

COBISS.MK-ID 69328901

СОДРЖИНА

ТЕМА 1.....7

СОВРЕМЕНИ ТЕХНОЛОГИИ ВО ИНФОРМАТИКАТА И ДИГИТАЛНА ТРАНСФОРМАЦИЈА.....	8
САЈБЕР БЕЗБЕДНОСТ И ЗАШТИТА НА ЛИЧНИТЕ ПОДАТОЦИ	12
КРИПТОГРАФИЈА	19
BLOCKCHAIN ТЕХНОЛОГИЈА – ДОВЕРЛИВИ ДИГИТАЛНИ ЗАПИСИ ВО СОВРЕМЕНИОТ СВЕТ ..	26
РОБОТИКА И АВТОМАТИЗАЦИЈА	32
ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА И МАШИНСКО УЧЕЊЕ	38
ГОЛЕМИ ЈАЗИЧНИ МОДЕЛИ И ГЕНЕРАТИВНА ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА	46
ПРОГРАМИРАЊЕ ВО C++	52

ТЕМА 2.....53

ДИГИТАЛНО ПРЕТСТАВУВАЊЕ НА ПОДАТОЦИ ВО КОМПЈУТЕР	54
БРОЈНИ СИСТЕМИ И КОНВЕРЗИИ	58
ПРЕТСТАВУВАЊЕ НА ЦЕЛИ БРОЕВИ И ОПСЕГ НА ПОДАТОЦИ.....	64
ОПТИМИЗАЦИЈА И ЕФИКАСНОСТ НА АЛГОРИТМИ	68
КОМБИНИРАЊЕ УСЛОВИ И ЦИКЛУСИ	73
МОДУЛАРНО ПРОГРАМИРАЊЕ И КОНЦЕПТОТ „РАЗДЕЛИ ПА ВЛАДЕЈ“	83
ДЕФИНИРАЊЕ ФУНКЦИИ ВО C++	88
ПАРАМЕТРИ И ВРАЌАЊЕ ВРЕДНОСТ	94
ЛОКАЛНИ И ГЛОБАЛНИ ПРОМЕНЛИВИ КАЈ ФУНКЦИИ	100
ОСНОВНИ МАТЕМАТИЧКИ ФУНКЦИИ ОД БИБЛИОТЕКАТА SMATH (ДЕЛ 1)	106
РЕШАВАЊЕ ТЕКСТУАЛНИ ЗАДАЧИ СО ПРИМЕНА НА ФУНКЦИИ	112
ЗАДАЧИ ЗА ВЕЖБАЊЕ ОД ФУНКЦИИ	118
ЕДНОДИМЕНЗИОНАЛНА НИЗА КАКО СТРУКТУРА НА ПОДАТОЦИ.....	121
ДЕКЛАРИРАЊЕ И ИНИЦИЈАЛИЗАЦИЈА НА ЕДНОДИМЕНЗИОНАЛНИ НИЗИ	127
ИНДЕКС НА ЕЛЕМЕНТ И ПРИСТАПУВАЊЕ ДО ЕЛЕМЕНТИ НА НИЗА	132
ВНЕСУВАЊЕ И ПЕЧАТЕЊЕ НА ЕЛЕМЕНТИ ОД НИЗА	137
ОСНОВНИ ОПЕРАЦИИ СО ЕЛЕМЕНТИ НА НИЗА	142
ВНЕСУВАЊЕ НА ЕЛЕМЕНТИ ВО НИЗА И ОСНОВНИ ПРЕСМЕТКИ	148
ОПРЕДЕЛУВАЊЕ НА НАЈГОЛЕМ И НАЈМАЛ ЕЛЕМЕНТ ВО НИЗА.....	153

ПРЕБРОЈУВАЊЕ НА ЕЛЕМЕНТИ ВО НИЗА	158
НИЗИ ОД КАРАКТЕРИ И ПРЕБАРУВАЊЕ НА ЗНАЦИ.....	163
ПРИМЕНА НА МАТЕМАТИЧКИ ФУНКЦИИ ВРЗ ЕЛЕМЕНТИТЕ НА НИЗА	168
КОМБИНИРАНА ПРИМЕНА НА ФУНКЦИИ И ЕДНОДИМЕНЗИОНАЛНИ НИЗИ.....	173
ЗАДАЧИ ЗА ВЕЖБАЊЕ СО НИЗИ.....	179

ТЕМА 3.....183

ОСНОВИ НА БАЗИТЕ НА ПОДАТОЦИ.....	184
КРЕИРАЊЕ ТАБЕЛИ ВО БАЗА НА ПОДАТОЦИ	189
ТИПОВИ ПОДАТОЦИ	196
ПРИМАРЕН КЛУЧ И ПОВРЗУВАЊЕ НА ТАБЕЛИ.....	204
ФОРМИ ЗА ВНЕСУВАЊЕ И ПРЕГЛЕД НА ПОДАТОЦИ.....	210
ПРАШАЛНИЦИ (QUERIES) – ПРЕБАРУВАЊЕ И СЕЛЕКТИРАЊЕ ПОДАТОЦИ	217
ИЗВЕШТАИ (REPORTS) – ПРИКАЖУВАЊЕ И ПЕЧАТЕЊЕ НА ПОДАТОЦИТЕ.....	226
ИЗРАБОТКА НА ЦИРКУЛАРНИ ПИСМА СО БАЗА НА ПОДАТОЦИ	232
SQL – ОСНОВЕН ЈАЗИК ЗА РАБОТА СО БАЗИ НА ПОДАТОЦИ	239

ТЕМА 4.....245

МУЛТИМЕДИЈА.....	247
ВЕКТОРСКА И РАСТЕРСКА ГРАФИКА.....	250
ОСНОВИ НА ОБРАБОТКА НА СЛИКА.....	253
СЛОЕВИ (LAYERS) ВО ГРАФИКА	267
ГЕНЕРАТИВНА ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА ЗА ГРАФИКА	271

ТЕМА 5.....279

ПОИМ ЗА ВЕБ-СТРАНИЦА	281
ОСНОВИ НА HTML	285



Живееме во време во кое технологијата е неразделен дел од секојдневниот живот. Компјутерите, интернетот, мобилните уреди и дигиталните услуги ни овозможуваат побрзо да комуницираме, учиме, работиме и создаваме. Зад сите овие технологии стојат знаења и вештини од областа на информатиката. Информатиката денес не претставува само користење компјутер. Таа опфаќа разбирање на начинот на кој функционираат современите технологии, обработка и организација на податоци, развој на програми, создавање дигитални содржини и решавање проблеми со помош на компјутерски системи.

Овој учебник е изработен според наставната програма по Информатика за втора година гимназиско образование и има цел да им помогне на учениците да стекнат современи знаења и практични вештини потребни за успешно учество во дигиталното општество. Содржините во учебникот ги опфаќаат современите технологии во информатиката, програмирањето, базите на податоци, мултимедијата, компјутерската графика и веб развојот. Посебно внимание е посветено на практичната примена на знаењата преку примери, задачи и активности кои ги поттикнуваат учениците да размислуваат, да истражуваат и да создаваат сопствени решенија.

Целта на учебникот не е само да понуди нови информации туку и да развие логичко размислување, креативност, дигитална писменост и подготвеност за доживотно учење. Во време кога технологијата постојано се менува, најважна станува способноста да се учи, да се приспособува и да се применува знаењето во нови ситуации.

Се надеваме дека овој учебник ќе биде корисен водич во откривањето на светот на информатиката и ќе ги мотивира учениците да продолжат да истражуваат, да создаваат и да ги развиваат своите идеи.

Информатиката не е само предмет што се изучува во училишта. Таа е јазикот на современиот свет и алатка за создавање на иднината.



TEMA 1





1.1

СОВРЕМЕНИ ТЕХНОЛОГИИ ВО ИНФОРМАТИКАТА И ДИГИТАЛНА ТРАНСФОРМАЦИЈА

Живееме во ера во која промените се случуваат со огромна брзина, а главен двигател на тие промени е информатиката. Денес, терминот „технологија“ веќе не се однесува само на обични компјутери и основни програми, туку на комплексен систем од иновации кои фундаментално го менуваат начинот на кој работиме, комуницираме и живееме.

Под поимот современи технологии во информатиката се подразбираат најновите, напредни алатки, хардверски уреди, софтверски решенија и мрежни системи што овозможуваат брза обработка, складирање и анализа на огромно количество податоци. Овие технологии претставуваат премин од некогашното едноставно, статично управување со бази на податоци кон активен, интелигентен екосистем. Тие овозможуваат анализирање на огромни количини на податоци и донесување паметни одлуки во реално време, поврзувајќи го дигиталниот со физичкиот свет.

Современите технологии постојано еволуираат и се интегрираат во сите пори на човечкото живеење, создавајќи го феноменот наречен дигитална трансформација – процес на интегрирање на дигиталната технологија во сите области на едно општество, во образованието, економијата, производството или во бизнисите, што суштински го менува начинот на кој функционираат тие.

Светот на информатиката денес е предводен од неколку клучни столбови:

- Вештачка интелигенција (AI) и Големи јазични модели (LLMs), кои претставуваат напредни системи што можат да процесираат и да генерираат текст, код и идеи на ниво слично на човечкото (на пр. GPT модели, Gemini, Claude), анализирајќи милијарди податоци за неколку секунди.
- Напредна роботика, која ги соединува информатиката, вештачката интелигенција и машинството. Тоа се автономни или полуавтономни машини способни да извршуваат комплексни физички задачи, да учат од околината и да реагираат на промени во реално време.
- Интернет на нештата (IoT) претставува мрежа од физички уреди (паметни телефони, апарати за домаќинството, индустриски сензори) кои се поврзани на Интернет и меѓусебно комуницираат.
- Технологии во облак (Cloud Computing), кои овозможуваат складирање и обработка на податоци на оддалечени сервери наместо на локален компјутер (на пр. Google Drive, Netflix).

Примената на овие технологии во нашето секојдневие е многу застапена. Моносите за нивна имплементација се практично неограничени. Практичната примена на овие технологии е најзастапена во:

- **Креативна работа и лични асистенти.** Со помош на големите јазични модели, секој човек денес има пристап до инстантен „дигитален мозок“. Овие модели помагаат во пишување есеи, превод на јазици во реално време, учење нови вештини и генерирање на компјутерски код.
- **Паметна автоматизација во домот и во индустријата.** Преку роботиката и IoT, во секојдневниот живот користиме роботски правосмукалки кои сами ја мапираат куќата. Во индустријата,

роботски раце вршат прецизни операции, пакување и опасни физички задачи, со што се елиминира ризикот за луѓето.

- **Образование и медицина.** Големите јазични модели служат како персонални тутори кои го приспособуваат објаснувањето според знаењето на ученикот. Во медицината, хируршки работи (како системот Da Vinci) им овозможуваат на лекарите да вршат екстремно прецизни операции на далечина.
- **Електронски услуги (Е-услуги).** Плаќање сметки со еден клик, онлајн банкарство и поддршка на корисници преку паметни „чет-ботови“, кои работат 24/7 благодарение на ВИ.
- **Сообраќајот,** во кој се користат системи за навигација што помагаат при изборот на најсоодветна маршрута.
- **Земјоделството,** за имплементирање на паметни системи за следење на временските услови и за наводнување.

1.1.1 ПРЕДНОСТИ, ПРЕДИЗВИЦИ И ВЛИЈАНИЕ ВРЗ ОПШТЕСТВОТО

Како и секоја голема револуција, така и дигиталната трансформација носи со себе сериозни придобивки, но и предизвици кои општеството треба да ги надмине.

Позитивни аспекти	Негативни аспекти
Ефикасност и продуктивност: Големите јазични модели и роботите ги преземаат мануелните и репетитивни задачи, ослободувајќи го човекот за креативно размислување.	Губење на работни места: Автоматизацијата, роботиката и ВИ можат да заменат одредени професии (од фабрички работници до администратори), што изнудува брза преквалификација.
Безбедност на работното место: Роботите можат да работат во опасни средини (рудници, длабоки мориња, нуклеарни центри) каде што е загрозен човечкиот живот.	Зависност и когнитивна мрзливост: Претераното потпирање на јазичните модели за пишување и размислување може да ги намали способностите за критичко размислување и креативност, особено кај младите.
Пристап до знаење и инклузивност: Јазичните бариери исчезнуваат благодарение на напредните LLM преведувачи, овозможувајќи информациите да бидат достапни за секого.	Ширење дезинформации: Големите јазични модели можат ненамерно да генерираат лажни информации што изгледаат уверливо (т.н. „халуцинации“), или да бидат злоупотребени за пропагандни цели.
Подобра дијагностика и нега: Спојот на ВИ и медицинската роботика овозможува подолг и поквалитетен животен век за луѓето.	Сајбер-безбедност и етички прашања: Кој е одговорен ако роботот направи грешка? Како да се заштитат личните податоци со кои се хранат јазичните модели?

Потребно е паметно да се применуваат современите технологии во секојдневниот живот. Следниве примери опишуваат како може полезно да се користат технологиите.

- **Адаптивно учење за доживотно образование.** Наместо класични курсеви, попрактично е да се користат отворени образовни платформи базирани на облак за преквалификација или за учење нови вештини (јазичи, програмирање) кои се приспособуваат на сопствената брзина на учење.

- **Паметно менаџирање на здравјето (IoT и Телемедицина).** Користење паметни часовници и апликации за следење на спиењето, пулсот и физичката активност, со цел споделување на тие податоци со матичниот лекар преку дигитални платформи (наместо физички посети за рутински контроли).
- **Автоматизација на рутински задачи со соработници со ВИ.** Користење алатки со вештачка интелигенција како „когнитивни асистенти“ за брзо составување на деловни е-пошти, структурирање белешки од состаноци или пребарување на големи текстуални архиви, со цел да се ослободи време за креативно решавање проблеми.
- **Свесно ограничување на користењето на дигитални уреди.** Како одговор на намалениот распон на внимание, подобро е користење апликации кои го ограничуваат пристапот до социјалните мрежи за време на работните часови, со цел зачувување на способноста за фокусирана работа.



ЗАКЛУЧОК:

Современите технологии во информатиката, од роботиката што го движи физичкиот свет, до големите јазични модели кои го трансформираат интелектуалниот труд се моќни алатки. Нивното влијание врз општеството зависи исклучиво од начинот на кој ги користиме. Нивниот огромен потенцијал треба да се искористи за олеснување на животот, со истовремено развивање на критичко размислување и свесност за етичките норми за заштита од негативните последици.



ДАЛИ ЗНАЕШ?

Дали знаеш дека еден паметен телефон денес има поголема пресметувачка моќ од компјутерите што биле користени при првите вселенски мисии до Месечината?



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Како се дефинира поимот „современи технологии“ во информатиката?
2. Наброј три примери за современи технологии кои денес се најзастапени во светот.
3. Објасни ја со свои зборови врската меѓу современите технологии и процесот на дигитална трансформација.
4. Како функционира Интернетот на нештата (IoT) и како го поврзува нашиот секојдневен живот со мрежата?
5. Зошто големите јазични модели понекогаш генерираат „халуцинации“ (неточни информации) и што значи тоа?
6. Опиши една ситуација каде што ти лично или твојот паметен дом користите IoT технологија во текот на денот.
7. Спореди ги позитивните и негативните страни од масовното воведување на роботите во фабриките. Кој добива, а кој губи?
8. Анализирај го влијанието на облачните технологии врз начинот на кој денес се зачувуваат и се споделуваат податоците во споредба со минатото (хард дискови, УСБ-мемории).
9. Кој треба да сноси етичка и правна одговорност ако медицински робот направи грешка за време на операција — програмерот, инженерот или лекарот? Образложи го својот став.
10. Процени дали брзиот развој на вештачката интелигенција води кон социјална изолација и намалување на креативноста кај

луѓето или, пак, ја зголемува нивната ефикасност.

11. Предложи идеја за нов „паметен уред“ (базиран на IoT или вештачка интелигенција) кој би можел да реши некој реален

проблем во твоето училиште или во твојот град.

12. Напиши краток акциски план (3–4 чекори) за тоа како еден бизнис што работи традиционално на хартија може безбедно да направи дигитална трансформација.



ИСТРАЖУВАЧКИ И ПРАКТИЧНИ ЗАДАЧИ

1. Направи листа на десет современи технологии што ги користиш во текот на еден ден. За секоја технологија напиши во која област се применува и како ти го олеснува секојдневниот живот.
2. Избери една јавна институција (училиште, библиотека, банка, болница или општина) и истражи кои дигитални услуги им ги нуди на граѓаните. Подготви краток извештај.
3. Размисли како изгледал еден училиштен ден пред дваесет години и спореди го со денешниот. Наведи најмалку пет промени што се резултат на современите технологии.
4. Во група со соучениците дискутирај за предностите и можните недостатоци од употребата на паметните телефони во секојдневниот живот. Запиши ги заклучоците.
5. Истражи една современа технологија за која сметаш дека ќе има големо влијание врз иднината. Подготви кратка презентација во која ќе објасниш како функционира и каде може да се применува.



ЗАКЛУЧОК:

Современите технологии претставуваат современи технички и информатички решенија што го олеснуваат секојдневниот живот.

Дигиталната трансформација претставува процес на воведување дигитални технологии во различни области на општеството.

Современите технологии имаат примена во образованието, здравството, индустријата, сообраќајот, трговијата и во комуникацијата.

Примената на современите технологии носи многубројни придобивки, но и одредени предизвици.

Одговорното и безбедното користење на современите технологии придонесува за заштита на личните податоци и на дигиталната приватност.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи за да најдеш одговор на прашањето „Како одредена апликација знае што сакам да видам во следниот момент?“. Пронаоѓањето на одговорот ја отвора вратата кон твојата иднина во светот на напредната информатика.





1.2

САЈБЕР БЕЗБЕДНОСТ И ЗАШТИТА НА ЛИЧНИТЕ ПОДАТОЦИ

Сајбер безбедноста претставува клучен сегмент во современиот дигитален свет, каде што заштитата на информациите станува сè поважна. Во услови на постојано зголемена употреба на интернет и дигитални услуги, личните податоци се изложени на различни ризици и злоупотреби. Затоа, зачувувањето на приватноста и обезбедувањето соодветна заштита на податоците се од суштинско значење за секој поединец и организација. Дигиталните закани, како што се хакерски напади, вируси и фишинг, претставуваат сериозна опасност за безбедноста на информациите. Разбирањето на овие предизвици и примената на основни принципи за заштита овозможуваат побезбедно користење на дигиталните технологии. Така, сајбер безбедноста станува неопходна вештина за функционирање во современото општество.

ВЕЌЕ ЗНАЕШ ДА...

- користиш интернет за пребарување информации;
- комуницираш преку дигитални платформи;
- користиш различни кориснички сметки;
- споделуваш дигитални содржини;
- препознаваш дека личните податоци треба да бидат заштитени.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Колку кориснички сметки користиш во текот на една седмица?
2. Дали некогаш си размислувал/а кои лични податоци ги оставаш при користење на интернет?
3. Како би постапил/а ако непознато лице ти ја побара лозинката или твоите лични информации?
4. Дали сите информации што се објавуваат на интернет треба да бидат споделени со други луѓе?
5. Зошто е важно да се биде внимателен при користење на дигитални услуги?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш зошто е важно да се заштитат личните и дигиталните податоци.
- опишуваш основни мерки за заштита (силни лозинки, антивирус, внимателно користење интернет).
- применуваш правила за безбедно користење на интернет во дадена ситуација (за да ги зашти-тиш своите податоци при користење социјални мрежи или е-пошта).
- разликуваш безбедни од небезбедни онлајн однесувања и да анализираш можни последици од несоодветна заштита на податоци.
- предлагаш начини за подобрување на личната дигитална безбедност.

Современите информатички технологии овозможуваат брза комуникација, пристап до голем број информации и користење различни дигитални услуги. Секојдневно се користат електронска пошта, социјални мрежи, платформи за учење, електронско банкарство и различни мобилни апликации. При нивното користење често се внесуваат лични податоци како: име и презиме, телефонскиот број, адресата на електронската пошта, лозинки и други информации. Поради тоа, безбедното користење на дигиталните технологии станува сè поважно за секој корисник.

Со развојот на современите технологии се појавуваат и различни ризици што може да влијаат врз безбедноста на податоците и врз приватноста на корисниците. Има постојани обиди за кражба на лични информации, неовластен пристап до кориснички сметки и ширење штетни програми. За да бидат намалени овие ризици, потребно е да се познаваат основните правила на сајбер безбедноста и да се применуваат мерки за заштита на личните податоци при секојдневното користење на дигиталните технологии.

1.2.1 ШТО ПРЕТСТАВУВА САЈБЕР БЕЗБЕДНОСТ И ЗОШТО Е ВАЖНА?

Секогаш кога се приклучуваш на интернет, без разлика дали прегледуваш социјални мрежи, пракаш е-пошта или играш онлајн игри, всушност ја отвораш вратата кон дигиталниот свет. Но, исто како што го заклучуваш домот за да се заштитиш од крадци, така треба да се грижиш и за безбедноста на твојот дигитален простор.

Сајбер безбедноста претставува збир од мерки, правила и постапки што овозможуваат заштита на компјутерските системи, мрежите и дигиталните податоци од различни закани и злоупотреби. Во современото општество голем број активности зависат од информатичките технологии и од интернетот. Користењето електронска пошта, социјални мрежи, дигитални платформи за учење, електронско банкарство и различни мобилни апликации подразбира постојана размена на информации преку дигитални мрежи. Поради тоа, безбедноста на овие системи има значајна улога во заштитата на корисниците и нивните податоци.

Сајбер безбедноста не се однесува само на заштита на големи компании и институции туку и на секој поединечен корисник. Секојдневното користење на паметни телефони, таблети и компјутери подразбира одредено ниво на одговорност при чувањето на личните информации. Со примена на соодветни мерки може да се намалат ризиците од неовластен пристап до податоците, кражба на идентитетот или злоупотреба на корисничките сметки. Познавањето на основните принципи на сајбер безбедноста придонесува за побезбедно користење на современите технологии.

1.2.2 ЛИЧНИ ПОДАТОЦИ И ДИГИТАЛНА ПРИВАТНОСТ

Личните податоци претставуваат информации преку кои може да биде идентификувано едно лице. Во оваа група спаѓаат името и презимето, адресата на живеење, телефонскиот број, адресата на електронската пошта, матичниот број, фотографиите и многу други информации што може да бидат поврзани со одреден корисник. Во дигиталното општество секојдневно се оставаат различни лични податоци при користење интернет-услуги, мобилни апликации и електронски платформи. Поради тоа, нивната заштита претставува важен дел од дигиталната безбедност.

Дигиталната приватност претставува право на секој корисник самостојно да одлучува кои информации ќе ги сподели и со кого ќе ги сподели. При користење интернет потребно е внимателно да се избираат податоците што се објавуваат на социјалните мрежи или на други дигитални платформи. Прекумерното споделување лични информации може да доведе до различни злоупотреби и да ја загрози безбедноста на корисникот. Одговорното користење на дигиталните услуги придонесува за подобра заштита на приватноста и за намалување на можностите за злоупотреба на личните податоци.

1.2.3 ПРАВИЛА ЗА БЕЗБЕДНО КОРИСТЕЊЕ ИНТЕРНЕТ

За да бидеш безбеден на интернет, не е потребно напредно информатичко знаење. Доволно е да ги практикуваш трите златни правила на дигиталната хигиена:

- **Креирање силни лозинки:** Лозинката е првата линија на одбрана. Слаба лозинка (како 123456 или ime123) хакерите можат да ја откријат за неколку секунди. Силната лозинка треба да има **најмалку 12 карактери**, да содржи комбинација од големи и мали букви, бројки и специјални знаци (на пр. @, #, \$, %). Дополнително, никогаш не треба да се користи иста лозинка за повеќе различни профили.
- **Користење и ажурирање на антивирус софтвер:** Антивирусните програми делуваат како дигитални чувари кои го скенираат уредот, откриваат сомнителен софтвер (вируси, тројанци, шпионски програми) и го блокираат пред да направи штета. Важно е овој софтвер редовно да се ажурира за да препознае нови закани.
- **Внимателно користење на интернетот:** Ова подразбира свесност за тоа каде кликуваш. Избегнувај преземање датотеки од непознати веб-страници, не кликувај на сомнителни реклами (што ветуваат брза заработка или бесплатни награди) и не се поврзувај на јавни, незаштитени Wi-Fi мрежи (како во кафулиња или во паркови) кога внесуваш чувствителни податоци.

1.2.4 НАЈЧЕСТИ ДИГИТАЛНИ ЗАКАНИ И РИЗИЦИ

Со развојот на интернетот и современите технологии се појавуваат различни дигитални закани што можат да влијаат врз безбедноста на корисниците. Меѓу најчестите закани се лажните пораки и интернет-измами, штетните програми, неовластениот пристап до корисничките сметки и кражбата на лични податоци. Во одредени ситуации напаѓачите се обидуваат да стекнат доверба кај корисниците и да ги наведат доброволно да ги откријат своите лозинки или други чувствителни информации. Ваквите обиди најчесто се реализираат преку електронска пошта, пораки или лажни интернет-страници.

Покрај директните напади, постојат и ризици што произлегуваат од невнимателното користење на дигиталните технологии. Користење слаби лозинки, преземање датотеки од непроверени извори или отворање сомнителни пораки може да доведе до загрозување на безбедноста на компјутерските системи и на личните податоци. Препознавањето на можните закани и внимателното користење на интернет-услугите претставуваат важен чекор во заштитата на дигиталниот идентитет и намалувањето на потенцијалните ризици.

1.2.5 БЕЗБЕДНО И ОДГОВОРНО КОРИСТЕЊЕ НА ИНТЕРНЕТОТ

Безбедното користење на интернетот подразбира примена на различни правила и навики што придонесуваат за заштита на корисниците и на нивните податоци. Една од основните мерки е користењето сигурни лозинки што содржат комбинација од букви, броеви и специјални знаци. Корисно е истата лозинка да не се употребува за повеќе различни услуги и редовно да се менува. Дополнителна заштита обезбедува и користењето повеќекратна проверка на идентитетот, при што за пристап до корисничката сметка е потребна дополнителна потврда.

Одговорното користење на интернетот подразбира и внимателно однесување при споделување информации и користење дигитални услуги. Потребно е да се проверува веродостојноста на интернет-страниците, внимателно да се отвораат електронски пораки и да не се споделуваат лични податоци со непознати лица. Редовното ажурирање на оперативните системи и програмите придонесува за подобра заштита од нови безбедносни закани. Со развивање добри дигитални навики и критичко размислување, корисниците можат значително да ја подобрат својата безбедност при користење на современите информатички технологии.

Најдобар начин да се разбере сајбер безбедноста е преку нејзина примена во секојдневните ситуации со кои се соочуваме.

Пример 1: Користење социјални мрежи (TikTok, Instagram, Snapchat...)

Социјалните мрежи се место каде што лесно се споделуваат премногу информации. За заштита на податоците во оваа ситуација, треба да се применат следните правила:

- **Личен профил (Private Account):** Измени ги поставките за приватност така што само прифатените пријатели ќе можат да ги гледаат твоите објави, слики и локација.
- **Внимателно со пријавување „check-in“ (Локација):** Не ја споделувај твојата точна локација во реално време. Тоа им кажува на злонамерните лица каде точно се наоѓаш (или дека во моментот не си дома).
- **Размисли пред да објавиш:** Секоја слика или информација (на пример, слика од лична карта, куќна адреса или училиште) останува на интернет засекогаш, дури и ако ја избришеш.

Пример 2: Користење електронска пошта (е-пошта)

Е-поштата е најчестата мета за т.н. **фишинг (phishing) напади**, измами преку кои напаѓачите се обидуваат да ти ги украдат лозинките или податоците од банкарски картички лажно претставувајќи се како легитимни компании (банка, социјална мрежа, пошта). За заштита во оваа ситуација, се применуваат следниве правила:

1. **Игнорирање на непознати испраќачи:** Ако добиеш е-пошта од непозната адреса со која се бара итен одговор (на пр. „Твојот профил ќе биде избришан, кликни тука за да ја потврдиш лозинката“), веднаш избриши ја.
2. **Проверка на линкови и прикачени фајлови:** Никогаш не отворај прикачени документи со наставки како .exe или .zip од непознати луѓе, затоа што тие можат автоматски да инсталираат вирус во твојот компјутер.

Пример 3: Вклучување на двофакторна автентикација (2FA): Ова претставува дополнителен заштитен слој. При најава на електронската пошта, покрај лозинката, системот бара и код кој ти пристигнува како СМС на твојот телефон. На тој начин, дури и некој да ја дознае лозинката, нема да може да го користи твојот профил без твојот телефон.

1.2.6 ДИГИТАЛНИОТ УЧЕНИК И САЈБЕР БЕЗБЕДНОСТА

Во текот на еден обичен училиштен ден ученикот користи голем број дигитални услуги. Се најавува на електронски платформи за учење, пребарува информации на интернет, комуницира преку различни апликации и користи електронска пошта. При сите овие активности се разменуваат различни податоци, а за пристап до услугите најчесто се користат кориснички сметки и лозинки. Безбедното користење на овие услуги придонесува за заштита на личните податоци и за намалување на можностите за злоупотреба.

Ситуацијата во која ученик добива порака дека освоил награда и дека е потребно да ги внесе своето корисничко име и лозинка и банкарската сметка на непозната интернет-страница. Ако не се провери испраќачот и адресата на страницата, може да дојде до злоупотреба на корисничката сметка. Внимателното однесување, проверката на изворот на информацијата и избегнувањето споделување лични податоци со непознати лица претставуваат едноставни, но важни мерки за подобрување на дигиталната безбедност.

Онлајн однесувањето зависи во голема мера со какви последици ќе се соочиш во небезбедниот интернет-простор. Секој чекор што го правиш во дигиталниот свет остава трага. Честопати, поради брзината на секојдневието, забораваме дека интернетот не е само игралиште туку е и простор полн со скриени ризици. Разликата помеѓу безбедното сурфање и потенцијалната дигитална катастрофа најчесто зависи од нашите сопствени одлуки и навики.

1.2.7 БЕЗБЕДНИ НАСПРОТИ НЕБЕЗБЕДНИ ОНЛАЈН ОДНЕСУВАЊА

За да се заштитиш, прво мора да знаеш да ги препознаеш ризичните ситуации. Онлајн однесувањето генерално може да се подели на две категории:

Небезбедно однесување (Ризик)	Безбедно однесување (Заштита)
Користење отворени јавни Wi-Fi мрежи (без лозинка) за најавување на социјални мрежи или е-банкарство.	Користење мобилен интернет (4G/5G) или заштитена домашна мрежа при пристап до чувствителни профили.
Прифаќање барања за пријателство од непознати лица и споделување приватни детали со нив.	Комуникација само со луѓе кои ги познавате во реалниот живот и чување на приватноста.
Кликање на привлечни линкови кои ветуваат бесплатни награди, телефони или „наградни игри“ (Phishing).	Игнорирање на сомнителни пораки и проверка на официјалните веб-страници за какви било награди.
Преземање и инсталирање пиратски игри, филмови или програми од несигурни софтвери и торент-страници.	Преземање софтвер исклучиво од официјални веб-страници и онлајн продавници (Google Play, App Store, официјални сајтови).

Последиците од невнимателното однесување и лошата заштита може да бидат сериозни и долготрајни. Ако твоите лични податоци (лозинки, слики, матични броеви) се откриени од други лица, можни се следните сценарија:

- **Кражба на идентитетот:** Хакерите можат да ги искористат твоите податоци и слики за да креираат лажни профили, да прават измами во твоето име или да ги уценуваат твоите пријатели или семејството барајќи пари.
- **Финансиска штета:** Преку фишинг пораки или небезбедни плаќања, напаѓачите можат да добијат пристап до трансакциските сметки или до кредитните картички на твоите родители, со што можат да ја испразнат сметката за неколку минути.
- **Зараза со уценувачки софтвер (Ransomware):** Наместо обичен вирус, на уредот може да се инсталира софтвер кој ќе ги заклучи сите твои документи, слики и проекти, а хакерите ќе бараат да им платиш откуп за да ти ги вратат.
- **Нарушување на емоционалното здравје и репутацијата:** Пробивањето на приватноста и јавното објавување на приватни разговори или слики (сајбер-булинг) предизвикува огромен стрес, срам и долгорочни последици за социјалниот живот на личноста.

Подобрувањето на заштитата не бара сложени технички вештини, ами воведување добри дигитални навики. Еве неколку конкретни предлози кои можеш да ги имплементираш веднаш:

1. **Активирај двофакторна автентикација (2FA):** Ова е најважниот чекор. Покрај лозинката, за најава на твојот профил (Instagram, TikTok, Google, итн.) секогаш ќе биде потребен и дополнителен код кој стигнува на твојот телефон. Дури и некој да ја знае лозинката, нема да може да се најави со твојот профил.
2. **Користи менаџер на лозинки (password manager):** Наместо да ја користиш истата лесна лозинка насекаде (што е огромна грешка), користи апликација која ќе генерира и безбедно ќе ги чува различните, комплексни лозинки за секој твој профил.
3. **Редовно прави резервна копија (Backup):** Еднаш неделно или месечно префрли ги твоите важни документи и слики на надворешен хард диск, УСБ стапче или на сигурен облак (Cloud). На

тој начин, дури и да го украдат или да го расипат уредот, податоците ќе останат безбедни и ќе може да се користат повторно.

4. **Редовно ажурирање на системот:** Не ги одложувај известувањата за ажурирање на оперативниот систем на телефонот или на компјутерот. Овие ажурирања често содржат „закрпи“ за новооткриените безбедносни пропусти, кои хакерите ги користат за напад.



ДАЛИ ЗНАЕШ?

Дали знаеш дека силната лозинка треба да биде доволно долга и да содржи комбинација од големи и мали букви, броеви и специјални знаци? Употребата на различни лозинки за различни дигитални услуги значително ја подобрува безбедноста на корисничките сметки.



ИСТРАЖИ

Направи истражување со кое ќе дознаеш кои правила за безбедно користење на интернет ги применува твоето училиште, твоите другари или твоето семејство. Врз основа на резултатите направи листа од најмалку десет правила што придонесуваат за подобра заштита на личните податоци и на дигиталната приватност.



ЗАКЛУЧОК:

Сајбер безбедноста не зависи од сложеноста на компјутерите, туку од нашето однесување. Со примена на силни лозинки, антивирус и со малку претпазливост при секое кликување, стануваме заштитени и одговорни корисници на дигиталниот свет. Интернет-просторот е безбеден онолку колку што сме и ние претпазливи. Со развивање на свесноста за опасностите и примена на едноставни заштитни механизми, самите стануваме креатори на сопствената безбедност и ја преземаме контролата врз нашиот дигитален живот.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Зошто сајбер безбедноста има значајна улога во современото општество?
2. Кои информации се сметаат за лични податоци?
3. Како може да биде загрозувана дигиталната приватност на корисникот?
4. Кои се најчестите ризици при користење на интернет и на дигиталните услуги?
5. На кој начин добрите дигитални навики придонесуваат за подобра заштита на личните податоци?



ИСТРАЖУВАЧКИ И ПРАКТИЧНИ ЗАДАЧИ

1. Направи листа на лични податоци што најчесто се користат при отворање корисничка сметка на интернет. Објасни зошто е важно тие податоци да бидат заштитени.
2. Истражи кои правила ги применуваат најпознатите интернет-платформи за заштита на корисничките сметки и кои лични податоци ги прибираат и обработуваат. Подготви краток извештај.
3. Размисли и опиши три ситуации во кои невнимателното користење на интернетот може да доведе до злоупотреба на личните податоци.

4. Во група со соучениците дискутирај за предностите и за ризиците од споделување фотографии и лични информации на социјалните мрежи. Запиши ги заклучоците.
5. Подготви краток водич со десет совети за безбедно и одговорно користење интернет што би можел да им биде корисен на учениците од твоето училиште.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Сајбер безбедноста е една од најбрзо развиваните области во информатиката. Голем број државни институции, компании, банки, болници и образовни установи секојдневно применуваат современи безбедносни системи за заштита на своите податоци и услуги. Постојат различни професии поврзани со оваа област, како што се: аналитичар за сајбер безбедност, експерт за заштита на мрежи, истражувач на дигитални закани и специјалист за дигитална форензика. Со развојот на современите технологии, потребата од стручни лица што ќе придонесуваат за заштита на дигиталниот простор постојано се зголемува.



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

Заокружи го точниот одговор.

1. Сајбер безбедноста се однесува на:

- а) изработка на компјутери;
- б) заштита на дигиталните системи и податоци;
- в) поправка на мобилни телефони;
- г) производство на компјутерски игри.

2. Објасни ги поимите.



Поим	Објаснување
Лични податоци	_____
Дигитална приватност	_____
Сајбер безбедност	_____

3. Дополни.

Безбедната лозинка треба да биде _____.

4. Кои се трите клучни карактеристики што ги содржи една силна и безбедна лозинка?

5. Објасни со свои зборови како функционира „фишинг“ (phishing) нападот и која е неговата главна цел.

6. Ако добиеш е-пошта од непознат испраќач со прикачен фајл со наставка .exe во кој се тврди дека е бесплатна видеоигра, кои конкретни чекори ќе ги преземеш за да се заштитиш?

7. Спореди го користењето на јавна Wi-Fi мрежа во кафуле со користењето на сопствениот мобилен интернет (4G/5G). Како влијае изборот на мрежа врз безбедноста на твоите лични податоци?

8. Процени ги можните последици од кражба на идентитетот на социјалните мрежи. Што мислиш, дали се последиците поопасни во финансиска или во емоционална/социјална смисла за еден тинејџер? Образложи го ставот.

9. Предложи три основни правила што треба да ги содржи еден училиштен „Кодекс за безбедно онлајн однесување“, со цел да се заштитат податоците на учениците во твоето училиште.

1.3



КРИПТОГРАФИЈА

Секојдневно испраќаме многубројни пораки, пазаруваме онлајн и се најавуваме на различни платформи. Но, дали некогаш си се запрашал/а како „патуваат“ овие податоци низ светот без некој да ги украде или да ги прочита? Одговорот лежи во една древна, но денес ултрамодерна наука – криптографијата.

Криптографијата е наука за безбедна комуникација која овозможува заштита на информациите од неовластен пристап во дигиталниот свет. Во нејзиното јадро лежи процесот на шифрирање, со кој обичниот и читлив текст се претвора во неразбирлив код со помош на математички алгоритми. За да стане пораката повторно разбирлива за примачот, се користи обратниот процес наречен дешифрирање. Во зависност од начинот на кој се користат клучовите, таа се дели на симетрична криптографија, каде што еден ист таен клуч служи и за заклучување и за отклучување на податоците. Наспроти тоа, асиметричната криптографија користи пар од два различни клуча, јавен клуч за шифрирање, кој е достапен за сите и приватен клуч – кој го поседува само примачот за дешифрирање.

ВЕЌЕ ЗНАЕШ ДА...

- објаснуваш зошто треба да бидат заштитени личните податоци;
- препознаваш безбедно и небезбедно користење на интернетот;
- објаснуваш што е дигитална приватност;
- наведуваш примери за безбедно користење на дигиталните технологии;
- анализираш можни ризици при размена на информации.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Како би испратил/а тајна порака до пријател без да можат други да ја прочитаат?
2. Дали луѓето во минатото имале потреба да ги заштитуваат своите информации?
3. Кои информации денес би сакал/а да останат доверливи?
4. Што мислиш, како банките и интернет-продавниците ги заштитуваат податоците на своите корисници?
5. Дали е можно само овластено лице да може да ја прочита испратената порака?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- дефинираш поим за криптографија со клучни основни поими: шифрирање, клуч, тајна порака;
- препознаваш примена на криптографијата во конкретна секојдневна ситуација;
- објаснуваш како се користи криптографијата при заштита на податоците;
- разликуваш заштитени од незаштитени информации;
- го објасниш значењето на криптографијата за приватноста и за безбедноста.

Комуникацијата и размената на информации отсекогаш претставувале важен дел од секојдневниот живот на луѓето. Во различни историски периоди имало потреба од испраќање пораки што требало да останат тајни и да бидат разбрани само од лицата за кои биле наменети. Воени планови, дипломатски договори и важни државни информации честопати биле заштитувани со различни методи за прикривање на нивната содржина. Потребата од заштита на информациите довела до развој на различни техники за шифрирање на пораките.

Со развојот на современите технологии криптографијата станува важен дел од секојдневниот живот. Денес таа не се користи само во воени и во државни системи туку и при користење банкарски услуги, електронска пошта, интернет-продавници и мобилни апликации. Благодарение на криптографските методи, чувствителните информации може безбедно да бидат разменети и заштитени од неовластен пристап.

1.3.1 ШТО ПРЕТСТАВУВА КРИПТОГРАФИЈАТА?

Криптографијата претставува научна област што се занимава со заштита на информациите од неовластен пристап. Основната цел на криптографијата е да овозможи одредена порака или податок да биде прочитан само од лицата за кои е наменет. За таа цел се применуваат различни постапки со кои разбирливата порака се претвора во облик што не може лесно да биде разбран од други лица. Лицето што ја испраќа пораката и лицето што ја прима треба да располагаат со потребните информации за правилно да ја прочитаат содржината. На овој начин се намалува можноста доверливите информации да бидат злоупотребени.

Криптографијата денес има примена во голем број области од секојдневниот живот. При користење електронско банкарство, интернет-продавници, мобилни апликации и различни дигитални услуги, личните податоци и финансиските информации се заштитуваат со криптографски методи. Исто така, електронската пошта, дигиталните сертификати и многубројни интернет-услуги користат различни техники за заштита на информациите. Благодарение на криптографијата, корисниците можат побезбедно да разменуваат податоци и да користат современи дигитални услуги. За да ја разбереме, мора да познаваме четири клучни поими: шифрирање, дешифрирање, криптографски клуч и тајна порака.

1.3.2 ШИФРИРАЊЕ, ДЕШИФРИРАЊЕ И КРИПТОГРАФСКИ КЛУЧ

Шифрирањето претставува постапка при која разбирливата порака се претвора во неразбирлив запис со цел да биде заштитена од неовластен пристап. Пораката што може да се прочита се нарекува изворна порака, а по извршеното шифрирање се добива шифрирана порака. Оваа порака изгледа како низа од случајни букви и знаци. Дури и некој да ја пресретне додека патува низ интернетот, таа за него е целосно бескорисна, затоа што не може да ја прочита. На пример: зборот „Здраво!“ се претвора во неразбирлив код, на пример: „Ху9#z!“. Ова се прави со помош на сложени математички правила (алгоритми).

Лицето што ја прима пораката треба да располага со соодветни информации за да може повторно да ја претвори во првобитната форма. Постапката на враќање на пораката во нејзиниот разбирлив облик се нарекува дешифрирање.

За успешно шифрирање и дешифрирање се користи криптографски клуч. Клучот претставува збир од правила или податоци што овозможуваат пораката правилно да биде заштитена и повторно прочитана. Во едноставните системи истиот клуч може да се користи и за шифрирање и за дешифрирање, додека кај посовремените системи се користат различни клучеви за овие две постапки. Без познавање на соодветниот клуч, шифрираната порака е многу потешко да биде разбрана од неовластени лица.

1.3.3 СОВРЕМЕНА ТЕХНОЛОГИЈА ВО ПРАКТИКАТА

Цезаровата шифра

Еден од најпознатите примери за шифрирање е Цезаровата шифра. Кај овој начин на заштита секоја буква од пораката се заменува со друга буква што се наоѓа неколку места подоцна во азбуката. Ако е поместувањето за три места, буквата А се заменува со Г, Б со Д, В со Ѓ и така натаму. Лицето што го знае правилото за поместување може лесно да ја прочита пораката, додека за неупатените таа изгледа како случајна низа од букви.

Цезаровата шифра денес нема практична примена за заштита на важни информации, затоа што нејзиното правило е едноставно и лесно може да биде откриено. Сепак, таа има големо образовно значење, бидејќи на едноставен начин ги објаснува основните идеи на криптографијата. Со нејзина помош може лесно да се разбере како функционираат шифрирањето, дешифрирањето и употребата на криптографски клуч.

Пример

Да претпоставиме дека поместувањето е за три букви.

Изворна порака	Шифрирана порака
А	Г
Б	Д
В	Ѓ
Г	Е

Ако е пораката **ИНТЕРНЕТ**, по шифрирањето ќе се добие нова низа од букви според избраното правило за поместување.

Процес на шифрирање (буква по буква):

И (+3 места подолу) = **Л**

Н (+3 места подолу) = **П**

Т (+3 места подолу) = **Х**

Е (+3 места подолу) = **З**

Р (+3 места подолу) = **У**

Н (+3 места подолу) = **П**

Е (+3 места подолу) = **З**

Т (+3 места подолу) = **Х**

Тајната порака (клуч 3) ќе биде следната низа од букви **ЛПХЗУПЗХ**



Современа криптографија

Со развојот на компјутерските системи и на интернетот, криптографијата добива многу поширока примена отколку во минатото. Денес голем број дигитални услуги користат современи криптографски методи за заштита на информациите што се разменуваат меѓу корисниците и информатичките системи. При најавување на корисничка сметка, испраќање електронска пошта, користење интернет-продавници или електронско банкарство, чувствителните информации се заштитуваат со различни криптографски постапки. На овој начин се намалува можноста податоците да бидат прочитани или да се изменат од неовластени лица.

Во современата криптографија се применуваат различни методи за заштита на информациите. Кај едни системи истиот криптографски клуч се користи за шифрирање и дешифрирање на пораките, додека кај други се користат два различни клуча. Изборот на методот зависи од потребното ниво на безбедност и од областа во која се применува. Современите криптографски алгоритми се создадени така што обезбедуваат високо ниво на заштита и овозможуваат безбедна комуникација меѓу корисниците и информатичките системи.

Примена на криптографијата во секојдневниот живот

Криптографијата е дел од многу дигитални услуги што секојдневно ги користат луѓето. При користење електронско банкарство, плаќање преку интернет или пристапување до различни веб-страници, личните и финансиските податоци се заштитуваат со криптографски механизми. Мобилните апликации за комуникација користат криптографија за да овозможат пораките да бидат прочитани само од лицата што учествуваат во разговорот. WhatsApp, Viber и Signal апликациите користат таканаречена end-to-end (од крај до крај) криптографија. Тоа значи дека пораката се шифрира на твојот телефон и се дешифрира дури кога ќе стигне кај твојот пријател. Дури ни компании (како Meta) не можат да ги прочитаат твоите разговори.

Исто така, многу електронски услуги користат дигитални сертификати што овозможуваат проверка на идентитетот на корисниците и на интернет-страниците. Криптографијата не е нешто што се користи само од страна на тајните служби. Таа е сеprisутна.

Примената на криптографијата придонесува за подобрување на безбедноста и довербата во современите информатички системи. Благодарение на криптографските методи, корисниците можат да користат различни дигитални услуги со поголема сигурност. Истовремено, важно е да се разбере дека криптографијата претставува само еден дел од целокупната дигитална безбедност. За постигнување повисоко ниво на заштита потребно е да се применуваат и други мерки, како што се користење сигурни лозинки, внимателно споделување информации и редовно ажурирање на информатичките системи.

Во текот на еден ден кој било ученик може повеќепати да користи криптографија без да биде свесен за тоа. При најавување на електронски дневник, испраќање порака преку мобилна апликација, купување производ преку интернет или користење електронско банкарство, податоците се заштитуваат со современи криптографски методи. На овој начин се намалува можноста личните информации да бидат злоупотребени.

Кога еден корисник пристапува до интернет-продавница и внесува податоци за плаќање, криптографските системи овозможуваат овие информации да бидат безбедно пренесени меѓу корисникот и серверот. Благодарение на тоа, чувствителните податоци не се испраќаат како обичен текст, туку во заштитена форма, што е многу потешко да биде прочитана од неовластени лица.

Како да се разликуваат заштитени од незаштитени информации?

Како ученик, многу е важно веднаш да направиш разлика дали информациите со кои ракуваш се безбедни:

- **Заштитени информации.** Тоа се податоци што се криптирани и до нив имаат пристап само овластени лица. Најлесно ќе ги препознаеш по иконата со **катанец** во прелистувачот и протоколот **HTTPS** (буквата S значи Secure / Безбедно). Внесувањето лозинки или лични податоци на вакви страници е заштитено.
- **Незаштитени информации.** Тоа се јавни информации што патуваат како обичен текст преку протоколот **HTTP** (без катанец). Исто така, незаштитени се сите информации што ги споделу-

ваш на јавни профили на социјалните мрежи, на отворени форуми или кога користиш јавен Wi-Fi во кафуле без лозинка. Секој со малку хакерско знаење може да ги пресретне.

Без криптографија, современиот дигитален свет би престанал да функционира. Нејзиното значење е огромно поради две причини:

- **Заштита на приватноста;** таа ни овозможува да имаме личен простор во дигиталниот свет. Ни дава слобода да комуницираме било со пријателите, семејството или со лекарите, знаејќи дека нашите приватни мисли, слики и тајни остануваат само наши.
- **Обезбедување дигитална безбедност;** таа ги штити државните институции, болниците, училиштата и банките од сајбер-напади. Криптографијата спречува злонамерни луѓе да ги менуваат туѓите оценки во е-дневникот, да крадат идентитети или да ги исклучат системите на една цела држава.



ЗАКЛУЧОК:

Криптографијата е невидливиот чувар на нашите дигитални животи. Разбирањето на нејзините основи помага да бидеме попаметни корисници кои знаат како да го заштитат својот сопствен дигитален отпечаток.



ДАЛИ ЗНАЕШ?

Дали знаеш дека секојпат кога на интернет-страницата ќе забележиш симбол на катанец покрај адресата, комуникацијата на твојот уред и страницата е заштитена со современи криптографски технологии?



ИСТРАЖИ

Истражи во кои секојдневни активности користиш криптографија без да бидеш свесен за тоа. Направи листа од најмалку пет примери и објасни каква улога има криптографијата во секој од нив.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Прашања за повторување
2. Што претставува криптографијата?
3. Која е разликата помеѓу шифрирање и дешифрирање?
4. Каква улога има криптографскиот клуч?
5. Зошто Цезаровата шифра има образовно значење и денес?
6. Во кои секојдневни дигитални услуги се применува криптографијата?
7. Зошто луѓето имаат потреба да ги заштитуваат своите информации?
8. Како се разликуваат шифрирањето и дешифрирањето?
9. Каква улога има криптографскиот клуч во заштитата на информациите?
10. Зошто Цезаровата шифра денес има претежно образовно значење?
11. Во кои современи дигитални услуги секојдневно се применува криптографијата?



ИСТРАЖУВАЧКИ И ПРАКТИЧНИ ЗАДАЧИ

1. Со помош на правилото на Цезаровата шифра создај тајна порака за твојот соученик. Потоа разменете ги пораките и обидете се да ги дешифрирате.
2. Истражи во кои секојдневни активности користиш криптографија без да бидеш свесен/свесна за тоа. Наведи најмалку пет примери.
3. Спореди ја Цезаровата шифра со современите методи за заштита на информации. Наведи најмалку три разлики.
4. Размисли зошто банките, интернет-продавниците и мобилните апликации користат криптографија. Подготви краток текст со твои согледувања.
5. Истражи што претставува дигитален сертификат и објасни зошто е важен за безбедната комуникација преку интернет.



ЗАПОМНИ ШТО НАУЧИ

Криптографијата претставува наука за заштита на информациите.

Шифрирањето ја претвора разбирливата порака во заштитен запис.

Дешифрирањето овозможува повторно читање на пораката.

Криптографскиот клуч има важна улога во заштитата на информациите.

Криптографијата има широка примена во современите информатички системи и при секојдневните дигитални услуги.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Современата криптографија е една од најважните области во информатиката и има значајна улога во функционирањето на интернетот. Освен за заштита на пораки и финансиски податоци, криптографските методи се применуваат при дигитално потпишување документи, изработка на дигитални сертификати, заштита на компјутерски мрежи и функционирање на blockchain технологијата. Со развојот на современите информатички системи, криптографијата постојано се усовршува за да обезбеди повисоко ниво на безбедност и доверливост на информациите.





ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

1. Заокружи го точниот одговор.

Криптографијата се користи за:

- а) забрзување на компјутерите;
- б) заштита на информациите;
- в) поправка на компјутерски мрежи;
- г) изработка на мобилни апликации.

2. Дефинирај го поимот криптографија, задолжително користејќи ги во дефиницијата трите клучни поими: шифрирање, клуч и тајна порака

3. Точно или неточно.

Шифрирањето и дешифрирањето претставуваат иста постапка.

4. Објасни ги поимите.

Поим	Објаснување
Шифрирање	_____
Дешифрирање	_____
Криптографски клуч	_____

5. Дополни.

Еден од најпознатите едноставни начини на шифрирање е _____.

6. Како помага криптографијата практично во заштита на податоците додека тие патуваат низ интернетот? Објасни го процесот на претворање на обичниот текст во шифртекст и обратно.

7. Наведи една конкретна ситуација од твојот секојдневен живот (на пример, при користење апликации за комуникација или онлајн купување) каде што несвесно користиш криптографија и опиши што се случува со твоите податоци во тој момент.

8. Направи споредба помеѓу заштитени и незаштитени информации на интернет. По кои дигитални знаци (симболи или протоколи во веб-прелистувачот) учениците може веднаш да препознаат дали една веб-страница ги штити или не ги штити нивните податоци?

9. Прцени го значењето на криптографијата за општеството: Што би се случило со приватноста на граѓаните и безбедноста на институциите (како банките и болниците) ако криптографијата одеднаш исчезне од дигиталниот свет? Образложи ги последиците.

10. Замисли дека треба да им објасниш на ученици на помала возраст зошто е важно да ги штитат своите податоци. Создај три едноставни „златни правила“ за безбедно пребарување што се засновуваат на разликувањето на заштитени од незаштитени информации на интернет.



ПРОБЛЕМСКА ЗАДАЧА

Замисли дека треба да испратиш доверлива порака до пријател/ка преку интернет. Објасни зошто е важно пораката да биде шифрирана и кои придобивки ги обезбедува криптографијата.



1.4

BLOCKCHAIN ТЕХНОЛОГИЈА – ДОВЕРЛИВИ ДИГИТАЛНИ ЗАПИСИ ВО СОВРЕМЕНИОТ СВЕТ

Blockchain технологијата претставува современо информатичко решение што овозможува сигурно чување и следење на дигиталните записи. Благодарение на криптографските методи и начинот на организација на податоците, секоја промена останува евидентирана и лесно може да биде проверена. Поради овие својства, blockchain технологијата наоѓа примена во различни области од современото општество, а нејзината употреба постојано се проширува.

ВЕЌЕ ЗНАЕШ ДА...

- објаснуваш што претставува криптографијата;
- ги разликуваш шифрирањето и дешифрирањето;
- ја објаснуваш улогата на криптографскиот клуч;
- наведуваш примери за заштита на информации;
- препознаваш примена на криптографијата во современите информатички системи.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Како може важни информации да бидат зачувани така што секоја промена ќе остане евидентирана?
2. Дали е можно една база на податоци да биде користена од голем број корисници без еден централен управувач?
3. Зошто е важно да се знае кој и кога направил одредена промена во некој запис?
4. Дали технологија што се користи за криптовалути може да има примена и во други области?
5. Како современите информатички технологии можат да придонесат за поголема доверба во дигиталните услуги?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што претставува blockchain технологијата;
- ја објаснуваш структурата на блок и поврзувањето на блоковите;
- го објасниш принципот на децентрализација;
- наведуваш примери за примена на blockchain технологијата.

Во современото општество секојдневно се создаваат голем број дигитални записи што треба безбедно да бидат зачувани и заштитени од неовластени промени. Банкарски трансакции, податоци за сопственоста, здравствени картони, договори и различни електронски документи претставуваат ин-

формации што треба да бидат точни и доверливи. Доколку овие записи може лесно да бидат изменети или избришани, може да настанат сериозни проблеми и недоразбирања.

1.4.1 ШТО ПРЕТСТАВУВА BLOCKCHAIN ТЕХНОЛОГИЈАТА?

Во современото општество постои потреба важните информации да бидат сигурно зачувани и достапни за проверка. При купување недвижен имот, пренос на парични средства, издавање дипломи или склучување договори, потребно е да постои точна евиденција за извршените активности. Ако може записите лесно да бидат изменети или избришани, може да настанат и големи проблеми и недоразбирања. Поради тоа, современите информатички технологии нудат решенија што овозможуваат сигурно чување и следење на дигиталните записи.

Blockchain технологијата претставува посебен начин на организирање и чување на податоците. Наместо информациите да бидат зачувани во еден централен запис, тие се организираат во повеќе поврзани блокови. Секој нов блок содржи нови информации и е поврзан со претходниот блок во синџир. Благодарение на ваквата организација и примената на криптографски методи, податоците се заштитени, а секоја промена останува евидентирана. Поради тоа, blockchain технологијата претставува сигурен начин за чување различни видови дигитални записи.

Блокови, трансакции и децентрализација

Основниот елемент на blockchain технологијата е блокот. Блокот претставува дигитална целина во која се зачувуваат одредени информации. Во зависност од примената на системот, блокот може да содржи податоци за финансиски трансакции, за промена на сопственоста, склучени договори или други важни записи. Откако ќе се создаде нов запис, се формира нов блок што се поврзува со претходниот и станува дел од синџирот на блокови. На овој начин се создава последователна историја на сите настани што се случиле во системот.

Blockchain технологијата често се поврзува со поимот децентрализација. Кај традиционалните системи најчесто има еден централен управувач што ги чува и ги контролира податоците. Кај blockchain системите записите може да се достапни за повеќе поврзани учесници, при што секоја нова промена се евидентира и може да биде проверена. Ваквиот начин на организација придонесува за поголема транспарентност и доверба во системот, затоа што историјата на записите останува зачувана и лесно може да биде следена.

Приказ на поврзување на блоковите



Секој нов блок се поврзува со претходниот и станува дел од заедничкиот синџир на записи.



Каде се чуваат блоковите?

Една од најважните карактеристики на blockchain технологијата е начинот на чување на податоците. Наместо сите записи да бидат сместени на еден централен компјутер или сервер, копии од blockchain системот може да се чуваат на повеќе поврзани компјутери што учествуваат во неговото функционирање. Овие компјутери се нарекуваат јазли (анг. nodes) и во секој од нив има копија од синџирот на блокови. Кога ќе се додаде нов блок во системот, информацијата се евидентира и се ажурира кај сите учесници во мрежата. Благодарение на ваквиот начин на организација, податоците не зависат од едно место за чување и нивната сигурност и достапност се значително подобрени.

Чувањето на блоковите на повеќе поврзани компјутери придонесува за поголема доверливост и отпорност на системот. Ако еден компјутер престане да работи или ако се појави технички проблем, останатите компјутери и натаму располагаат со зачуваните податоци. Истовремено, секоја нова промена треба да биде евидентирана во системот, што овозможува полесна проверка на точноста на записите. На овој начин blockchain технологијата обезбедува сигурно чување на информациите и создава доверба меѓу учесниците што ја користат мрежата.

1.4.2 BLOCKCHAIN ТЕХНОЛОГИЈАТА ВО СОВРЕМЕНИОТ СВЕТ

Една од најпознатите примени на blockchain технологијата се криптовалутите. Криптовалутите претставуваат дигитални средства за размена што користат blockchain за сигурно евидентирање на трансакциите. Секоја трансакција се запишува во блок и станува дел од синџирот на записи. Благодарение на тоа, може да се следи историјата на трансакциите и да се намали можноста за неовластени измени. Bitcoin претставува една од најпознатите криптовалути, но денес постојат и многу други системи што ја користат оваа технологија. Blockchain технологијата не се користи само за криптовалути. Таа наоѓа примена во здравството, во логистиката, образованието, јавната администрација и при управувањето со различни дигитални записи.

Предности и можности на Blockchain технологијата

Blockchain технологијата обезбедува повеќе предности при чувањето и обработката на дигиталните записи. Благодарение на поврзаноста на блоковите, секоја промена останува евидентирана и може да биде проверена. Ова придонесува за поголема транспарентност и доверба во системот. Дополнително, примената на криптографски методи обезбедува подобра заштита на информациите и ја намалува можноста за неовластени измени на податоците.

Покрај многубројните предности, примената на blockchain технологијата носи и одредени предизвици. За нејзино успешно користење потребна е соодветна техничка инфраструктура и добра организација на системите што ја применуваат. Исто така, различните области на примена имаат различни потреби и барања, па изборот на технологијата зависи од конкретната намена. Со развојот на информатиката и дигиталната трансформација, се очекува blockchain технологијата да наоѓа примена во сè поголем број современи информатички системи и услуги.



1.4.3 СОВРЕМЕНА ТЕХНОЛОГИЈА ВО ПРАКТИКАТА

Blockchain технологијата и катастарските евиденции

Во многу држави катастарските служби водат евиденција за сопственоста на земјиштето и недвижностите. При купување, продажба, наследување или подарување имот, потребно е секоја промена да биде точно евидентирана. Ако не се ажурирани записите или ако може лесно да бидат изменети, може да настанат спорови околу сопственоста и да се создадат сериозни правни проблеми. Поради тоа, различни држави истражуваат современи информатички решенија што ќе овозможат подобра сигурност и транспарентност на катастарските податоци.

Во Африка постојат иницијативи за примена на blockchain технологијата при управување со катастарските евиденции и земјишните записи. Една од државите во кои се развиваат вакви проекти е Гана, каде што современите информатички решенија се разгледуваат како можност за подобрување на евиденцијата на сопственоста. Со помош на blockchain технологијата може да се зачува историјата на сите промени на еден имот, така што секоја продажба, наследување или друг вид пренос на сопственоста останува трајно евидентиран. Ваквиот пристап придонесува за поголема доверба во податоците и за намалување на можностите за злоупотреби.



ДАЛИ ЗНАЕШ?

Дали знаеш дека blockchain технологијата денес не се користи само за криптовалути? Освен во финансискиот сектор, таа се применува или се тестира во здравството, во образованието, транспортот, логистиката и при издавањето дигитални документи.



ИСТРАЖИ

Истражи три области во кои се применува blockchain технологијата. За секоја област наведи:

- што се евидентира;
- какви придобивки обезбедува blockchain;
- на кој начин придонесува технологијата за подобра сигурност на податоците.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува blockchain технологијата?
2. Што претставува блок во blockchain системот?
3. Што претставува трансакција?
4. Што значи децентрализација?
5. Зошто криптовалутите претставуваат една од најпознатите примени на blockchain технологијата?
6. Наведи најмалку три области во кои може да се примени blockchain технологијата.



ИСТРАЖУВАЧКИ И ПРАКТИЧНИ ЗАДАЧИ

1. Замисли дека во твоето училиште треба да се чуваат електронски дипломи. Објасни на кој начин blockchain технологијата би можела да помогне во нивната евиденција.
2. Истражи една криптовалута по твој избор и подготви краток извештај за нејзината примена и за основните карактеристики.
3. Подготви табела во која ќе споредиш класична база на податоци и blockchain систем според начинот на чување на информациите.
4. Размисли во кои области во Република Северна Македонија би можела да се примени blockchain технологијата. Наведи најмалку три примери и образложи го својот избор.
5. Истражи на кој начин blockchain технологијата може да придонесе за подобрување на катастарските евиденции и за управувањето со сопственоста на недвижностите.



ЗАКЛУЧОК:

Blockchain технологијата претставува начин за сигурно организирање и чување на дигитални записи. Основни елементи на blockchain системот се блоковите и нивното поврзување во синџир. Трансакцијата претставува запис што се внесува во системот. Децентрализацијата овозможува записите да бидат достапни и проверливи од повеќе учесници. Криптовалутите претставуваат една од најпознатите примени на blockchain технологијата. Таа има примена и во многу други области од современото општество.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Blockchain технологијата постојано се развива и навлегува во нови области на примена. Освен во финансискиот сектор, се користи за следење на производите во синџирите на снабдување, за издавање дигитални сертификати, за управување со медицински податоци и за заштита на интелектуалната сопственост. Во многу држави и организации се истражуваат нови начини за нејзина примена, со цел да се подобрат сигурноста, транспарентноста и доверливоста на различните информатички системи.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

1. Заокружи го точниот одговор.

Blockchain технологијата најчесто се користи за:

- а) обработка на фотографии;
- б) сигурно чување и следење на дигитални записи;
- в) изработка на оперативни системи;
- г) поправка на компјутерски мрежи.

2. Точно или неточно.

Криптовалутите претставуваат единствена примена на blockchain технологијата.

3. Објасни ги поимите.

Поим	Објаснување
Блок	_____
Трансакција	_____
Децентрализација	_____

4. Дополни.

Најпознатата примена на blockchain технологијата се _____.

5. Како се применува принципот на децентрализација при извршување на една дигитална трансакција во blockchain мрежата во споредба со традиционалното банкарско плаќање?

6. Анализирај ја улогата на децентрализацијата во безбедноста на податоците. Зошто е речиси невозможно да се избрише или да се промени некоја веќе запишана трансакција откако блокот ќе се поврзе во синџирот?

7. Процени ги предностите и недостатоците на криптовалутите како најмасовна примена на blockchain технологијата. Дали сметаш дека тие можат целосно да ги заменат традиционалните пари во иднина? Образложи го својот став.

8. Освен за криптовалути, предложи идеја како blockchain технологијата (со своите карактеристики за неизменливост и децентрализација) би можела да се примени за решавање на некој реален општествен проблем, како што се гласањето на избори или заштитата на авторските права.

**СИТУАЦИСКА ЗАДАЧА**

Замисли дека треба да се води евиденција за сопственоста на куќите во едно населено место. Објасни на кој начин blockchain технологијата би можела да придонесе за подобра сигурност и доверливост на тие податоци.



1.5

РОБОТИКА И АВТОМАТИЗАЦИЈА

Современите информатички технологии овозможуваат создавање машини што можат самостојно да извршуваат различни задачи. Благодарение на развојот на компјутерските системи, сензорите и автоматското управување, роботите денес се применуваат во голем број области од секојдневниот живот и во индустријата. Тие можат да извршуваат прецизни, повторливи или опасни задачи, со што им помагаат на луѓето и придонесуваат за подобрување на работните процеси.

ВЕЌЕ ЗНАЕШ ДА...

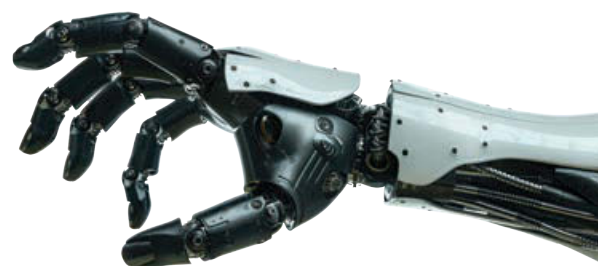
- наведуваш примери на современи технологии;
- објаснуваш како информатичките системи обработуваат информации;
- препознаваш примена на дигиталните технологии во секојдневниот живот;
- наведуваш примери за автоматизација на различни процеси;
- објаснуваш како придонесуваат современите технологии за развојот на општеството.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Дали можат роботите да извршуваат задачи што порано ги извршувале луѓето?
2. Во кои професии си видел употреба на работи?
3. Како би можел робот да им помогне на луѓето во секојдневниот живот?
4. Дали роботите секогаш имаат човечки изглед?
5. Кои задачи би му ги доверил на еден робот?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- го објаснуваш поимот роботика.
- наведуваш примери за примена на работи.
- ја разбираш улогата на автоматизацијата.
- го поврзуваш користењето на роботите со современи технологии.



Роботиката претставува современа научна и техничка област што се занимава со проектирање, изработка и примена на роботски системи. Нејзиниот развој е поврзан со информатиката, електрониката, машинството и автоматиката. Денес роботите може да се сретнат во фабрики, болници, во земјоделството, транспортот, домаќинствата и во многу други области. Благодарение на нивната примена, многу задачи се извршуваат побрзо, попрецизно и побезбедно.



1.5.1 ПОИМ ЗА РОБОТИКА

Роботиката претставува научна и техничка област што се занимава со создавање и примена на работи. Роботот е машина што може да извршува однапред определени задачи со помош на програма и информации што ги добива од својата околина. За својата работа роботите користат различни сензори, управувачки системи и механички делови што им овозможуваат да се движат, да препознаваат предмети и да извршуваат одредени активности. Во зависност од нивната намена, роботите може да имаат различен изглед и различни можности.

Роботите не треба задолжително да имаат човечки изглед. Најголем дел од роботските системи што се користат денес имаат форма што е приспособена на задачите што ги извршуваат. Во индустријата најчесто се користат роботски раце за склопување производи, во домаќинствата има работи за чистење, а во земјоделството – автоматизирани машини за следење и обработка на посевите. Различните видови работи се создаваат со цел да им помагаат на луѓето и да овозможат побрзо и попрецизно извршување на задачите.

1.5.2 АВТОМАТИЗАЦИЈА И ПРИМЕНА НА РОБОТИТЕ

Автоматизацијата претставува процес при кој одредени задачи се извршуваат автоматски, со минимално или без директно учество на човекот. Роботските системи се важен дел од автоматизацијата, затоа што можат да извршуваат голем број повторливи активности со голема брзина и точност. Во производството роботите составуваат производи, вршат заварување, бојадисување и пакување, додека во складиштата помагаат при сортирање и транспорт на стоки.

Примената на роботите не е ограничена само на индустријата. Во медицината се користат роботски системи што им помагаат на лекарите при сложени операции. Во земјоделството роботите можат да ја следат состојбата на посевите и да придонесат за попрецизно наводнување. Во транспортот се развиваат автономни возила и дрoнови, а во домаќинствата се користат работи за чистење и одржување на просторот. Благодарение на развојот на информатиката, роботите стануваат дел од сè поголем број активности во современото општество.

1.5.3 МЕДИЦИНСКИ РОБОТИ, ДРОНОВИ И СОВРЕМЕНИ РОБОТСКИ СИСТЕМИ

Развојот на роботиката овозможува создавање специјализирани работи што им помагаат на луѓето во различни области од животот и работата. Во медицината се користат роботски системи што им овозможуваат на лекарите да извршуваат сложени и прецизни хируршки зафати. Тие помагаат и при рехабилитација на пациенти, транспорт на медицинска опрема и достава на лекови во болниците. Благодарение на нивната голема прецизност, медицинските работи придонесуваат за подобрување на здравствените услуги и за намалување на можноста за грешки при одредени постапки.

Посебна група современи роботски системи претставуваат дрoновите. Тие се беспилотни летала што може да се управуваат од далечина или автоматски да извршуваат однапред зададени задачи. Се применуваат при фотографирање и снимање, набљудување на земјоделски површини, следење на сообраќајот, пребарување и спасување, како и при достава на различни производи. Во некои држави дрoновите се користат и за транспорт на медицински материјали до тешко достапни места. Развојот на ваквите системи покажува дека роботиката има широка примена во современото општество.

1.5.4 РОБОТИ ВО АВТОМОБИЛСКАТА ИНДУСТРИЈА

Во современите фабрики за производство на автомобили голем број работни процеси се извршуваат со помош на индустриски работи. Роботските раце вршат заварување, боење, склопување и

преместување на тешки делови со голема брзина и прецизност. Тие можат непрекинато да работат и да извршуваат исти задачи со еднаков квалитет, што придонесува за зголемување на производството.

Иако роботите извршуваат голем дел од работата, тие не ги заменуваат целосно луѓето. За нивно програмирање, одржување и контрола се потребни стручни лица со различни знаења и вештини. На овој начин роботиката создава нови можности за развој на современи професии поврзани со информатиката, електрониката и инженерството.

1.5.5 ПРЕДНОСТИ И ОГРАНИЧУВАЊА НА РОБОТИКАТА

Роботиката носи голем број придобивки за современото општество. Роботите можат да извршуваат задачи што бараат голема прецизност, брзина или издржливост. Во индустријата помагаат при производство на различни производи, во медицината придонесуваат за подобрување на здравствениите услуги, а во земјоделството овозможуваат поефикасно следење и обработка на посевите. Роботите можат да работат во опасни средини, како што се рудници, подрачја зафатени од природни катастрофи или во вселенски мисии, со што се намалува ризикот за луѓето. Нивната примена придонесува за подобрување на продуктивноста и квалитетот на различни работни процеси.

И покрај бројните предности, роботските системи имаат и одредени ограничувања. Тие работат според програми и правила што ги создава човекот и не можат самостојно да заменат многу човечки способности како што се креативноста, емпатијата и моралното расудување. За нивната работа се потребни редовно одржување, техничка поддршка и стручни лица што ќе ги програмираат и контролираат. Развојот на роботиката отвора и нови прашања за безбедноста, одговорноста и влијанието на автоматизацијата врз различните професии. Поради тоа, роботските системи треба да се развиваат и да се применуваат на начин што ќе придонесува за подобрување на животот и работата на луѓето.

1.5.6 РОБОТИКАТА И ИДНИНАТА

Развојот на роботиката продолжува со голема брзина и се очекува во иднина роботските системи да станат уште позастапени во секојдневниот живот. Во паметните градови се развиваат автономни возила, роботи за одржување на јавните простори и автоматизирани системи за управување со сообраќајот. Во медицината се создаваат роботи што ќе им помагаат на лекарите при дијагностицирање и лекување на пациентите, додека во земјоделството се развиваат системи што ќе ја следат состојбата на посевите и автоматски ќе извршуваат различни активности.

Развојот на роботиката е тесно поврзан со развојот на информатиката и на вештачката интелигенција. Современите роботи користат различни сензори за да собираат информации од околината и компјутерски системи за обработка на тие информации. Благодарение на тоа, тие можат да препознаваат предмети, да избегнуваат пречки и да донесуваат едноставни одлуки при извршувањето на своите задачи. Во иднина се очекува роботите и вештачката интелигенција уште повеќе да соработуваат и да создаваат нови можности за развој на современото општество.



1.5.7 РОБОТИТЕ ВО СПАСУВАЧКИ МИСИИ

По земјотреси, пожари и други природни катастрофи, спасувачките екипи честопати се соочуваат со опасни услови за работа. Во такви ситуации се користат специјални роботи и дронави што можат да пристапат до места каде што движењето на луѓето е тешко или опасно. Тие се опремени со камери и различни сензори што овозможуваат пребарување на теренот и пронаоѓање на лица на кои им е потребна помош.

Примената на роботските системи во спасувачките мисии придонесува за побрзо собирање информации и подобра организација на спасувачките активности. Истовремено, се намалува ризикот за луѓето што учествуваат во овие операции. Овој пример покажува дека роботиката не служи само за автоматизација на производството, туку има важна улога и во заштитата на човечкиот живот.



ДАЛИ ЗНАЕШ?

Дали знаеш дека на дното на океаните и на површината на Марс работат специјални роботи што собираат податоци и извршуваат истражувања на места каде што човекот тешко може да пристапи?



ИСТРАЖИ

Избери една област во која се користат роботи (медицина, индустрија, земјоделство, транспорт, вселена или спасувачки мисии). Подготви краток извештај во кој ќе објасниш:

- какви роботи се користат;
- кои задачи ги извршуваат;
- какви придобивки обезбедуваат;
- со какви предизвици се соочуваат.



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО:

1. Што претставува роботиката?
2. Што претставува автоматизацијата?
3. Дали сите роботи имаат човечки изглед? Образложи го одговорот.
4. Во кои области од секојдневниот живот и во работата се применуваат роботите?
5. Како придонесуваат роботите за подобрување на производството и на услугите?
6. Кои се најважните предности од примена-та на роботските системи?
7. Кои ограничувања и предизвици постојат при нивната примена?



ИСТРАЖУВАЧКИ И ПРАКТИЧНИ ЗАДАЧИ

1. Направи табела во која ќе споредиш четири различни видови роботи (индустриски, медицински, домашни и дронави). За секој наведи:

- област на примена;
- задачи што ги извршува;
- придобивки од неговата употреба.

2. Истражи како се користат роботите во современото земјоделство. Подготви краток извештај и наведи најмалку три активности што можат да ги извршуваат.

3. Замисли дека твоето училиште набавува робот што ќе им помага на учениците и наставниците. Опиши:

- какви задачи би извршувал;
- кои технологии би ги користел;
- како би го подобрил работењето на училиштето.

4. Направи кратка презентација за примена на роботите во медицината или при спасувачките мисии. Вклучи примери и објасни какви придобивки обезбедуваат.

5. Размисли кои професии би можеле да користат роботи во иднина. Наведи најмалку пет примери и објасни на кој начин би им помагале на луѓето.



ЗАКЛУЧОК:

Роботиката претставува научна и техничка област што се занимава со создавање и примена на роботи. Роботите не мора да имаат човечки изглед и може да бидат приспособени за различни задачи. Автоматизацијата овозможува одредени активности да се извршуваат автоматски. Роботите имаат примена во индустријата, во медицината, земјоделството, транспортот, домаќинствата и во многу други области. Роботските системи можат да извршуваат опасни, тешки и повторливи задачи. Развојот на роботиката е тесно поврзан со информатиката и со современите технологии.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во последните години се развива нова генерација роботи што можат да соработуваат со луѓето при извршување различни задачи. Овие системи користат сензори, камери и современи информатички решенија за подобро да ја препознаат својата околина. Покрај индустриските роботи, се развиваат роботи за помош на стари лица, автономни возила, роботи за истражување на океаните и на вселената, како и роботи што учествуваат во научни експерименти. Се очекува роботиката и во иднина да има значајна улога во развојот на современото општество.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

1. Заокружи го точниот одговор.

Роботиката претставува област што се занимава со:

- а) производство на електрична енергија;
- б) создавање и примена на роботски системи;
- в) проучување на планетите;
- г) поправка на компјутери.

2. Точно или неточно.

Сите роботи имаат човечки изглед.

3. Дополни.

Процесот при кој одредени задачи се извршуваат автоматски се нарекува _____.

4. Објасни ги поимите.

Поим	Објаснување
Робот	_____
Роботика	_____
Автоматизација	_____



СИТУАЦИСКА ЗАДАЧА

Замисли дека во едно складиште треба секој ден да се преместуваат тешки товари. Објасни како би можеле да помогнат роботските системи во оваа работа и кои придобивки би ги обезбедиле.



1.6

ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА И МАШИНСКО УЧЕЊЕ

Луѓето секојдневно користат различни дигитални услуги што можат да препознаваат говор, да анализираат фотографии, да предлагаат содржини и да помагаат при донесување одлуки. Мобилните телефони можат да го препознаат лицето на корисникот, интернет-платформите предлагаат филмови и музика според интересите на корисниците, а системите за навигација ја избираат најсоодветната маршрута до одредена локација. Овие можности се резултат на развојот на современите информатички технологии и на примената на вештачката интелигенција.

ВЕЌЕ ЗНАЕШ ДА...

- наведуваш примери на современи технологии;
- објаснуваш како користат роботите информации од околината;
- препознаваш примена на автоматизацијата;
- наведуваш примери за дигитални услуги;
- објаснуваш како информатичките системи обработуваат податоци.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Како мобилниот телефон го препознава лицето на својот сопственик?
2. Како интернет-продавниците предлагаат производи што би можеле да те интересираат?
3. Како апликациите за навигација ја избираат најдобрата маршрута?
4. Дали компјутерските системи можат да учат од податоците што ги обработуваат?
5. Во кои области од секојдневниот живот си сретнал примена на вештачка интелигенција?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што претставува вештачката интелигенција;
- објаснуваш што претставува машинското учење;
- наведуваш примери за примена на вештачката интелигенција;
- анализираш како функционира ВИ;
- препознаваш предности и ризици на даден пример.

Вештачката интелигенција претставува област од информатиката што се занимава со создавање компјутерски системи способни да обработуваат информации и да решаваат задачи што вообичаено бараат човечка интелигенција. За својата работа овие системи користат големи количества податоци и различни алгоритми за анализа и обработка на информациите. Благодарение на тоа, тие можат да препознаваат обрасци, да предвидуваат можни резултати и да им помагаат на луѓето при извршување на различни активности.

1.6.1 ШТО ПРЕТСТАВУВА ВЕШТАЧКАТА ИНТЕЛИГЕНЦИЈА?

Вештачката интелигенција претставува област од информатиката што се занимава со развој на компјутерски системи способни да извршуваат задачи што обично бараат човечко размислување. Таквите системи можат да препознаваат слики, да обработуваат говор, да анализираат податоци и да помагаат при донесување одредени одлуки. За својата работа тие користат информации што ги добиваат од различни извори и применуваат посебни алгоритми за нивна обработка. Основната цел на вештачката интелигенција е да создаде системи што ќе можат поефикасно да решаваат различни практични задачи.

Примената на вештачката интелигенција денес е присутна во многу области од секојдневниот живот. При користење на интернет-пребарувачи, социјални мрежи, платформи за музика и филмови или за мобилни телефони, корисниците секојдневно користат услуги што применуваат алгоритми на вештачка интелигенција. Во медицината овие системи помагаат при анализа на медицински снимки, во сообраќајот придонесуваат за развој на автономни возила, а во образованието овозможуваат создавање интерактивни системи за учење. Благодарение на ваквата примена, вештачката интелигенција станува важен дел од современото општество.

1.6.2 ШТО ПРЕТСТАВУВА МАШИНСКОТО УЧЕЊЕ?

Машинското учење претставува област на вештачката интелигенција што овозможува компјутерските системи да учат од податоците што ги обработуваат. Наместо за секоја ситуација да бидат напишани посебни правила, системот анализира голем број примери и постепено открива одредени закономерности и обрасци. Колку повеќе соодветни податоци обработува системот, толку подобро може да ги извршува задачите за кои е наменет.

Машинското учење има широка примена во современите информатички системи. Се користи при препознавање на говор, автоматски превод на текстови, препознавање лица и предмети на фотографии, откривање несакани електронски пораки и препорачување различни содржини на интернет. На пример, кога корисникот гледа видеа на интернет, системот анализира кои содржини најчесто ги избира и предлага нови видеа со слична тематика. На овој начин машинското учење им помага на компјутерските системи подобро да ги разберат потребите на корисниците и да обезбедат покорисни услуги.

1.6.3 ПРИМЕНА НА ВЕШТАЧКАТА ИНТЕЛИГЕНЦИЈА

Вештачката интелигенција денес има примена во голем број области од современото општество. Во здравството помага при анализа на медицински податоци и снимки, во транспортот придонесува за развој на системи за управување со сообраќајот и автономни возила, а во образованието се користи за создавање интерактивни образовни содржини и системи за учење. Во земјоделството се применува за следење на состојбата на посевите, додека во индустријата помага при автоматизација на производствените процеси и контрола на квалитетот на производите.

Голем број дигитални услуги што секојдневно ги користат луѓето применуваат различни методи на вештачка интелигенција. Интернет-пребарувачите помагаат при пронаоѓање информации, системите за навигација предлагаат соодветни маршрути, а различни платформи препорачуваат музика, филмови или други содржини според интересите на корисниците. При обработката на фотографии и видеозаписи, современите системи можат да препознаваат лица и предмети, додека при користење на мобилни телефони можат да помогнат при препознавање говор и претворање на говорот во текст. Овие примери покажуваат дека вештачката интелигенција веќе претставува дел од секојдневниот живот.

1.6.4 КАКО ФУНКЦИОНИРА ВИ?

Секој пат кога твојот телефон ќе се отклучи со препознавање на гласот или кога YouTube ќе ти го предложи следното омилено видео, ти ја користиш моќта на вештачката интелигенција (ВИ). Иако на прв поглед изгледа како магија, зад овие процеси стојат прецизни математички чекори, анализа на податоци и алгоритми за машинско учење. За да разбереме како навистина функционира ВИ, ќе анализираме два реални примери, кои ги користиме секој ден.

ПРИМЕР 1: Системи за препораки (Како TikTok, Netflix или Spotify знаат што сакаш?)

Дали некогаш си се запрашал/а како успева една апликација да го задржи твоето внимание со часови? Таа користи ВИ систем за препораки. Овој систем не погодува случајно, туку работи низ четири клучни фази:

1. **Собирање податоци (твојот дигитален отпечаток):** Додека ја користиш апликацијата, алгоритмот будно го следи секој твој потег. Тој ги собира следните податоци: на кои објави се задржуваш подолго, кои видеа ги прескокнуваш по првата секунда, што ти се допаѓа, што споделуваш или коментираш.
2. **Препознавање на обрасци (пронаоѓање шеми):** ВИ ги анализира овие податоци и те групира со илјадници други корисници што имаат слични навики. Ако корисникот Марко и ти гледате исти три видеа за готвење, ВИ ве означува како „дигитални близнаци“ со слични интереси.
3. **Филтрирање и предвидување:** Ако Марко прегледа четврто видео (на пример, нов рецепт за пица) и го изгледа докрај, алгоритмот математички пресметува дека има 95% шанса тоа видео да ти се допадне и тебе.
4. **Резултат:** Видеото за пица се појавува на твојата почетна страница. Ако го прегледаш, системот добива потврда дека неговата пресметка била точна и алгоритмот станува уште попаменителен за следната препорака.

ПРИМЕР 2: Препознавање говор (Како функционираат Siri, Google Assistant или диктирањето пораки?)

Кога ќе кажеш „Hey Siri“ или кога користиш опција за претворање на говор во текст (Voice-to-Text), вештачката интелигенција треба да го премости јазот помеѓу човечкиот (биолошки) и компјутерскиот (дигитален) свет. Тој сложен процес се одвива на следниот начин:

1. **Акустична анализа:** Твојот глас е звучен бран. Микрофонот на уредот го снима тој бран и го претвора во дигитален сигнал (низа од бројки). ВИ го прочистува овој сигнал од околните шумови (на пример, бучавата од автомобилите на улица).
2. **Разбивање на фонеме:** ВИ го дели снимениот говор на најмали можни звучни единици наречени фонеме. На пример, ако го кажете зборот „компјутер“, алгоритмот го препознава како спој на звуците к-о-м-п-ј-у-т-е-р.
3. **Јазично моделирање:** Ова е клучниот дел на ВИ. Алгоритмот ги споредува тие звуци со огромна база на податоци (милиони часови снимен човечки говор) за да сфати кој збор најмногу одговара на тој звук. Тој исто така го анализира и контекстот на реченицата. Ако кажете „Сакам јаголко“, ВИ знае дека мислите на овошје, а не на технолошката компанија Apple, благодарение на претходниот збор „сакам“.
4. **Извршување на наредбата:** Откако ВИ ќе го разбере текстот, таа ја активира соодветната функција во телефонот – го пишува зборот на екранот, ја емитува песната што си ја побарал/а или ти одговара на прашањето со генериран глас.

Од овие два примери гледаме дека ВИ не „размислува“ како човек со емоции и свест. Таа функционира како **моќен статистички процесор**. Принципот е секогаш ист: **Влезни податоци** (твоите „лајко-

ви“ или глас) → **Математичка обработка и споредба со претходно знаење** → **Излезен резултат** (нова препорака или напишан текст).

Колку повеќе податоци има ВИ на располагање, толку попрецизно ќе функционира во секојдневните примери.

1.6.5 ПРЕДНОСТИ И РИЗИЦИ НА ВЕШТАЧКАТА ИНТЕЛИГЕНЦИЈА

Вештачката интелигенција (ВИ) без сомнение е една од најважните технолошки иновации во историјата на човештвото. Таа веќе не е само теорија од научната фантастика, туку алатка која активно ги преобликува медицината, образованието, бизнисот и нашето секојдневие. Сепак, како и секоја моќна технологија, ВИ со себе носи огромен потенцијал за напредок, но и сериозни предизвици. За да бидеме подготвени за иднината, мора детално да ги анализираме двете страни на „медалот“: предностите и можните ризици од нејзиното користење.

ПРЕДНОСТИ ОД КОРИСТЕЊЕТО НА ВИ

Имплементацијата на ВИ во општеството носи револуционерни придобивки кои го прават животот полесен, побезбеден и поефикасен.

- **Автоматизација и огромна ефикасност:** ВИ може да обработува милијарди податоци за неколку секунди и да извршува репетитивни (повторливи/досадни) задачи без одмор. Ова им овозможува на луѓето да се ослободат од административните работи и да се фокусираат на креативно размислување и решавање комплексни проблеми.
- **Напредок во медицината и спасување животи:** ВИ алгоритмите можат да ги анализираат медицинските снимки (како рендген или магнетна резонанца) со прецизност поголема од човечкото око, откривајќи болести во најрана фаза. Исто така, ВИ драматично го забрзува процесот на откривање и тестирање на нови лекови.
- **Персонализирано учење:** Во образованието, паметните системи можат да го приспособат материјалот за учење според брзината и стилот на секој ученик поединечно. Ако некој ученик „заглави“ на одредена тема, ВИ асистентот може да понуди алтернативни објаснувања и вежби.
- **Зголемена безбедност во опасни средини:** Роботи управувани од ВИ можат да извршуваат задачи што се смртоносни за луѓето, како што се деминирање терени, истражување на длабоките океани, гаснење големи пожари или работа во нуклеарни центри.

МОЖНИ РИЗИЦИ ОД КОРИСТЕЊЕТО НА ВИ

И покрај сите предности, неконтролираниот развој на вештачката интелигенција отвора сериозни етички, социјални и безбедносни прашања.

- **Влијание врз пазарот на труд (губење работни места):** Брзиот развој на ВИ предизвикува страв од масовна невработеност. Системите базирани на ВИ можат да заменат професии во области како што се: корисничка поддршка, превод, возење (автономни возила), па дури и помлади програмери или писатели, што им наложува на луѓето итна потреба од преквалификација.



- **Пристрасност и дискриминација (algorithmic bias):** ВИ учи од податоците што ги создале луѓето. Ако тие податоци содржат историски предрасуди или дискриминација (на расна, родова или национална основа), ВИ ќе ги научи тие предрасуди и ќе продолжи да донесува неправедни одлуки – на пример, при селекција на кандидати за работа или при одобрување банкарски кредити.
- **Губење на приватноста:** За да функционира прецизно, на ВИ ѝ се потребни огромни количини лични податоци. Сериозен е ризикот од постојан дигитален надзор, каде што секој наш чекор, разговор, пребарување и емоција се следат, се анализираат и потенцијално се злоупотребуваат или за комерцијални или за политички цели.
- **Ширење дезинформации и „Deepfakes“:** Напредните ВИ алатки овозможуваат лесно креирање на лажни вести, уверливи текстови, па дури и лажни видео- и аудиоснимки каде што јавни личности или обични луѓе изгледаат како да кажуваат работи што никогаш не ги изговориле. Ова ја загрозува довербата во информациите и може да предизвика политичка нестабилност.
- **Когнитивна мрзливост кај луѓето:** Претераното потпирање на ВИ за пишување домашни задачи, решавање проблеми и донесување одлуки може на долг рок да ги намали критичкото размислување, логиката и креативните способности кај младите генерации.

И покрај многубројните можности, во работата со вештачката интелигенција има и одредени ризици. Нејзината работа зависи од квалитетот и од количината на податоците што ги обработува, како и од правилата според кои е создадена. Современите системи можат да направат грешки или да дадат неточни резултати, па затоа е потребна проверка и надзор од страна на луѓето. Одговорната примена на вештачката интелигенција подразбира нејзино користење како помошна технологија што придонесува за подобрување на различни активности и услуги.

Вештачката интелигенција треба да се користи креативно, етички и одговорно. Клучот за нејзината успешна примена лежи во донесувањето на **строги етички закони и регулативи** кои ќе ги заштитат човековите права и приватноста, додека истовремено ќе го поттикнуваат технолошкиот напредок. Целта на човештвото не треба да биде заменување на човекот со ВИ, туку создавање синергија во која ВИ ќе му помага на човекот да постигне повеќе.

1.6.6 ВЕШТАЧКАТА ИНТЕЛИГЕНЦИЈА И СОВРЕМЕНОТО ОПШТЕСТВО

Вештачката интелигенција има сè поголемо значење во развојот на современото општество. Нејзината примена придонесува за подобрување на различни услуги и олеснување на секојдневните активности на луѓето. Во здравството помага при анализа на медицински податоци, во образованието овозможува создавање современи дигитални алатки за учење, а во сообраќајот придонесува за подобро управување со сообраќајните системи. Во индустријата се користи за автоматизација на производството и контрола на квалитетот, додека во земјоделството помага при следење на состојбата на посевите и подобро користење на ресурсите.

Развојот на вештачката интелигенција отвора нови можности за научни истражувања, економски развој и подобрување на квалитетот на животот. Истовремено, потребно е современите технологии да се користат одговорно и внимателно. Луѓето имаат важна улога при развојот, контролата и примената на овие системи, бидејќи тие ги создаваат правилата според кои работат. Затоа, покрај техничките знаења, важно е да се развиваат и критичкото размислување, дигиталната писменост и одговорното користење на информатичките технологии.

Вештачката интелигенција и иднината

Со развојот на информатиката се очекува вештачката интелигенција да наоѓа примена во сè поголем број области. Современите системи ќе помагаат при научни истражувања, развој на нови лекови,

подобрување на образованието, заштита на животната средина и управување со сложени технички системи. Во многу професии вештачката интелигенција ќе биде алатка што ќе им помага на луѓето побрзо да обработуваат информации и полесно да решаваат различни задачи.

И покрај брзиот развој на технологијата, човекот останува најважниот фактор во донесувањето одлуки и примената на современите системи. Вештачката интелигенција претставува средство што може да придонесе за подобрување на различни активности, но не може целосно да ги замени човечкото знаење, креативноста, искуството и одговорноста. Поради тоа, идниот развој на оваа област ќе зависи од успешната соработка меѓу луѓето и современите информатички технологии.

Вештачката интелигенција во образованието

Современите образовни платформи користат различни технологии на вештачка интелигенција за да им помогнат на учениците и на наставниците. Таквите системи можат да предлагаат дополнителни материјали за учење, да помагаат при пронаоѓање информации и да обезбедуваат интерактивни активности за совладување на наставните содржини. На овој начин учениците можат полесно да пристапат до различни извори на знаење и да го организираат процесот на учење.

Вештачката интелигенција не го заменува наставникот, туку претставува дополнителна алатка што може да придонесе за подобрување на наставниот процес. Наставникот има важна улога при изборот на наставните содржини, насочувањето на учениците и проверката на точноста на информациите. Одговорното користење на современите технологии придонесува за развој на дигиталните компетенции и на критичкото размислување кај учениците.



ДАЛИ ЗНАЕШ?

Дали знаеш дека секојдневно користиш системи што применуваат вештачка интелигенција? Интернет-пребарувачите, апликациите за навигација, системите за препознавање говор и многу дигитални услуги користат различни методи за обработка на податоци и даваат помош при решавање задачи.



ИСТРАЖИ

Истражи пет примери за примена на вештачката интелигенција во секојдневниот живот. За секој пример објасни:

- каде се користи;
- каква задача извршува;
- на кој начин им помага на луѓето.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува вештачката интелигенција?
2. Што претставува машинското учење?
3. Во кои области од современото општество се применува вештачката интелигенција?
4. На кој начин помага вештачката интелигенција во секојдневниот живот?
5. Кои се најважните предности од примената на вештачката интелигенција?
6. Кои ограничувања постојат при користењето на системите со вештачка интелигенција?
7. Зошто е важно вештачката интелигенција да се користи одговорно?





ИСТРАЖУВАЧКИ И ПРАКТИЧНИ ЗАДАЧИ

1. Направи листа на десет примери за примена на вештачката интелигенција во секојдневниот живот. За секој пример наведи каде се користи и каква задача извршува.
2. Истражи како се користи вештачката интелигенција во медицината, во образованието или во земјоделството. Подготви краток извештај.
3. Замисли дека твоето училиште сака да воведо систем со вештачка интелигенција. Предложи три активности во кои би можел да помогне и објасни ги придобивките.
4. Направи табела во која ќе споредиш човек и систем со вештачка интелигенција според:
 - брзина на обработка на податоците;
 - учење;
 - креативност;
 - донесување одлуки;
 - потреба од човечки надзор.
5. Прогледај ги предностите од воведувањето на ВИ во медицината и во образованието наспроти потенцијалниот ризик од губење на работни места во иднина. Дали сметаш дека придобивките го оправдуваат развојот на оваа технологија? Образложи го ставот.
6. Имајќи ги предвид ризиците поврзани со приватноста на податоците и појавата на „Deepfake“ (лажни) видео- и аудиоснимки, донеси суд: Кои етички предизвици од користењето на ВИ се најопасни за младите генерации и како влијаат тие врз нивната способност за критичко размислување?
7. Размисли и напиши краток текст на тема:

„Како може вештачката интелигенција да им го подобри секојдневниот живот на луѓето?“



ЗАПОМНИ ШТО НАУЧИ

Вештачката интелигенција претставува област од информатиката што се занимава со создавање интелигентни компјутерски системи.

Машинското учење претставува дел од вештачката интелигенција што овозможува учење од податоците.

Вештачката интелигенција има примена во здравството, во образованието, индустријата, транспортот, земјоделството и во многу други области.

Современите дигитални услуги секојдневно користат различни методи на вештачка интелигенција.

Вештачката интелигенција овозможува многу придобивки, но има и некои ограничувања.

Одговорната примена на вештачката интелигенција има значајна улога во развојот на современото општество.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ



Развојот на вештачката интелигенција претставува една од најдинамичните области во современата информатика. Денес се создаваат системи што можат да обработуваат големи количества податоци, да помагаат при научни истражувања и да придонесуваат за решавање сложени проблеми. Вештачката интелигенција се користи при истражување на вселената, за развој на нови лекови, за заштита на животната средина и во многу други активности. Со понатамошниот развој на технологијата се очекува нејзината примена да продолжи да се зголемува, при што човекот ќе има важна улога во контролата и одговорното користење на овие системи.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

1. Заокружи го точниот одговор.

Вештачката интелигенција претставува:

- а) програма за обработка на текст;
- б) област од информатиката што создава интелигентни компјутерски системи;
- в) компјутерска игра;
- г) оперативен систем.

2. Точно или неточно.

Машинското учење претставува дел од вештачката интелигенција.

3. Дополни.

Областа што им овозможува на компјутерските системи да учат од податоците се нарекува _____.

4. Објасни ги поимите.

Поим	Објаснување
Вештачка интелигенција	_____
Машинско учење	_____
Алгоритам	_____



СИТУАЦИСКА ЗАДАЧА

Замисли дека едно училиште сака да користи систем со вештачка интелигенција за подобрување на наставата. Наведи две активности во кои системот би можел да помогне и објасни зошто е потребен човечки надзор при неговото користење.



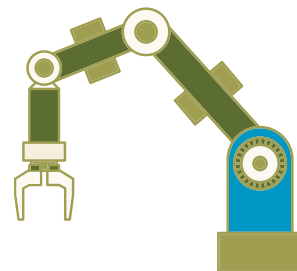
1.7

ГОЛЕМИ ЈАЗИЧНИ МОДЕЛИ И ГЕНЕРАТИВНА ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА

Големите јазични модели, попознати по англиската скратеница LLM (Large Language Models) претставуваат напредни системи на вештачка интелигенција кои се тренирани на огромни количини текстуални податоци со цел да го разберат, да го обработат и да го интерпретираат човечкиот јазик. Нивната примарна и најфасцинантна способност е револуционерното генерирање текст, каде што моделот, врз основа на зададено прашање или контекст, може самостојно да пишува есеи, статии, песни или детални објаснувања кои звучат природно, како да ги напишал човек. Оваа технолошка основа овозможи создавање на паметни chatbot системи (како ChatGPT), кои овозможуваат интерактивна комуникација во реално време и даваат моментални одговори на најразлични кориснички прашања. Благодарение на својата флексибилност, овие модели наоѓаат масовна примена во образованието, каде што служат како персонализирани дигитални тутори кои им помагаат на учениците да совладаат тешки лекции или да преведуваат јазици. Истовремено, тие направија огромна трансформација и во светот на технологијата, овозможувајќи моќна примена во програмирањето, каде што развивачите ги користат за автоматско пишување, прегледување, пронаоѓање грешки (дебагирање) и објаснување на компјутерски код

ВЕЌЕ ЗНАЕШ ДА...

- објаснуваш што претставува вештачката интелигенција;
- објаснуваш што претставува машинското учење;
- наведуваш примери за примена на вештачката интелигенција;
- ги анализираш придобивките и ограничувањата на ВИ;
- препознаваш примена на современите информатички технологии.

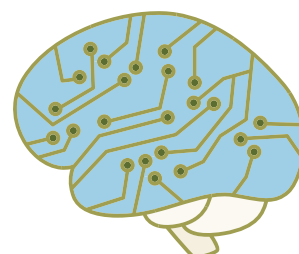


ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Како може компјутерски систем да одговори на прашање поставено на природен јазик?
2. Како е можно компјутер да создаде текст, слика или музика?
3. Дали компјутерскиот систем секогаш дава точни информации?
4. Како може генеративната вештачка интелигенција да помогне во образованието?
5. Зошто е важно критички да се проверуваат информациите добиени од современите дигитални системи?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што претставуваат големите јазични модели;
- наведуваш примери за нивна примена;
- разбираш како се користат за генерирање текст;
- идентификуваш можности за примена во учење и работа.



Во последните години вештачката интелигенција значително се развива и добива нови можности за обработка и создавање различни видови содржини. Современите компјутерски системи можат да разбираат прашања поставени на природен јазик, да создаваат текстови, да помагаат при преведување, да изработуваат резимеа и да генерираат различни идеи. Благодарение на овие способности, тие наоѓаат примена во образованието, во науката, бизнисот и во многу други области.

Големите јазични модели претставуваат современи системи на вештачка интелигенција што се обучуваат со обработка на големи количества текстуални податоци. Благодарение на тоа, тие можат да препознаваат јазични обрасци и да создаваат нови содржини според барањата на корисниците. Генеративната вештачка интелигенција не само што обработува информации туку може да создава текстови, слики, музика и други дигитални содржини. Поради широката примена на овие технологии, важно е корисниците да ги разберат нивните можности, но и ограничувањата.

1.7.1 ШТО ПРЕТСТАВУВААТ ГОЛЕМИТЕ ЈАЗИЧНИ МОДЕЛИ?

Големите јазични модели претставуваат современи компјутерски системи што можат да обработуваат и да создаваат текст на природен јазик. За својата работа тие користат големи количества податоци и различни алгоритми на вештачка интелигенција со чија помош ги препознаваат врските меѓу зборовите и речениците. Благодарение на тоа, можат да одговараат на прашања, да објаснуваат поими, да преведуваат текстови и да помагаат при создавање различни содржини.

Големите јазични модели (LLM), се супернапредни системи на вештачка интелигенција кои се дизајнирани да го разбираат и да го генерираат човечкиот јазик. Зборот „големи“ не е случаен. Тој означува две работи:

1. **Огромна база на податоци.** Овие модели прочитале милијарди страници текст од интернетот – книги, статии, Википедија, научни трудови и компјутерски код.
2. **Милијарди параметри.** Тие имаат сложена внатрешна структура (дигитална невронска мрежа) која им овозможува да ги поврзат зборовите и да го сфатат контекстот на реченицата речиси еднакво добро како и луѓето.

Наједноставно кажано, LLM е како суперпааметен дигитален читател кој ги прочитал речиси сите книги на светот и сега може да разговара со тебе на која било тема.

Големите јазични модели имаат примена во образованието, во науката, бизнисот и за различни дигитални услуги. Тие можат да помогнат при пребарување информации, подготовка на текстови, учење странски јазици и организирање на различни активности. Сепак, важно е да се знае дека овие системи не размислуваат на ист начин како луѓето, туку создаваат содржини врз основа на обработените податоци и алгоритмите според кои работат.

Денес постојат неколку водечки технолошки гиганти кои ги развиваат најмоќните LLM алатки во светот. Сигурно веќе користиш некои од нив:

- **ChatGPT (OpenAI):** Првата алатка која направи револуција во јавноста. Одлична е за секојдневен разговор, креативно пишување и објаснување на комплицирани концепти.
- **Gemini (Google):** Модел кој е директно поврзан со пребарувачот на Google, што му овозможува брз пристап до најсвежите и најточните информации во реално време на интернет.
- **Claude (Anthropic):** Алатка позната по својот исклучително природен стил на пишување, логичко размислување и способност за обработка на огромни, долги документи.
- **Copilot (Microsoft):** Асистент кој е вграден во програмите што ги користиш за училиште (Word, PowerPoint) и помага при пишување и дизајнирање во нив.



1.7.2 КАКО ФУНКЦИОНИРА ГЕНЕРИРАЊЕТО ТЕКСТ?

Многумина мислат дека LLM моделите „размислуваат“ или дека ја знаат вистината. Реалноста е поинаква – тие се **моќни статистички машини за предвидување**.

Кога ќе му поставиш прашање на моделот (тоа прашање се нарекува инструкција (prompt)), тој не ја пребарува базата за да копира и да „залепи“ готов одговор. Наместо тоа, тој го прави следново:

- Го анализира твоето прашање збор по збор.
- Врз основа на сè што прочитал во минатото, пресметува кој збор логички треба да биде следен.
- На пример, ако реченицата започнува со: „Главниот град на Македонија е...“, моделот математички предвидува дека со веројатност од 99.9% следниот збор е „Скопје“.

Процесот се одвива со неверојатна брзина, збор по збор, создавајќи текст кој изгледа како да го пишува вистински човек во реално време.

LLM моделите можеш да ги користиш како твој личен асистент што е достапен 24/7.

Во учењето како твој личен супертутор:

- Објаснување сложени лекции. Ако не ја разбираш лекцијата по физика или по хемија, можеш да му кажеш на моделот: „Објасни ми го Вториот Њутнов закон како да имам 10 години и дај ми пример од секојдневието“.
- Учење странски јазици. Можеш да водиш разговор со ChatGPT на англиски или на германски јазик, а во одговорот тој да ги поправа твоите граматички грешки.
- Подготовка за тестови. Можеш да му ги прикачиш твоите белешки и да му кажеш: „Направи ми квиз со 5 прашања со повеќекратен избор за да се тестирам“.

Во работата и во програмирањето:

- Пишување и дебагирање код. Ако учиш кодирање во училиште (на пример, Python или C++), LLM може веднаш да ти каже каде ти недостасува точка-запирка (;) или зошто кодот не работи.
- Генерирање идеи. Ако треба да изработиш проект или есеј, а немаш идеја како да почнеш, алатката може да ти понуди пет различни структури и теми за пишување.
- Автоматизација на досадни задачи. Во бизнисот, овие модели се користат за брзо одговарање на е-пошти, пишување извештаи и преведување големи договори.

Важно правило: LLM алатките се одлични соработници, но лоши замени за твоите когнитивни способности. Тие понекогаш можат да „халуцинираат“ (да измислат неточни факти со голема самодоверба). Затоа, секогаш проверувај ги важните информации и користи ги овие алатки за да си го подобриш знаењето, а не за да ја прескокнеш мисловната работа!

Големите јазични модели и генеративната вештачка интелигенција имаат примена во различни области од современото општество. Во бизнисот се користат за подготовка на документи и анализа на информации, додека во науката можат да помогнат при обработка на големи количества податоци и организирање на истражувања. Благодарение на нивната примена, многу активности можат да бидат извршени побрзо и поефикасно.

Генеративната вештачка интелигенција може да создава различни видови дигитални содржини. Освен текстови, современите системи можат да помогнат при создавање слики, музика, презентации и други мултимедијални материјали. Овие технологии наоѓаат примена во образованието, во дизајнот, уметноста, медиумите и во многу други области. Сепак, создадените содржини треба внимателно да се анализираат и да се проверуваат точноста и соодветноста за конкретната намена.

1.7.3 ОДГОВОРНО КОРИСТЕЊЕ НА ГЕНЕРАТИВНАТА ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА

Современите системи на генеративна вештачка интелигенција претставуваат корисни алатки што можат да им помогнат на луѓето при различни активности. Сепак, нивното користење бара одговорност и критичко размислување. Информациите што ги создаваат овие системи не треба автоматски да се прифаќаат како целосно точни, туку треба да се споредуваат со сигурни извори и внимателно да се проверуваат. Особено е важно ова правило да се применува при користење информации за образование, здравство и други значајни области.

При користење на генеративната вештачка интелигенција потребно е да се почитуваат етичките принципи и правилата за безбедно користење на дигиталните технологии. Треба да се внимава на заштитата на личните податоци и приватноста, како и на почитувањето на авторските права и интелектуалната сопственост. Генеративната вештачка интелигенција треба да се користи како помошна алатка што придонесува за учење, истражување и создавање нови идеи, а не како замена за човечкото знаење, креативност и одговорност.

1.7.4 СОВРЕМЕНА ТЕХНОЛОГИЈА ВО ПРАКТИКАТА

Генеративната вештачка интелигенција во образованието

Ученик/чка подготвува презентација за климатските промени. Со помош на генеративна вештачка интелигенција може да добие идеи за структурата на презентацијата, објаснување на поимите и предлози за дополнителни теми за истражување. Потоа ги проверува информациите во учебници, енциклопедии и други сигурни извори и ги дополнува со сопствени заклучоци.

На овој начин генеративната вештачка интелигенција претставува алатка што помага во процесот на учење, но не го заменува самостојното размислување и истражување. Одговорното користење на ваквите системи придонесува за развој на дигиталните компетенции, за критичкото размислување и способноста за анализа на различни извори на информации.



ДАЛИ ЗНАЕШ?

Дали знаеш дека генеративната вештачка интелигенција може да создава различни видови содржини, како што се текстови, слики, музика и видеозаписи, но човекот има важна улога во проверката и одговорното користење на овие содржини?



ИСТРАЖИ

Истражи три области во кои се применува генеративната вештачка интелигенција. За секоја област наведи:

- какви содржини може да создава;
- како им помага на луѓето;
- зошто е важно да се користи одговорно.





ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставуваат големите јазични модели?
2. Што претставува генеративната вештачка интелигенција?
3. Какви видови содржини може да создава генеративната вештачка интелигенција?
4. Во кои области се применуваат големите јазични модели?
5. Зошто е потребно критички да се проверуваат информациите добиени од генеративната вештачка интелигенција?
6. Зошто е важно да се почитуваат авторските права и приватноста при користење на овие технологии?
7. Како може да помогне генеративната вештачка интелигенција во образованието?



ИСТРАЖУВАЧКИ И ПРАКТИЧНИ ЗАДАЧИ

1. Истражи три различни примени на генеративната вештачка интелигенција во секојдневниот живот и подготви краток извештај.
2. Замисли дека треба да подготвиш презентација за заштита на животната средина. Опиши на кој начин може да ти помогне генеративната вештачка интелигенција при подготовката и кои информации задолжително треба самостојно да ги провериш.
3. Направи табела во која ќе споредиш класичен интернет-пребарувач и генеративна вештачка интелигенција според:
 - начинот на работа;
 - видот на информациите;
 - примената;
 - потребата за проверка на информациите.
4. Размисли и наведи пет правила за безбедно и одговорно користење на генеративната вештачка интелигенција.
5. Напиши краток есеј на тема:
„Како може да помогне генеративната вештачка интелигенција во учењето и во образованието?“



ЗАПОМНИ ШТО НАУЧИ

Големите јазични модели претставуваат современи системи на вештачка интелигенција што обработуваат и создаваат текст. Генеративната вештачка интелигенција може да создава различни видови дигитални содржини. Овие технологии имаат примена во образованието, во науката, уметноста, бизнисот и во многу други области. Создадените содржини треба критички да се анализираат и да се проверуваат. При користење на генеративната вештачка интелигенција треба да се почитуваат авторските права и приватноста. Генеративната вештачка интелигенција претставува помошна алатка што го дополнува, а не ги заменува човечкото знаење и одговорноста.





ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Развојот на генеративната вештачка интелигенција продолжува со голема брзина и овозможува создавање сè посложени дигитални содржини. Освен текстови, современите системи можат да создаваат слики, музика, видеозаписи и компјутерски програми. Во иднина се очекува нивната примена да биде уште поширока во образованието, во науката, медицината и во различни стопански дејности. Истовремено, се развиваат правила и препораки за нивно безбедно, етичко и одговорно користење, со цел технологијата да придонесе за развојот на општеството и подобрување на квалитетот на животот.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

1. Заокружи го точниот одговор.

Големите јазични модели се користат за:

- а) производство на компјутери;
- б) обработка и создавање текстуални содржини;
- в) производство на електрична енергија;
- г) поправка на мобилни телефони.

2. Точно или неточно.

Генеративната вештачка интелигенција може да создава текстови, слики и други дигитални содржини.

3. Дополни ја реченицата.

Современите системи што можат да создаваат нови дигитални содржини се нарекуваат _____.

4. Објасни ги поимите.

Поим	Објаснување
Голем јазичен модел	_____
Генеративна ВИ	_____
Одговорно користење	_____



СИТУАЦИСКА ЗАДАЧА

Ученик користи генеративна вештачка интелигенција за подготовка на училиштен проект. Објасни кои чекори треба да ги преземе за да обезбеди точност на информациите и одговорно користење на технологијата.

ПРОГРАМИРАЊЕ ВО C++

Програмирањето претставува процес на создавање компјутерски програми што овозможуваат решавање различни проблеми и автоматизирање на задачи. Со помош на програмските јазици се задаваат инструкции според кои компјутерот обработува податоци, извршува пресметки, донесува одлуки и реализира различни активности. Денес програмирањето има широка примена во многу области од современото општество, како што се: образованието, науката, индустријата, медицината, комуникациите и забавата, при што претставува една од основните вештини на дигиталното време.

Во оваа тематска целина се обработуваат основните принципи на програмирањето преку програмскиот јазик C++. Посебно внимание се посветува на начините на претставување на податоците во компјутерот, изработката на алгоритми и програми, примената на функции и користењето на еднодимензионални низи при решавање програмски задачи. Преку практични примери и активности се развива алгоритамското размислување и се стекнуваат основни знаења и вештини за анализа и изработка на компјутерски програми.

НОВИ ПОИМИ

- бинарен броен систем;
- декаден броен систем;
- октален броен систем;
- хексадекаден броен систем;
- функција;
- параметар;
- библиотека;
- еднодимензионална низа;
- индекс на елемент.

ВЕЌЕ ЗНАЕШ ДА...

- користиш компјутер и различни апликации;
- внесуваш и да обработуваш податоци;
- применуваш логичко размислување при решавање проблеми;
- разликуваш различни видови информации;
- следиш и да извршуваш зададени инструкции;
- анализираш едноставни секојдневни ситуации;
- користиш основни информатички технологии.



TEMA 2





2.1

ДИГИТАЛНО ПРЕТСТАВУВАЊЕ НА ПОДАТОЦИ ВО КОМПЈУТЕР

ВЕКЕ ЗНАЕШ ДА...

- објасниш што претставува компјутерски систем;
- наведеш примери за дигитални уреди;
- разликуваш хардвер и софтвер;
- користиш основни типови податоци во C++;
- изработиш едноставни програми со променливи и оператори.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш како компјутерот ги претставува податоците;
- разликуваш бит и бајт;
- објасниш како се претставуваат броеви, текст, слики и звук;
- препознаеш различни типови податоци;
- анализираш како се користи меморијата за чување информации.

Во современиот дигитален свет, голем број информации постојано се обработуваат, складираат и пренесуваат преку различни компјутерски системи. Текстовите што се пишуваат, фотографиите што се снимаат, музиката што се слуша и видеата што се прикажуваат не се зачувуваат во компјутерот во нивната природна форма, туку претходно се претвораат во дигитални податоци. Поради тоа, потребно е да се разбере начинот на кој компјутерот ги претставува информациите и како се чуваат во меморијата.

Секој податок во компјутерот се претставува со комбинации од бинарни вредности. Иако корисниците гледаат букви, броеви, слики и звук, во заднината сите информации се претставуваат со нули и единици. Таквиот начин на претставување овозможува податоците лесно да се обработуваат од електронските компоненти на компјутерот. Разбирањето на дигиталното претставување на податоците претставува основа за понатамошно изучување на програмирањето и информатиката.

Компјутерските системи работат со електронски сигнали што може да имаат две состојби. Поради тоа, во компјутерите се користи бинарен начин на претставување на податоците. Основната единица на информација се нарекува бит. Битот може да има само две вредности – 0 или 1. Иако еден бит самостојно носи многу мала количина информации, со комбинирање на повеќе битови може да се претстават сложени податоци. Осум бита формираат бајт, кој претставува основна мерка за големина на податоците. Во секојдневната употреба често се користат и поголеми мерни единици како: килобајт, мегабајт, гигабајт и терабајт. Со развојот на технологијата, количината на податоци што се обработуваат постојано се зголемува, па современите уреди располагаат со мемории во кои може да се складираат огромни количини информации. Податоците што се чуваат во меморијата мора да бидат организирани на начин

што ќе овозможи нивно брзо читање и обработка. Затоа, дигиталното претставување е основен принцип врз кој функционира целата современа информатичка технологија.

Пример 1: Претворање на мерни единици за податоци

```

main.cpp x
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int megabajti;
6
7      cout << "Vnesi broj na megabajti: ";
8      cin >> megabajti;
9
10     int kilobajti = megabajti * 1024;
11
12     cout << "Vo kilobajti ima: " << kilobajti << endl;
13
14     return 0;
15 }
16

```

Покрај броевите, во компјутерите се претставуваат и текстуални податоци. Секоја буква, цифра или симбол добива одреден бинарен код преку специјални стандарди за кодирање. Еден од најпознатите стандарди е ASCII кодот, а денес широко се користи и Unicode стандардот кој овозможува претставување на голем број светски јазици и симболи. Благодарение на ваквите стандарди, текстот може правилно да се прикажува на различни уреди и оперативни системи. Освен текстот, дигитално се претставуваат и сликите. Секоја дигитална слика е составена од голем број мали точки наречени пиксели. Секој пиксел содржи информација за бојата, а со комбинирање на илјадници или милиони пиксели се добива целосна слика. Колку е поголем бројот на пиксели, толку е подобар квалитетот на сликата. На сличен начин се претставува и звукот, при што звучните бранови се претвораат во дигитални вредности што компјутерот може да ги обработува. На тој начин, различни типови информации може да бидат зачувани, копирани и обработувани во дигитална форма.

Пример 2: Приказ на ASCII вредност на знак

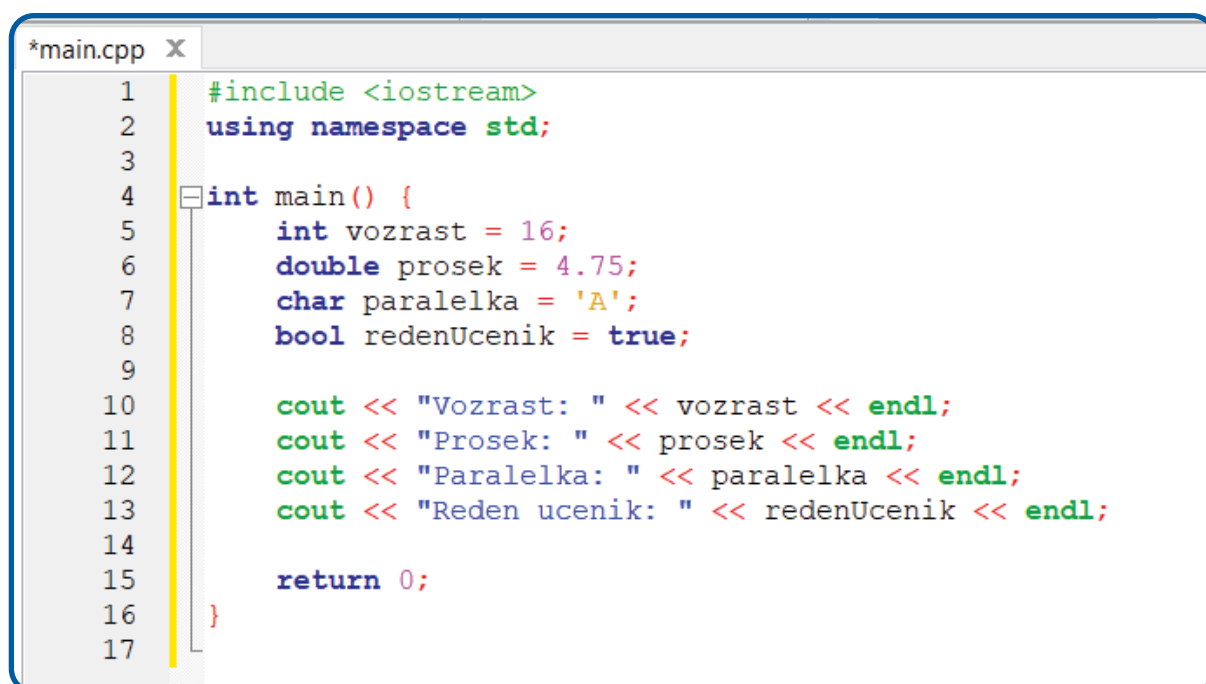
```

main.cpp x
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      char znak;
6
7      cout << "Vnesi znak: ";
8      cin >> znak;
9
10     cout << "ASCII vrednosta e: " << (int)znak << endl;
11
12     return 0;
13 }
14

```

Во програмските јазици, податоците се организираат преку различни типови податоци. Секој тип податок зафаќа определен простор во меморијата и служи за чување различен вид информации. На пример, типот `int` се користи за цели броеви, `double` за реални броеви, `char` за знаци, а `bool` за логички вредности. Изборот на соодветен тип податок е многу важен, затоа што влијае врз точноста на податоците и количината меморија што ќе се користи. Ако за едноставна вредност се избере тип што зафаќа многу меморија, програмата може непотребно да троши ресурси. Од друга страна, ако се избере тип со мал опсег, може да дојде до грешки при пресметките. Поради тоа, програмерите внимателно ги избираат типовите податоци според потребите на програмата. Разбирањето на начинот на кој се организираат податоците во меморијата претставува важен чекор кон создавање поефикасни програми.

Пример 3: Користење различни типови податоци



```
*main.cpp X
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int vozrast = 16;
6      double prosek = 4.75;
7      char paralelka = 'A';
8      bool redenUcenik = true;
9
10     cout << "Vozrast: " << vozrast << endl;
11     cout << "Prosek: " << prosek << endl;
12     cout << "Paralelka: " << paralelka << endl;
13     cout << "Reden ucenik: " << redenUcenik << endl;
14
15     return 0;
16 }
17
```

Меморијата на компјутерот претставува простор во кој се складираат програмите и податоците. Секоја променлива што се користи во програма добива свое место во меморијата. Компјутерот автоматски доделува адреса за секој податок, а програмата пристапува до тие податоци преку имињата на променливите. Начинот на кој се организира меморијата има големо значење за брзината и ефикасноста на програмите. Современите компјутери располагаат со различни типови меморија, како RAM меморија и трајна меморија за складирање податоци. RAM меморијата служи за привремено чување информации додека работи програмата, а трајната меморија ги зачувува податоците и по исклучување на компјутерот. Количината меморија што ја користи програмата зависи од бројот и од типот на податоците што се обработуваат. Затоа, разбирањето на дигиталното претставување и организацијата на податоците претставува важна основа за понатамошно изучување на алгоритми, функции и структури на податоци.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува бит?
2. Колку бита содржи еден бајт?
3. Како се претставуваат текстуалните податоци во компјутер?
4. Која е улогата на типот `double` во C++?



ЗАДАЧИ

1. Напиши програма што ќе пресметува колку бајти имаат зададени килобајти.
2. Напиши програма што ќе прикажува ASCII вредност за внесен знак.
3. Напиши програма што ќе чува и прикажува податоци за ученик/чка со користење различни типови податоци.
4. Напиши програма што ќе пресметува колку мегабајти имаат зададени гигабајти.
5. Напиши програма што ќе прикажува дали внесен знак е голема или мала буква.



ЗАПОМНИ ШТО НАУЧИ

- Компјутерите ги претставуваат податоците во бинарна форма.
- Основна единица за информација е битот.
- Осум бита формираат еден бајт.
- Текстот, сликите и звукот се претставуваат со бинарни вредности.
- Различните типови податоци зафаќаат различен простор во меморијата.
- Меморијата овозможува чување и обработка на податоците.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите компјутерски системи се користат специјални техники за компресија на податоците. Со компресијата се намалува големината на датотеките без значително губење на квалитетот. Благодарение на тоа, фотографиите, музиката и видеата може полесно да се пренесуваат преку интернет и да зафаќаат помал простор во меморијата. Форматите како JPG, MP3 и MP4 користат различни алгоритми за компресија со цел да се намали количината податоци што треба да се складираат.



2.2

БРОЈНИ СИСТЕМИ И КОНВЕРЗИИ

ВЕЌЕ ЗНАЕШ ДА...

- објасниш како компјутерот ги претставува податоците;
- разликуваш бит и бајт;
- наведеш примери за типови податоци;
- објасниш како се претставуваат текст, слики и звук;
- користиш основни типови податоци во C++.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Колку бита имаат 4 бајти?
2. Кој тип податок во C++ се користи за чување знак?
3. Што претставува пиксел?
4. Напиши програма што ќе пресметува колку килобајти имаат зададени мегабајти.
5. Објасни зошто податоците во компјутерот се претставуваат дигитално.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- разликуваш бинарен, декаден, октален и хексадекаден броен систем;
- објасниш што претставува основа на броен систем;
- вршиш рачни конверзии помеѓу бројни системи;
- анализираш позициони вредности на цифрите;
- применуваеш конверзии во програмски задачи.

Во секојдневниот живот најчесто се користи декадниот броен систем, кој е составен од десет цифри – од 0 до 9. Сепак, компјутерите не функционираат на ист начин. Бидејќи електронските кола можат сигурно да препознаат само две состојби, во компјутерските системи се користи бинарниот броен систем. Поради тоа, сите информации што се обработуваат во компјутерот се претставуваат преку комбинации од нули и единици.

Покрај бинарниот систем, во информатиката често се користат и окталниот и хексадекадниот броен систем. Овие системи овозможуваат полесно претставување на долги бинарни записи и подобра организација на податоците. Разбирањето на бројните системи претставува важен дел од информатиката, бидејќи преку нив се разбира начинот на кој компјутерот ги обработува и ги чува информациите.

Бројниот систем претставува начин на претставување броеви со користење определен број симболи. Бројот на симболи што се користат се нарекува основа на броен систем. Во декадниот броен систем основата е 10, бидејќи се користат десет цифри. Во бинарниот систем основата е 2, па се користат само цифрите 0 и 1. Секоја цифра во бројот има позициона вредност која зависи од нејзината позиција и од основата на системот. Кај декадниот систем, позициите претставуваат степени на бројот 10, додека кај бинарниот систем позициите претставуваат степени на бројот 2. Ако бинарниот број има 4 цифри, се користат степените на бројот 2 од 0 до 3. Ако бинарниот број има 15 цифри, ќе се користат степените на бројот два од 0 до 14. Само вредностите каде што во бинарниот број има 1 се собираат. На пример, бинарниот број 1011_2 се пресметува на следниот начин:

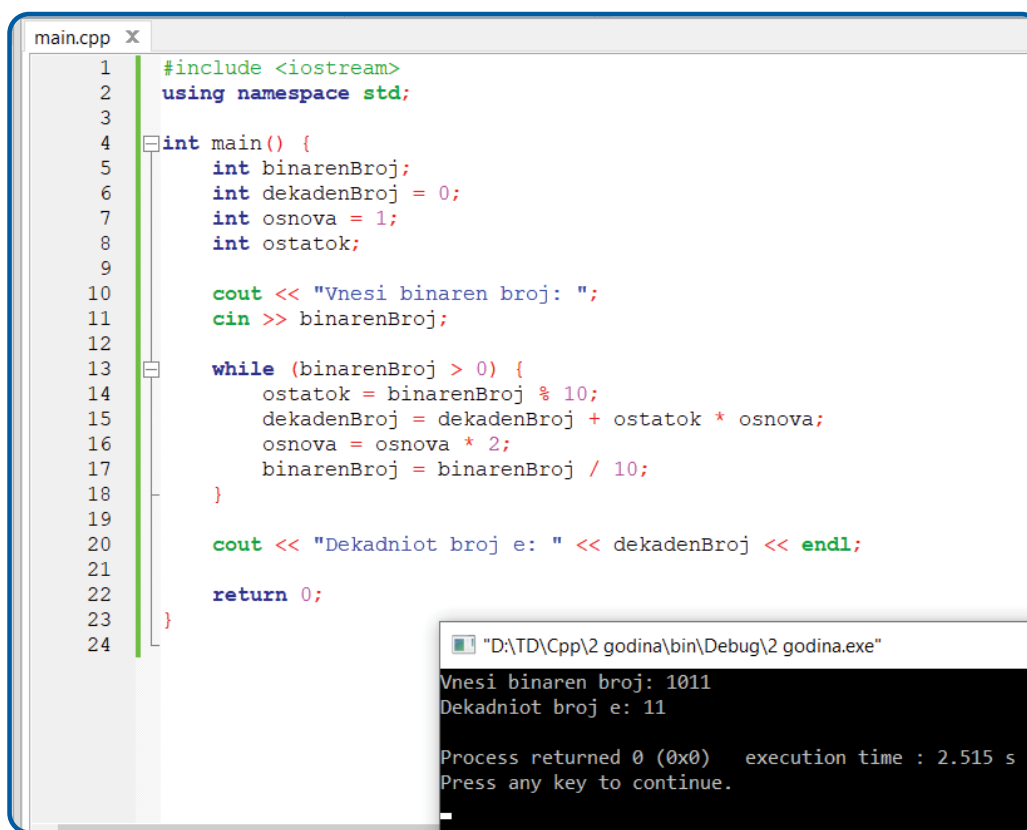
Бинарна цифра	1	0	1	1
Степен на 2	2^3	2^2	2^1	2^0
Вредност	8	0	2	1

Со собирање на вредностите се добива:

$$8 + 0 + 2 + 1 = 11_{10}$$

На тој начин се врши конверзија од бинарен во декаден систем. Бинарниот систем претставува основа на функционирањето на компјутерите, затоа што електронските компоненти лесно можат да претстават две состојби – вклучено и исклучено. Поради тоа, сите податоци и инструкции во компјутерот се претставуваат во бинарна форма.

Пример 1: Претворање бинарен број во декаден



```
main.cpp X
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int binarenBroj;
6      int dekadnenBroj = 0;
7      int osnova = 1;
8      int ostatok;
9
10     cout << "Vnesi binaren broj: ";
11     cin >> binarenBroj;
12
13     while (binarenBroj > 0) {
14         ostatok = binarenBroj % 10;
15         dekadnenBroj = dekadnenBroj + ostatok * osnova;
16         osnova = osnova * 2;
17         binarenBroj = binarenBroj / 10;
18     }
19
20     cout << "Dekadni broj e: " << dekadnenBroj << endl;
21
22     return 0;
23 }
24
```

Output window: "D:\TD\Cpp\2 godina\bin\Debug\2 godina.exe"

```
Vnesi binaren broj: 1011
Dekadni broj e: 11

Process returned 0 (0x0)   execution time : 2.515 s
Press any key to continue.
```

Конверзијата од декаден во бинарен систем се врши со последователно делење на бројот со 2. При секое делење се запишува остатокот, а бинарниот број се добива кога остатоците ќе се прочитаат во обратен редослед. На пример, ако треба да се претвори бројот 13_{10} во бинарен систем, постапката се изведува на следниот начин:

Делење	Количник	Остаток
13 : 2	6	1
6 : 2	3	0
3 : 2	1	1
1 : 2	0	1

Со читање на остатоците оддолу нагоре се добива:

$$13_{10} = 1101_2$$

Оваа постапка претставува една од најважните конверзии во информатиката. Благодарение на ваквите конверзии, програмерите подобро разбираат како компјутерот ги обработува бројките во меморијата. Во програмските јазици често се користат алгоритми што автоматски ги извршуваат овие постапки.

Пример 2: Претворање декаден број во бинарен

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int dekadtenBroj;
6      int binarenBroj[32];
7      int i = 0;
8
9      cout << "Vnesi dekadten broj: ";
10     cin >> dekadtenBroj;
11
12     while (dekadtenBroj > 0) {
13         binarenBroj[i] = dekadtenBroj % 2;
14         dekadtenBroj = dekadtenBroj / 2;
15         i++;
16     }
17
18     cout << "Binarniot broj e: ";
19
20     for (int j = i - 1; j >= 0; j--) {
21         cout << binarenBroj[j];
22     }
23
24     cout << endl;
25
26     return 0;
27 }

```

"D:\TD\Cpp\2 godina\bin\Debug\2 godina
 Vnesi dekadten broj: 13
 Binarniot broj e: 1101
 Process returned 0 (0x0) execution
 Press any key to continue.

Покрај бинарниот систем, во информатиката се користат и окталниот и хексадекадниот броен систем. Окталниот систем има основа 8 и ги користи цифрите од 0 до 7, додека хексадекадниот систем има основа 16 и ги користи цифрите од 0 до 9 и буквите A, B, C, D, E и F. Овие системи овозможуваат долгите бинарни броеви да се претставуваат пократко и попрегледно. При конверзија од бинарен во октален систем, бинарните цифри се групираат по три, а при конверзија во хексадекаден систем се групираат по четири цифри. Групирањето се врши од десно кон лево. Ако останале една или две цифри најлево, се дополнува со нули од лево.

Пример за конверзија во октален систем:

10110110₂

Групирање по 3 цифри:

010 110 110

010₂ = 2₈

110₂ = 6₈

110₂ = 6₈

Значи:

10110110₂ = 266₈

Пример за конверзија во хексадекаден систем:

10110110₂

Групирање по 4 цифри:

1011 0110

1011₂ = B₁₆

0110₂ = 6₁₆

Значи:

10110110₂ = B6₁₆

Хексадекадниот систем има голема примена во програмирањето, мемориските адреси и претставувањето бои во веб-дизајнот. На пример, бојата #FF0000 претставува црвена боја во RGB системот.

Пример 3: Приказ на октален и хексадекаден запис

```
main.cpp x
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int broj;
6
7      cout << "Vnesi dekadnen broj: ";
8      cin >> broj;
9
10     cout << "Oktalen zapis: " << oct << broj << endl;
11
12     cout << "Heksadekaden zapis: " << hex << broj << endl;
13
14     return 0;
15 }
16
```

Различните бројни системи претставуваат важен дел од информатиката и компјутерската архитектура. Иако современите компјутери автоматски ги извршуваат конверзиите, програмерите мора да разбираат како функционираат тие. Таквото знаење овозможува полесно разбирање на организацијата на меморијата, начинот на работа на процесорот и обработката на податоците. Освен тоа, бројните системи се користат и во компјутерски мрежи, криптографија, дигитална електроника и развој на софтвер. Поради тоа, нивното изучување претставува основа за понатамошно изучување на програмирањето и современите информатички технологии.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува основа на броен систем?
2. Кои цифри се користат во бинарниот систем?
3. Како се врши конверзија од бинарен во декаден систем?
4. Колкава е основата на хексадекадниот систем?
5. Зошто во информатиката се користи хексадекаден запис?



ЗАДАЧИ

1. Рачно претвори го бројот 10101_2 во декаден систем.
2. Рачно претвори го бројот 25_{10} во бинарен систем.
3. Рачно претвори го бројот 11101010_2 во октален и во хексадекаден систем.
4. Напиши програма што ќе претвора бинарен број во декаден.
5. Напиши програма што ќе прикажува октален и хексадекаден запис на внесен број.



ЗАПОМНИ ШТО НАУЧИ

- Бројниот систем претставува начин на запишување броеви.
- Бинарниот систем има основа 2.
- Окталниот систем има основа 8.
- Хексадекадниот систем има основа 16.
- Конверзијата од бинарен во декаден систем се врши преку позициони вредности.
- Конверзијата од декаден во бинарен систем се врши со последователно делење со 2.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во компјутерските процесори и мемориски системи често се користат хексадекадни вредности затоа што овозможуваат пократко и попрегледно претставување на бинарните информации. Во програмските јазици, мемориските адреси често се прикажуваат во хексадекаден облик. Исто така, хексадекадниот систем се користи и при претставување бои во веб-технологиите, каде што секоја боја се претставува со комбинација од шест хексадекадни знаци.



Во програмските јазици постојат различни типови податоци за претставување цели броеви. Секој тип зафаќа различен простор во меморијата и овозможува претставување на различен опсег вредности. Ако се обиде програмата да зачува број што е поголем од дозволеният опсег, може да се појави грешка наречена *overflow*. Поради тоа, програмерите мора внимателно да избираат какви типови податоци ќе користат во програмите.

Компјутерите ги претставуваат сите броеви во бинарна форма. Бидејќи е меморијата ограничена, за секој број се резервира определен број битови. На пример, ако за еден број се користат 8 бита, тогаш може да се претстават ограничен број различни вредности. Кај позитивните броеви, секој бит учествува во формирањето на бројот преку степени на бројот 2. На пример, бројот 13 во бинарна форма се претставува како 00001101_2 . Во меморијата секој бит има точно определено место и значење. Колку повеќе битови се користат, толку поголем опсег броеви може да се претстави. Во програмските јазици најчесто се користат типови како: *short*, *int*, *long long* и други, при што секој тип користи различна количина меморија. Поради тоа, изборот на тип податок има директно влијание врз количината меморија што ја користи програмата. Во современите компјутерски системи, правилниот избор на типови податоци претставува важен дел од оптимизацијата на програмите.

Пример 1: Приказ на големина на типови податоци

```

main.cpp x
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      cout << "int: " << sizeof(int) << " bajti" << endl;
7      cout << "short: " << sizeof(short) << " bajti" << endl;
8      cout << "long long: " << sizeof(long long) << " bajti" << endl;
9
10     return 0;
11 }
12

```

```

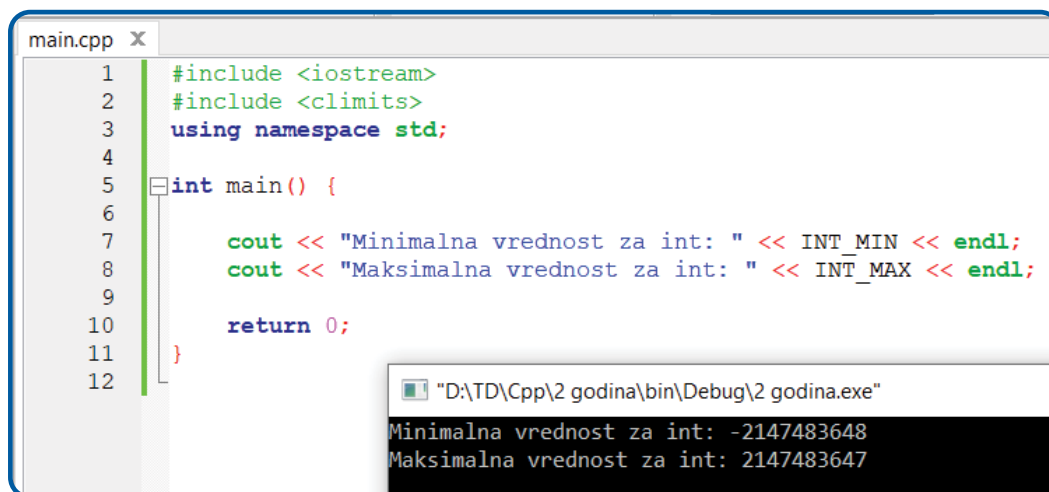
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.exe"
int: 4 bajti
short: 2 bajti
long long: 8 bajti

Process returned 0 (0x0)   execution time : 0.037 s
Press any key to continue.

```

Покрај позитивните броеви, во компјутерите мора да се претставуваат и негативни броеви. За таа цел се користи специјален начин на запишување наречен *two's complement*. Кај овој начин на претставување, најлевиот бит обично го означува знакот на бројот. Благодарение на ваквиот систем, процесорот може едноставно да извршува собирање и одземање и со позитивни и со негативни броеви. На пример, ако за претставување на броеви се користат 8 бита, тогаш може да се претстават вредности од -128 до 127 . Тоа значи дека постои ограничување на најголемиот и на најмалиот број што може да се зачува во меморијата. Ако се обиде програмата да зачува број надвор од дозволеният опсег, резултатот може да биде неточен. Ваквите ограничувања често не се забележуваат кај мали програми, но кај големи системи можат да предизвикаат сериозни проблеми. Поради тоа, разбирањето на опсегот на податоци претставува важен дел од програмирањето.

Пример 2: Приказ на минимална и максимална вредност



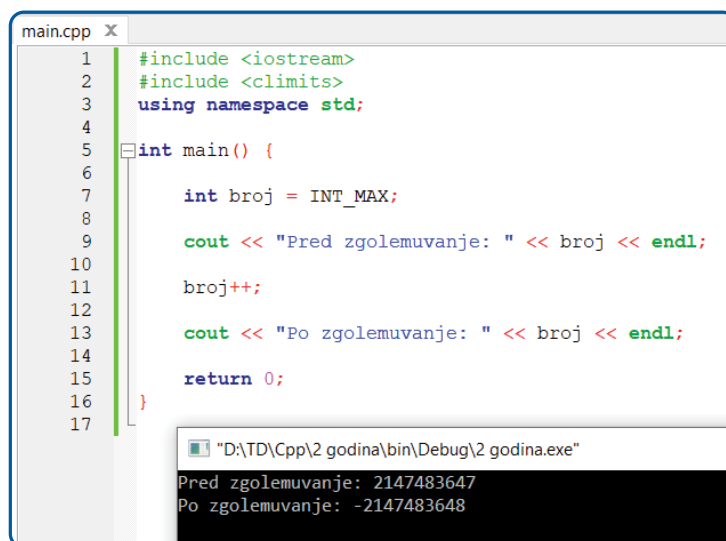
```
main.cpp x
1  #include <iostream>
2  #include <climits>
3  using namespace std;
4
5  int main() {
6
7      cout << "Minimalna vrednost za int: " << INT_MIN << endl;
8      cout << "Maksimalna vrednost za int: " << INT_MAX << endl;
9
10     return 0;
11 }
12
```

"D:\TD\Cpp\2 godina\bin\Debug\2 godina.exe"

Minimalna vrednost za int: -2147483648
Maksimalna vrednost za int: 2147483647

Опсегот на податоци претставува интервал на вредности што може да се претстават со определен тип податок. Колку е поголем бројот на битови што се користат, толку е поголем и опсегот. На пример, типот `short` зафаќа помалку меморија од типот `long long`, но може да чува помали броеви. Во програмите често мора внимателно да се процени колкав опсег ќе биде потребен. Ако се користи тип со мал опсег за големи пресметки, може да дојде до `overflow`. `Overflow` претставува ситуација во која резултатот ја надминува максималната вредност што може да се зачува. Во такви случаи, програмата може да прикаже неочекуван или погрешен резултат. Ваквите грешки се особено опасни кај финансиски, научни и безбедносни системи. Поради тоа, програмерите мора да ги разбираат ограничувањата на типовите податоци и правилно да ги избираат според потребите на програмата.

Пример 3: Overflow кај цел број



```
main.cpp x
1  #include <iostream>
2  #include <climits>
3  using namespace std;
4
5  int main() {
6
7      int broj = INT_MAX;
8
9      cout << "Pred zgolemuvanje: " << broj << endl;
10
11     broj++;
12
13     cout << "Po zgolemuvanje: " << broj << endl;
14
15     return 0;
16 }
17
```

"D:\TD\Cpp\2 godina\bin\Debug\2 godina.exe"

Pred zgolemuvanje: 2147483647
Po zgolemuvanje: -2147483648

Современите програмски јазици овозможуваат користење различни типови податоци со цел програмите да бидат поефикасни и попрецизни. Во некои ситуации е потребно да се чуваат многу големи

броеви, додека во други ситуации е важно да се заштеди меморија. Поради тоа, програмерите избираат различни типови податоци според конкретниот проблем. Освен тоа, правилното разбирање на опсегот на броевите овозможува полесно откривање грешки и подобро анализирање на програмите. Познавањето на начинот на кој се претставуваат броевите во меморијата претставува основа за понатамошно изучување на алгоритми, структури на податоци и оптимизација на програми.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Зошто компјутерите користат бинарно претставување на броевите?
2. Што претставува опсег на податоци?
3. Што е overflow?
4. Кој тип податок може да чува поголеми вредности – int или long long?
5. Зошто е важно правилно да се избере тип на податок?



ЗАДАЧИ

1. Напиши програма што ќе ја прикажува големината на типот char, int и double.
2. Напиши програма што ќе ги прикажува минималната и максималната вредност на типот int.
3. Напиши програма што ќе демонстрира overflow кај типот short.
4. Напиши програма што ќе внесува два големи броја и ќе ги собира со користење long long.
5. Истражи колкав опсег има типот unsigned int и спореди го со int.



ЗАПОМНИ ШТО НАУЧИ

- Компјутерите ги претставуваат броевите во бинарна форма.
- Секој тип податок користи различен број бајти.
- Опсегот претставува најмала и најголема вредност што може да се зачува.
- Overflow се појавува кога вредноста го надминува дозволениот опсег.
- Правилниот избор на тип податок е важен за точноста и ефикасноста на програмата.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите програмски јазици постојат и типови податоци со многу голем опсег, кои се користат во научни симулации, криптографија и обработка на големи бази на податоци. Некои програмски библиотеки овозможуваат работа со броеви што содржат стотици или илјадници цифри. Таквите броеви не може директно да се претстават со стандардните типови податоци, па се користат специјални алгоритми и структури за нивна обработка.



2.4

ОПТИМИЗАЦИЈА И ЕФИКАСНОСТ НА АЛГОРИТМИ

ВЕЌЕ ЗНАЕШ ДА...

- објасниш што претставува алгоритам;
- користиш if и switch структури;
- користиш for и while циклуси;
- анализираш временска сложеност;
- решаваш текстуални задачи во C++.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува алгоритам?
2. Напиши пример за for циклус што се повторува 10 пати.
3. Која е разликата помеѓу for и while циклус?
4. Што претставува временска сложеност?
5. Напиши програма што ќе го пресметува квадратот на внесен број.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- анализираш ефикасност на алгоритми;
- споредуваш различни решенија за ист проблем;
- препознаваш непотребни операции;
- оптимизираш алгоритми;
- го анализираш влијанието на големината на податоците врз брзината на програмата.

Во програмирањето многу често има повеќе различни начини за решавање на ист проблем. Иако различните решенија можат да даваат ист резултат, тие не секогаш работат со иста брзина и не користат иста количина меморија. Кај мали програми ваквите разлики често не се забележуваат, но при обработка на големи количини податоци тие можат значително да влијаат врз работата на системот.

Поради тоа, современото програмирање не се фокусира само на точноста на програмата, туку и на нејзината ефикасност. Програмерите постојано анализираат колку операции извршува алгоритмот,

колку меморија користи и дали постои подобар начин за решавање на проблемот. Таквиот процес се нарекува оптимизација на алгоритми и претставува важен дел од развојот на современ софтвер.

Ефикасноста на алгоритмот најчесто се анализира преку бројот на операции што се извршуваат при решавање на проблемот. Колку повеќе операции се потребни, толку повеќе време ќе биде потребно за извршување на програмата. На пример, ако треба да се пресмета збирот на броевите од 1 до n , може да се користи циклус што ќе ги собира сите броеви еден по еден. Таквиот пристап е точен, но при многу големи вредности на n ќе се извршат огромен број операции. Од друга страна, истиот проблем може да се реши со математичка формула која веднаш го дава резултатот. Во таков случај, алгоритмот извршува многу помал број операции и работи побрзо. Поради тоа, програмерите постојано бараат начини за намалување на непотребните пресметки. Ваквото размислување претставува основа на оптимизацијата на алгоритми и е еден од најважните делови на современото програмирање.

Пример 1: Споредба на два алгоритми

```

main.cpp x
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6      int n;
7      cin >> n;
8
9      int zbir1 = 0;
10
11     for (int i = 1; i <= n; i++) {
12         zbir1 += i;
13     }
14
15     int zbir2 = n * (n + 1) / 2;
16
17     cout << "So ciklus: " << zbir1 << endl;
18     cout << "So formula: " << zbir2 << endl;
19
20     return 0;
21 }
22

```

```

"D:\TD\Cpp\2 godina\bin\Debug\2 godin...
500
So ciklus: 125250
So formula: 125250

Process returned 0 (0x0)   execution time : 3.100 s
Press any key to continue.

```

При анализа на алгоритми многу важно е да се разгледа како се менува брзината на програмата кога се зголемува количината податоци. Алгоритам што работи брзо за 10 елементи може да стане многу бавен кога ќе обработува милион елементи. Поради тоа, програмерите не ја анализираат само моменталната брзина, туку и начинот на кој се однесува алгоритмот при големи влезни податоци. Во информатиката често се користат ознаки како: $O(n)$, $O(n^2)$ и $O(\log n)$ за опишување на сложеноста. Алгоритам со $O(n^2)$ сложеност најчесто станува многу побавен при големи податоци отколку алгоритам со $O(n)$. На пример, вгнездени циклуси често предизвикуваат значително зголемување на бројот на операции. Поради тоа, програмерите внимателно анализираат дали може да се избегне непотребно повторување или дали има подобро решение. Во големите компјутерски системи, правилната оптимизација може значително да ја подобри работата на програмата.

Пример 2: Неефикасно и поефикасно пребарување

```
*main.cpp X
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int niza[5] = {2, 4, 6, 8, 10};
7      int broj;
8      bool pronajden = false;
9      cin >> broj;
10     for (int i = 0; i < 5; i++) {
11
12         if (niza[i] == broj) {
13             pronajden = true;
14             break;
15         }
16     }
17     if (pronajden) {
18         cout << "Brojot e pronajden." << endl;
19     }
20     else {
21         cout << "Brojot ne e pronajden." << endl;
22     }
23     return 0;
24 }
25
```

Во претходниот пример се користи `break` наредба за прекинување на циклусот веднаш по пронаоѓањето на бројот. Без `break`, циклусот би продолжил да проверува и непотребни елементи. Иако е мала разликата кај низа со пет елементи, кај големи низи тоа може значително да влијае врз брзината на програмата. Ваквите мали подобрувања често имаат големо значење во професионалното програмирање.

Програмерите постојано анализираат како да го намалат бројот на операции и како поефикасно да ја користат меморијата. Таквиот процес не значи само „побрза програма“, туку и подобра организација на кодот и полесно одржување на системот.

Пример 3: Влијание на вгнездени циклуси

The screenshot shows a C++ IDE with a file named `main.cpp`. The code is as follows:

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n;
7      cin >> n;
8      for (int i = 1; i <= n; i++) {
9          for (int j = 1; j <= n; j++) {
10             cout << "* ";
11          }
12          cout << endl;
13      }
14      return 0;
15  }
16

```

Below the code editor, a console window titled `"D:\TD\Cpp\2 godina\bin\Debug\2 godina.ex..."` displays the output of the program. It shows a 5x5 grid of asterisks, followed by the message `Process returned 0 (0x0) execution time : 1.186 s` and `Press any key to continue.`

Во претходниот пример се користат вгнездени циклуси. Ако `n` има вредност 5, ќе се извршат 25 повторувања. Ако `n` има вредност 1000, ќе се извршат милион повторувања. Ова покажува како може брзината на алгоритмот значително да се промени кога се зголемува големината на влезните податоци. Поради тоа, програмерите мора внимателно да размислуваат кога користат вгнездени циклуси и да анализираат дали има поефикасно решение. Во современите системи што обработуваат огромни количини податоци, оптимизацијата претставува еден од најважните фактори за успешно функционирање на програмите.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува оптимизација на алгоритам?
2. Зошто може два алгоритми да имаат различна брзина?
3. Како влијаат вгнездените циклуси врз сложеноста?
4. Зошто е важно да се избегнуваат непотребни операции?
5. Како влијае големината на влезните податоци врз брзината на програмата?



ЗАДАЧИ

1. Напиши програма што ќе пресметува колку броеви од 1 до n се деливи со 7.
2. Напиши програма што ќе внесува n броеви и ќе го пронаоѓа најголемиот број.
3. Напиши програма со вгнездени циклуси што ќе печати правоаголник од знакот #.
4. Анализирај колку проверки ќе се извршат ако програмата проверува секој елемент од низа со 1000 елементи.
5. Напиши две различни решенија за пресметување збир од 1 до n и спореди кое решение е поефикасно.



ЗАПОМНИ ШТО НАУЧИ

- Различни алгоритми можат да решаваат ист проблем со различна ефикасност.
- Оптимизацијата има цел да ги намали бројот на операции и потрошувачката на меморија.
- Вгнездените циклуси значително ја зголемуваат сложеноста.
- `break` може да помогне за намалување на непотребни проверки.
- Анализата на алгоритми е важна при работа со големи количини податоци.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите интернет сервиси и системи за вештачка интелигенција во секоја секунда се обработуваат милиони податоци. Доколку алгоритмите не се доволно оптимизирани, серверите би работеле многу побавно и би трошеле огромна количина ресурси. Поради тоа, голем дел од работата на професионалните програмери се состои во анализа и подобрување на алгоритмите. Во некои случаи, оптимизацијата на алгоритам може да ја забрза програмата стотици или илјадници пати.

2.5



КОМБИНИРАЊЕ УСЛОВИ И ЦИКЛУСИ

ВЕЌЕ ЗНАЕШ ДА...

- користиш if и switch структури;
- користиш for и while циклуси;
- анализираш ефикасност на алгоритми;
- користиш вгнездени циклуси;
- споредуваш различни програмски решенија.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува оптимизација на алгоритам?
2. Зошто break може да ја подобри ефикасноста?
3. Што се случува кога се користат два вгнездени циклуси?
4. Напиши програма што ќе пресметува збир од 1 до n.
5. Објасни зошто големите податоци бараат поефикасни алгоритми.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

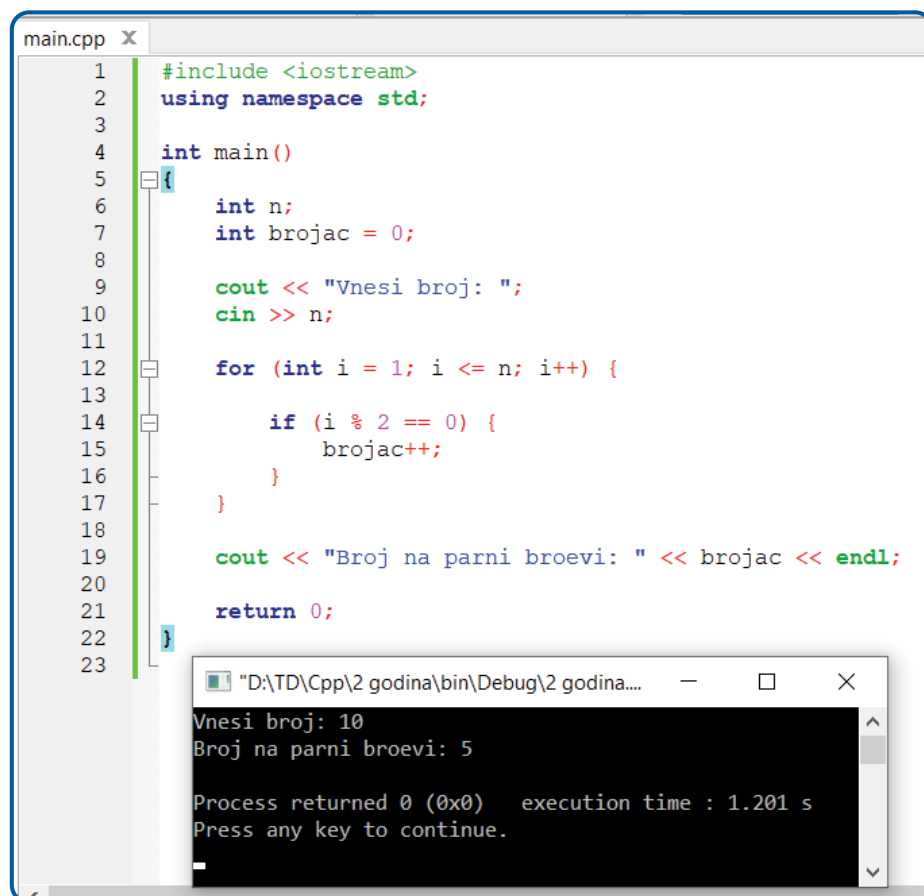
- комбинираш услови и циклуси во иста програма;
- решаваш посложени текстуални задачи;
- користиш вгнездени услови во циклуси;
- анализираш логика на повторување и избор;
- конструираш поорганизирани програмски решенија.

Во програмирањето многу проблеми не може да се решат само со услови или само со циклуси. Во реалните програми најчесто е потребно комбинирање на различни програмски структури за да се добие целосно решение. На пример, програмата може повеќепати да внесува податоци, а при секое внесување да проверува дали е исполнет определен услов. Ваквите ситуации бараат истовремена употреба на циклуси и услови.

Комбинирањето на програмските структури претставува важен чекор кон развој на посложени алгоритми. Со таквиот пристап програмите стануваат пофлексибилни и можат да обработуваат различни ситуации. Поради тоа, програмерите мора внимателно да анализираат каде треба да се постави условот, кога треба да се повторува постапката и како правилно да се организира логиката на програмата.

Циклусите овозможуваат повторување на определени инструкции, додека условите овозможуваат донесување одлуки. Кога се комбинираат овие структури, програмата може истовремено да повторува операции и да проверува различни услови. На пример, ако треба да се пребројат само парните броеви од 1 до n , циклусот ќе ги изминува сите броеви, а условот ќе проверува дали секој број е парен. На тој начин, условот се извршува во рамките на циклусот. Ваквиот начин на работа претставува една од најчестите техники во програмирањето. Во современите програми често се обработуваат голем број податоци, при што за секој податок се извршуваат различни проверки. Поради тоа, правилното комбинирање на услови и циклуси претставува основа за решавање посложени проблеми и изградба на ефикасни алгоритми.

Пример 1: Пребројување парни броеви



```
main.cpp X
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n;
7      int brojac = 0;
8
9      cout << "Vnesi broj: ";
10     cin >> n;
11
12     for (int i = 1; i <= n; i++) {
13
14         if (i % 2 == 0) {
15             brojac++;
16         }
17     }
18
19     cout << "Broj na parni broevi: " << brojac << endl;
20
21     return 0;
22 }
23
```

Output window: "D:\TD\Cpp\2 godina\bin\Debug\2 godina..."

```
Vnesi broj: 10
Broj na parni broevi: 5

Process returned 0 (0x0)   execution time : 1.201 s
Press any key to continue.
```

Во многу задачи е потребно условите да бидат вгнездени во циклуси или циклусите да бидат вгнездени во услови. Таквите конструкции овозможуваат решавање многу посложени проблеми. На пример, при анализа на ученици во паралелка може да биде потребно да се внесуваат оценки сè додека не се внесе одредена вредност, а потоа за секоја оценка да се провери дали е позитивна или негативна. Вак-

вите алгоритми бараат внимателна организација на логиката, затоа што погрешно поставен услов може да предизвика бесконечен циклус или неточен резултат. Поради тоа, програмерите често користат постепен пристап – најпрво го анализираат повторувањето, а потоа ги додаваат условите. Таквиот начин на работа овозможува полесно тестирање и подобра организација на кодот.

Пример 2: Збир на позитивни броеви

```

main.cpp X
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int broj;
7      int zbir = 0;
8
9      for (int i = 1; i <= 5; i++) {
10
11         cout << "Vnesi broj: ";
12         cin >> broj;
13
14         if (broj > 0) {
15             zbir += broj;
16         }
17     }
18
19     cout << "Zbirot na pozitivnite broevi e: " << zbir << endl;
20
21     return 0;
22 }
23

```

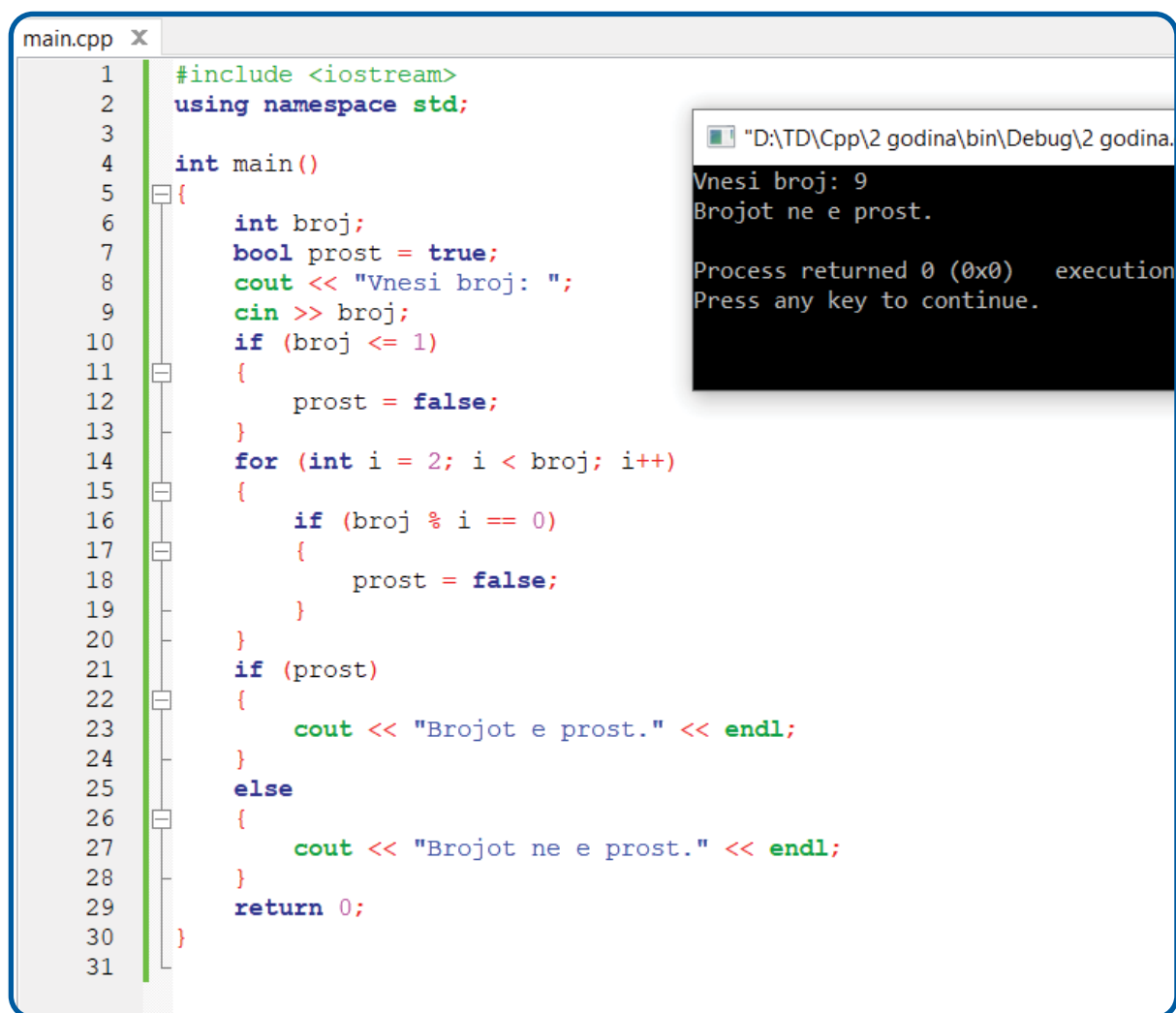
```

"D:\TD\Cpp\2 godina\bin\Debug\2 godina...
Vnesi broj: 2
Vnesi broj: -3
Vnesi broj: 4
Vnesi broj: -5
Vnesi broj: 1
Zbirot na pozitivnite broevi e: 7

Process returned 0 (0x0)   execution time : 8.145 s
Press any key to continue.

```

Комбинирањето услови и циклуси често се користи и при обработка на текстуални задачи. Во таквите задачи потребно е да се анализираат различни ситуации и да се избере соодветна постапка. На пример, програмата може да проверува дали е бројот прост, дали е делив со друг број или дали исполнува повеќе услови истовремено. Во вакви ситуации, условите може да бидат многу сложени и да содржат повеќе логички оператори. Освен тоа, циклусите често се користат за повторување на проверките над повеќе податоци. Поради тоа, ваквите програми претставуваат добра основа за развој на алгоритамско размислување и логичка анализа. Колку повеќе се комбинираат различни структури, толку програмите стануваат помоќни и посposобни за решавање реални проблеми.



The screenshot displays a C++ IDE with a file named `main.cpp`. The code implements a function to check if a number is prime. It includes the `<iostream>` header and uses the `std` namespace. The `main` function prompts the user to enter a number (`Vnesi broj:`). If the input is less than or equal to 1, it immediately declares the number not prime. For numbers greater than 1, it iterates from 2 to the input number, checking for divisibility. If any divisor is found, it sets a `prost` flag to `false`. After the loop, it checks the flag and prints the result: `"Brojot e prost."` if true, and `"Brojot ne e prost."` if false. The program returns 0.

Overlaid on the IDE is a terminal window showing the execution. The user has entered `9`, and the program has output `Brojot ne e prost.`. The terminal also shows the standard Windows command prompt messages: `Process returned 0 (0x0) execution completed.` and `Press any key to continue.`

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int broj;
7      bool prost = true;
8      cout << "Vnesi broj: ";
9      cin >> broj;
10     if (broj <= 1)
11     {
12         prost = false;
13     }
14     for (int i = 2; i < broj; i++)
15     {
16         if (broj % i == 0)
17         {
18             prost = false;
19         }
20     }
21     if (prost)
22     {
23         cout << "Brojot e prost." << endl;
24     }
25     else
26     {
27         cout << "Brojot ne e prost." << endl;
28     }
29     return 0;
30 }
31
```

"D:\TD\Cpp\2 godina\bin\Debug\2 godina.
Vnesi broj: 9
Brojot ne e prost.
Process returned 0 (0x0) execution completed.
Press any key to continue.

Во претходниот пример циклусот последователно ги проверува можните делители на бројот, додека условот проверува дали постои деливост без остаток. Комбинацијата на овие структури овозможува правилна анализа на бројот и добивање точен резултат. Ваквиот пристап се користи во голем број алгоритми каде што е потребна анализа на податоци, проверка на услови и повторување на постапки. Поради тоа, комбинирањето услови и циклуси претставува еден од најважните чекори во развојот на посложени програмски решенија.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Зошто во програмите се комбинираат услови и циклуси?
2. Што претставува вгнезден услов?
3. Каква улога има условот во рамките на циклус?
4. Зошто е важно правилно да се организира логиката на програмата?
5. Како помагаат циклусите при обработка на повеќе податоци?



ЗАДАЧИ

1. Напиши програма што ќе ги пребројува непарните броеви од 1 до n .
2. Напиши програма што ќе внесува 10 броја и ќе го пресметува збирот само на негативните броеви.
3. Напиши програма што ќе проверува дали внесен број е совршен број.
4. Напиши програма што ќе ги печати сите броеви деливи со 3 и 5 во определен интервал.
5. Напиши програма што ќе проверува колку внесени броеви се позитивни, а колку негативни.



ЗАПОМНИ ШТО НАУЧИ

- Циклусите и условите најчесто се користат заедно.
- Условите овозможуваат проверка на различни ситуации.
- Циклусите овозможуваат повторување постапки.
- Комбинирањето на структури овозможува решавање посложени проблеми.
- Правилната организација на логиката е многу важна во програмирањето.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите компјутерски системи огромен број алгоритми се базираат на комбинирање услови и повторувања. Пребарувачите на интернет, системите за препораки и вештачката интелигенција постојано обработуваат огромни количини податоци со користење различни услови и циклуси. Во некои програми се извршуваат милијарди проверки во секунда. Поради тоа, правилната организација на условите и циклусите има огромно влијание врз брзината и ефикасноста на современите системи.



2.6

РЕШАВАЊЕ КОМПЛЕКСНИ ТЕКСТУАЛНИ ЗАДАЧИ

ВЕЌЕ ЗНАЕШ ДА...

- комбинираш услови и циклуси;
- користиш вгнездени циклуси;
- анализираш ефикасност на алгоритми;
- оптимизираш едноставни програмски решенија;
- организираш логика во програма.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува оптимизација на алгоритам?
2. Зошто вгнездените циклуси можат значително да ја забават програмата?
3. Напиши програма што ќе пребројува колку броеви од 1 до n се деливи со 5.
4. Што претставува непотребна операција во алгоритам?
5. Објасни зошто различни решенија можат да имаат различна ефикасност.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- анализираш комплексни текстуални задачи;
- издвојуваш потребни податоци и услови;
- конструираш алгоритми во повеќе чекори;
- комбинираш повеќе програмски техники;
- тестираш и подобруваш програмски решенија.

Во програмирањето многу задачи не се решаваат со една едноставна пресметка или со користење само една програмска структура. Често е потребно истовремено да се анализираат повеќе услови, да се обработуваат голем број податоци и да се организираат повеќе чекори во една програма. Ваквите задачи се нарекуваат комплексни текстуални задачи, бидејќи бараат подетална анализа и повнимателна организација на алгоритмот.

При решавање комплексни проблеми, многу е важно задачата најпрво внимателно да се прочита и да се издвојат клучните информации. Потребно е да се одредат влезните податоци, условите што треба да се проверуваат и резултатот што треба да се добие. Потоа, проблемот постепено се дели на помали чекори што можат полесно да се реализираат во програма. Таквиот пристап овозможува полесно откривање грешки и создавање поорганизирани решенија.

При анализа на комплексна текстуална задача, првиот чекор претставува правилно разбирање на проблемот. Во многу случаи задачата содржи повеќе услови што мора истовремено да се исполнат. Доколку програмерот не ги анализира внимателно сите барања, програмата може да дава неточни резултати. Поради тоа, често се користи постепен пристап при кој проблемот се дели на повеќе подзадачи. На пример, ако треба да се анализираат резултати од тестирање на ученици, најпрво може да се пребројат положените ученици, потоа да се пресмета просекот, а на крај да се прикаже статистика. Со таквата поделба алгоритмот станува појасен и полесен за реализација. Во современото програмирање, ваквиот начин на размислување претставува основа за решавање сложени проблеми и развој на поголеми софтверски системи.

Пример 1: Анализа на оценки

```

main.cpp x
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n;
7      int oценка;
8      int polozeni = 0;
9
10     cout << "Kolku oцenki ke se vnesuvaat? ";
11     cin >> n;
12
13     for (int i = 1; i <= n; i++)
14     {
15         cin >> oценка;
16         if (oценка >= 2)
17         {
18             polozeni++;
19         }
20     }
21     cout << "Broj na polozeni ucenici: " << polozeni << endl;
22     return 0;
23 }
24

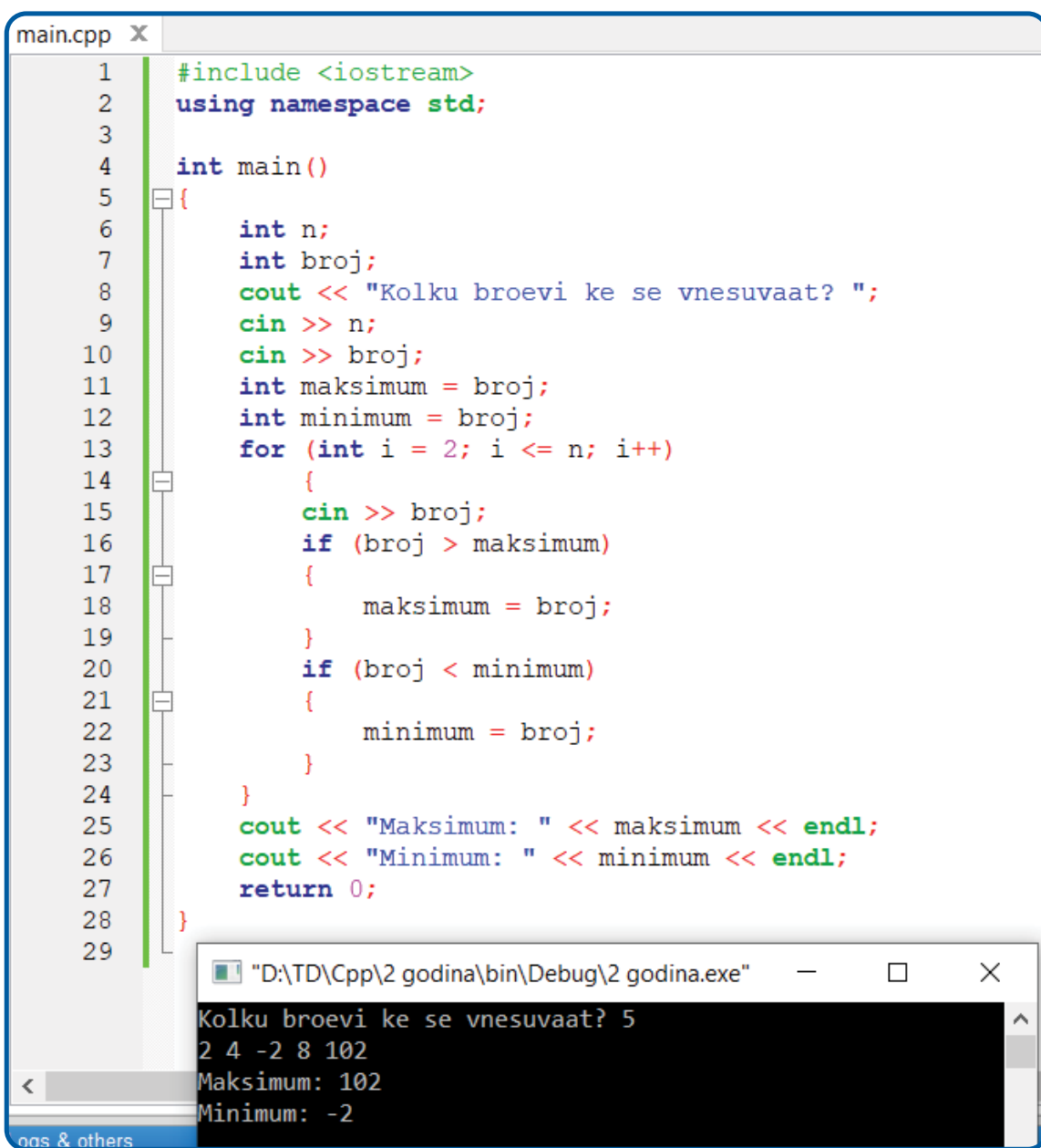
```

Output window: "D:\TD\Cpp\2 godina\bin\Debug\2 godina...
Kolku oцenki ke se vnesuvaat? 5
1
2
3
4
5
6
Broj na polozeni ucenici: 4

Комплексните задачи често бараат комбинирање на повеќе програмски техники во иста програма. Во една задача може истовремено да се користат циклуси, услови, бројачи и различни пресметки. Поради тоа, многу е важно програмата да биде добро организирана и логички поделена. Програмерите често користат дополнителни променливи за следење на различни состојби во програмата. На пример, при анализа на броеви може истовремено да се следи најголема вредност, најмала вредност и број на позитивни броеви. Ваквите задачи бараат внимателно планирање на алгоритмот пред започнување на програмирањето. Доколку логиката не биде правилно организирана, програмата може да стане тешка

за разбирање и одржување. Затоа, постепеното конструирање на решението претставува многу важен дел од програмирањето.

Пример 2: Пронаоѓање најголема и најмала вредност



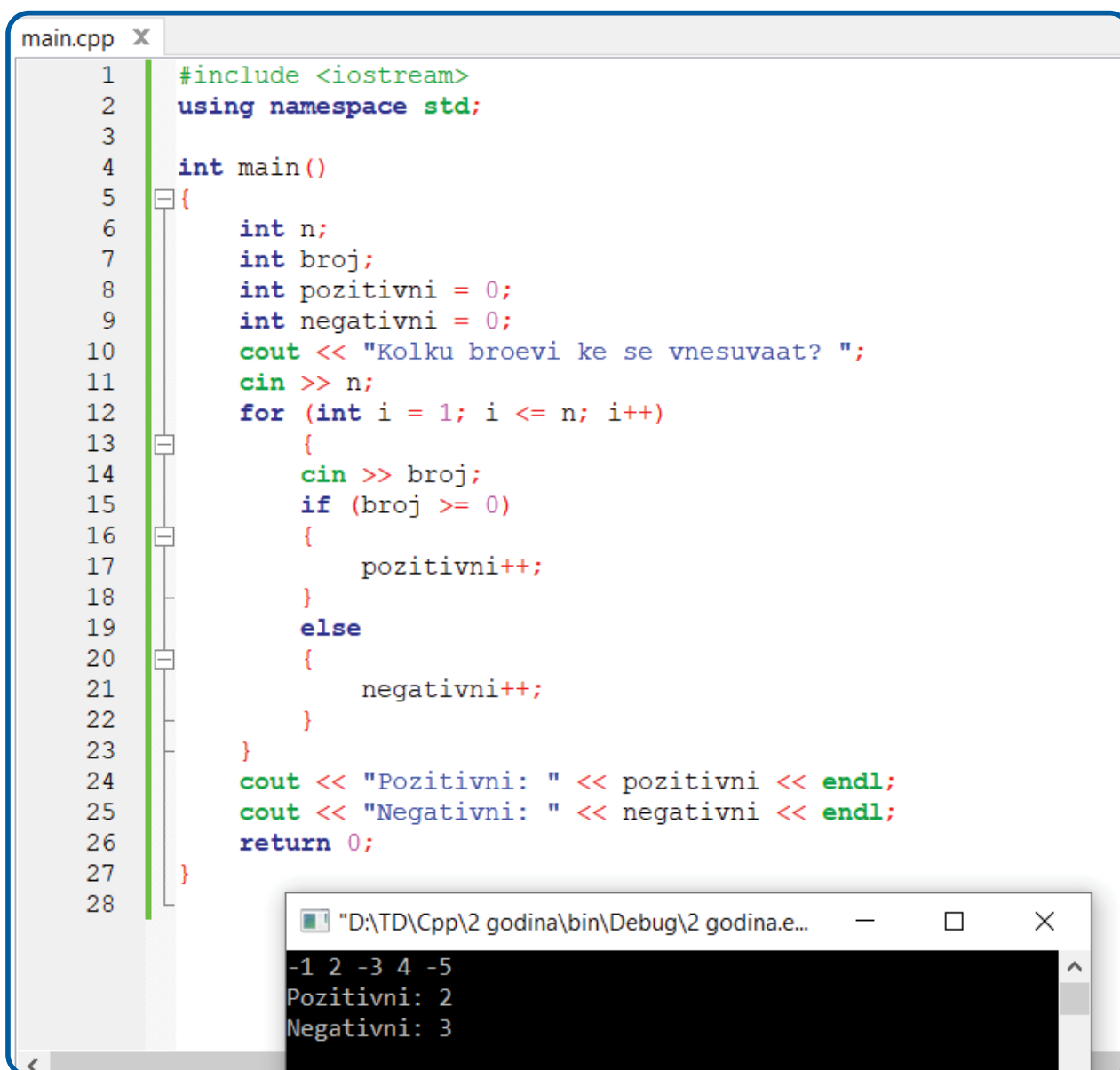
```
main.cpp X
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n;
7      int broj;
8      cout << "Kolku broevi ke se vnesuvaat? ";
9      cin >> n;
10     cin >> broj;
11     int maksimum = broj;
12     int minimum = broj;
13     for (int i = 2; i <= n; i++)
14     {
15         cin >> broj;
16         if (broj > maksimum)
17         {
18             maksimum = broj;
19         }
20         if (broj < minimum)
21         {
22             minimum = broj;
23         }
24     }
25     cout << "Maksimum: " << maksimum << endl;
26     cout << "Minimum: " << minimum << endl;
27     return 0;
28 }
29
```

"D:\TD\Cpp\2 godina\bin\Debug\2 godina.exe" — □ ×

```
Kolku broevi ke se vnesuvaat? 5
2 4 -2 8 102
Maksimum: 102
Minimum: -2
```

При решавање посложени задачи, многу е важно алгоритмот да се тестира со различни влезни податоци. Некои програми правилно работат за мали вредности, но даваат грешки при поголеми или неочекувани податоци. Поради тоа, програмерите постојано тестираат различни ситуации за да проверат дали е програмата стабилна и точна. Особено е важно да се анализираат граничните случаи, како внесување нула, негативни броеви или многу големи вредности. Со ваквото тестирање полесно се откриваат логички грешки и проблеми во алгоритмот. Во професионалното програмирање, тестирањето претставува задолжителен дел од развојот на секоја програма.

Пример 3: Анализа на позитивни и негативни броеви



```
main.cpp x
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n;
7      int broj;
8      int pozitivni = 0;
9      int negativni = 0;
10     cout << "Kolku broevi ke se vnesuvaat? ";
11     cin >> n;
12     for (int i = 1; i <= n; i++)
13     {
14         cin >> broj;
15         if (broj >= 0)
16         {
17             pozitivni++;
18         }
19         else
20         {
21             negativni++;
22         }
23     }
24     cout << "Pozitivni: " << pozitivni << endl;
25     cout << "Negativni: " << negativni << endl;
26     return 0;
27 }
28
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.e...
-1 2 -3 4 -5
Pozitivni: 2
Negativni: 3
```

Комплексните текстуални задачи претставуваат важен дел од развојот на алгоритамското размислување. При нивното решавање не е доволно само да се познава синтаксата на програмскиот јазик туку потребно е внимателно анализирање на проблемот и правилна организација на чекорите. Колку што задачите стануваат посложени, толку повеќе се развива способноста за логичко размислување, анализа и планирање на решенијата. Поради тоа, ваквите задачи претставуваат важен чекор кон развојот на посериозни програмски вештини.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува комплексна текстуална задача?
2. Зошто е важно задачата да се подели на помали чекори?
3. Зошто програмите треба да се тестираат со различни податоци?
4. Каква улога имаат бројачите во програмите?
5. Зошто е важна организацијата на алгоритмот?



ЗАДАЧИ

1. Напиши програма што ќе внесува n броеви и ќе го пресметува нивниот просек.
2. Напиши програма што ќе пребројува колку внесени броеви се деливи со 3.
3. Напиши програма што ќе го пронаоѓа вториот најголем број од внесените вредности.
4. Напиши програма што ќе проверува колку внесени оценки се одлични.
5. Напиши програма што ќе анализира колку од внесените броеви се парни, а колку непарни.



ЗАПОМНИ ШТО НАУЧИ

- Комплексните задачи бараат внимателна анализа.
- Проблемот најчесто се дели на повеќе помали чекори.
- Во сложени задачи се комбинираат повеќе програмски техники.
- Тестирањето е важен дел од развојот на програмите.
- Добрата организација на алгоритмот овозможува појасни и поефикасни решенија.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите софтверски системи програмерите ретко решаваат едноставни задачи. Голем дел од современите апликации обработуваат огромни количини податоци и истовремено проверуваат голем број услови. Банкарските системи, онлајн продавниците и социјалните мрежи користат комплексни алгоритми што постојано анализираат информации и донесуваат одлуки. Поради тоа, способноста за решавање комплексни текстуални задачи претставува една од најважните вештини во современото програмирање.

2.7



МОДУЛАРНО ПРОГРАМИРАЊЕ И КОНЦЕПТОТ „РАЗДЕЛИ ПА ВЛАДЕЈ“

ВЕЌЕ ЗНАЕШ ДА...

- користиш условни структури во C++;
- користиш циклуси за повторување на операции;
- комбинираш услови и циклуси во иста програма;
- ја анализираш ефикасноста на алгоритмите;
- решаваш посложени програмски задачи.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува оптимизација на алгоритам?
2. Зошто е важно да се намали бројот на непотребни операции во програмата?
3. Напиши програма што ќе пребројува колку броеви од 1 до n се деливи со 4.
4. Што претставува вгнезден циклус?
5. Објасни зошто поголемите програми се потешки за одржување од помалите.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што претставува модуларно програмирање;
- го применуваш принципот „раздели па владеј“;
- анализираш сложени проблеми и да ги делиш на помали задачи;
- организираш програмски решенија во логички целини;
- подготвуваш програми за користење функции.

Со зголемувањето на сложеноста на програмите, се појавува потреба од подобра организација на кодот и поефикасен пристап при решавање на проблемите. Малите програми може релативно лесно да се напишат како една целина, но кај поголемите програмски решенија таквиот пристап создава голем број тешкотии. Ако се сместат сите инструкции на едно место, програмата станува тешка за читање, раз-

бирање, тестирање и одржување. Поради тоа, во современото програмирање се користат техники кои овозможуваат сложените проблеми да се организираат на поуреден начин.

Еден од најважните принципи што се користат при решавање сложени проблеми е принципот „раздели па владеј“. Наместо целиот проблем да се разгледува како една голема задача, тој се разделува на повеќе помали задачи кои може полесно да се анализираат и да се решат. Откако секоја помала задача ќе биде успешно решена, добиените решенија се комбинираат во едно целосно решение. Ваквиот пристап претставува основа на модуларното програмирање и се користи во речиси сите современи програмски јазици и софтверски системи.

Во секојдневниот живот многу проблеми природно се решаваат со поделба на повеќе помали активности. Кога се организира училишна екскурзија, не се решаваат сите задачи истовремено. Најпрво се планира маршрута, потоа се организира превоз, се резервира сместување и на крај се подготвуваат сите потребни документи. Секоја од овие активности претставува посебна задача која може независно да се анализира и да се реализира. Истата идеја се применува и во програмирањето. Наместо програмерот да се обидува одеднаш да го реши целиот проблем, тој го разделува на повеќе логички целини. На тој начин решението станува поразбирливо и полесно за развој. Дополнително, грешките полесно се откриваат, затоа што секој дел од програмата може да се анализира одделно. Ваквиот начин на работа е суштината на модуларното програмирање. Модулите најчесто извршуваат точно определени задачи и меѓусебно соработуваат за да се добие конечниот резултат. Иако во оваа лекција сè уште нема да се користат функции, ќе се започне со развивање на начинот на размислување што подоцна ќе овозможи полесно разбирање на функциите и нивната примена во програмите.

Пример 1: Анализа на проблем

Задача: Да се пресмета успехот на ученик/чка врз основа на пет оценки.

Проблемот може да се подели на следните помали задачи:

Внесување на оценките.

Пресметување на збирот на оценките.

Пресметување на просекот.

Одредување на успехот.

Прикажување на резултатот.

На овој начин, наместо да се анализира целиот проблем одеднаш, тој се разделува на повеќе едноставни чекори.

Откако ќе се подели проблемот на помали задачи, следниот чекор претставува нивна организација. Многу е важно секоја задача да има јасно определена улога. На пример, делот што е задолжен за внесување податоци не треба истовремено да врши сложени пресметки. Исто така, делот што пресметува резултати не треба да биде задолжен за прикажување на податоците на екранот. Ваквата поделба овозможува програмата да биде поорганизирана и полесна за разбирање. Во големите програмски системи, една програма може да содржи десетици или стотици логички целини кои меѓусебно комуницираат. Ако не е добра организацијата, кодот станува хаотичен и тешко може да се надградува. Поради тоа, уште во раните фази на изучување на програмирањето треба да се развива навика за поделба на проблемот и организирање на решението во јасни логички целини. Ваквиот начин на размислување подоцна значително го олеснува користењето на функции, библиотеки и други програмски техники.

Пример 2: Псевдокод според принципот „раздели па владеј“

Задача: Да се пресмета конечната цена на производ со можен попуст.

Проблемот може да се подели на следните помали задачи:

Внесување цена на производот.

Внесување количина.

Пресметување на вкупната сума.

Проверка дали следува попуст.

Пресметување на конечната цена.

Прикажување на резултатот.

Псевдокод:

Почеток

Внеси cena

Внеси kolicina

Presmetaj suma = cena * kolicina

Ако suma > 5000

Presmetaj popust = suma * 0.10

Presmetaj konecnaCena = suma - popust

Инаку

konecnaCena = suma

Крај ако

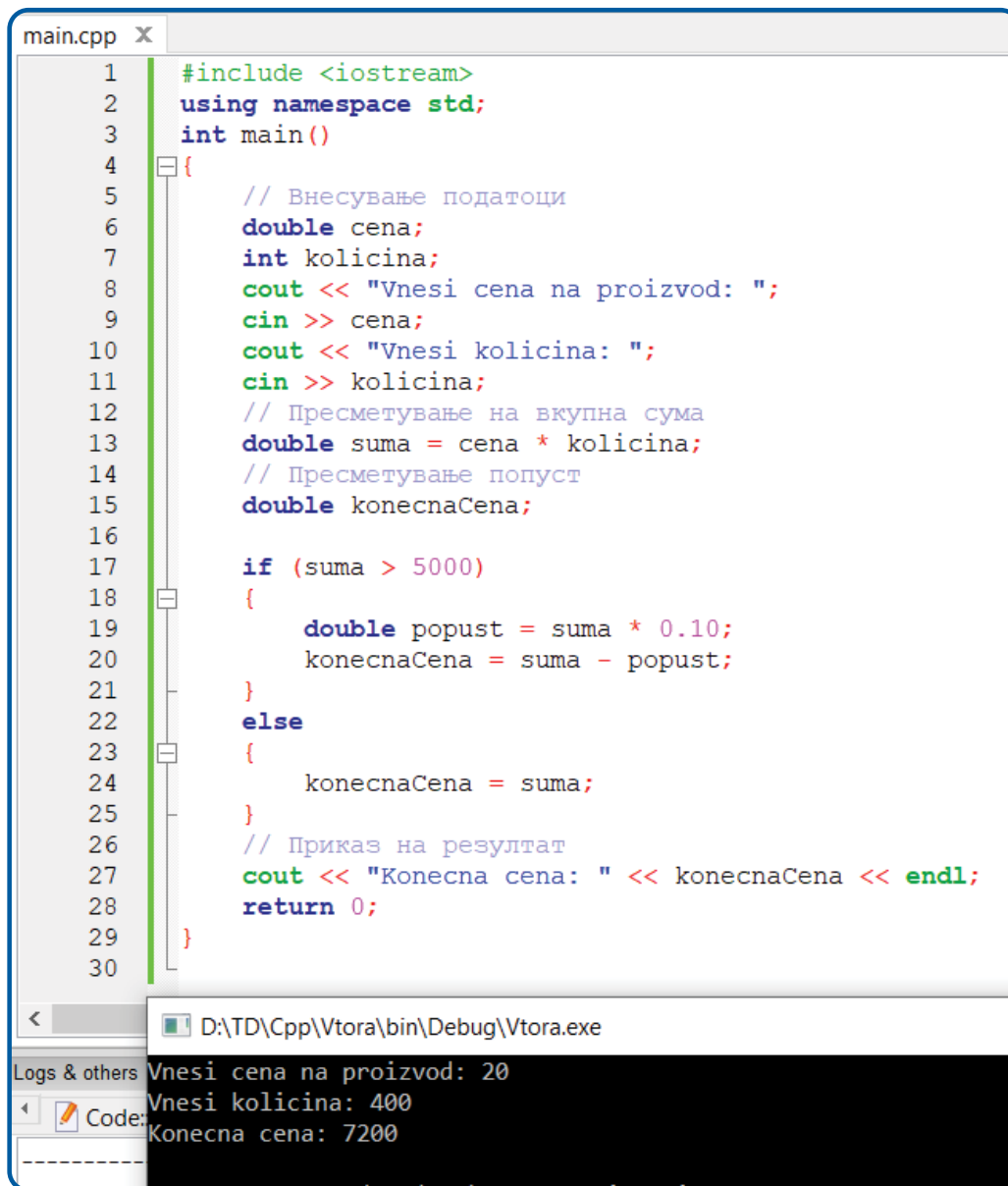
Прикажи konecnaCena

Крај

Во овој пример може да се забележи дека проблемот е најпрво поделен на неколку помали задачи. Секоја задача има јасно определена улога во решението. Благодарение на ваквата организација, алгоритмот станува полесен за разбирање и е подготвен за понатамошна реализација во програмски јазик. Токму ваквиот начин на размислување ја претставува суштината на принципот „раздели па владеј“ и е основа на модуларното програмирање.

Пример 3: Програма организирана во логички целини

Во следниот пример ќе биде прикажано како може да се организира една програма во повеќе логички целини иако сè уште не се користат функции. Програмата пресметува конечна цена на производ со можен попуст.



```
main.cpp x
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      // Внесување податоци
6      double cena;
7      int kolicina;
8      cout << "Vnesi cena na proizvod: ";
9      cin >> cena;
10     cout << "Vnesi kolicina: ";
11     cin >> kolicina;
12     // Пресметување на вкупна сума
13     double suma = cena * kolicina;
14     // Пресметување попуст
15     double konecnaCena;
16
17     if (suma > 5000)
18     {
19         double popust = suma * 0.10;
20         konecnaCena = suma - popust;
21     }
22     else
23     {
24         konecnaCena = suma;
25     }
26     // Приказ на резултат
27     cout << "Konecna cena: " << konecnaCena << endl;
28     return 0;
29 }
30
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

Logs & others Vnesi cena na proizvod: 20
Vnesi kolicina: 400
Konecna cena: 7200

Во програмата може јасно да се забележат три логички целини. Најпрво се врши внесување на податоците, потоа се обработуваат преку пресметување на сумата и попустот, а на крај се прикажува резултатот. Иако е програмата напишана во рамките на `main()` функцијата, логичките целини веќе може лесно да се препознаат. Во следните лекции овие целини ќе почнат да се издвојуваат во посебни функции. На тој начин програмите ќе станат поуредни, полесни за разбирање и поедноставни за одржување. Ова е првиот практичен чекор кон вистинско модуларно програмирање.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува модуларно програмирање?
2. Што претставува принципот „раздели па владеј“?
3. Зошто сложените проблеми се делат на помали задачи?
4. Како помага модуларниот пристап при откривање грешки?
5. Зошто е важно програмата да биде организирана во логички целини?



ЗАДАЧИ

1. Анализирај програма за пресметување плата и подели ја на логички целини.
2. Анализирај програма за пресметување потрошувачка на гориво и одреди кои задачи треба да се извршат по редослед.
3. Напиши псевдокод за програма што ќе пресметува просек од четири тестови.
4. За програма за евиденција на библиотека наведи најмалку четири логички целини.
5. Објасни како би помогнал принципот „раздели па владеј“ при изработка на училиштен информациски систем.



ЗАПОМНИ ШТО НАУЧИ

- Модуларното програмирање претставува организација на програмата во помали логички целини.
- Принципот „раздели па владеј“ овозможува сложените проблеми да се решаваат преку поделба на помали задачи.
- Добрата организација на програмата овозможува полесно разбирање и одржување на кодот.
- Логичките целини претставуваат основа за користење функции.
- Модуларното размислување претставува важна вештина за секој програмер.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во големите софтверски компании ретко кој програмер работи самостојно на целата програма. На место тоа, програмата се дели на голем број модули, а секој тим развива одреден дел од системот. На пример, кај една онлајн продавница еден тим може да работи на регистрација на корисници, друг на плаќање, трет на обработка на нарачки, а четврти на статистички извештаи. Иако секој тим работи на различен модул, сите модули на крајот функционираат како единствен систем. Токму затоа модуларното програмирање претставува една од најважните идеи во современиот развој на софтвер и е основа за создавање големи и сложени програмски решенија.



2.8

ДЕФИНИРАЊЕ ФУНКЦИИ ВО C++

ВЕЌЕ ЗНАЕШ ДА...

- објасниш што претставува модуларно програмирање;
- го применуваш принципот „раздели па владеј“;
- анализираш програмски проблем и да го делиш на помали задачи;
- организираш програма во логички целини;
- користиш услови и циклуси во C++.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува модуларно програмирање?
2. Што претставува принципот „раздели па владеј“?
3. Зошто големите проблеми се делат на помали задачи?
4. Наведи три логички целини што може да постојат во една програма.
5. Објасни како помага добрата организација на програмата при откривање грешки.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што претставува функција;
- дефинираш едноставни функции во C++;
- повикуваш функции од главната програма;
- разликуваш дефиниција и повик на функција;
- организираш програми со повеќе функции.

Како што стануваат поголеми програмите, се појавува потреба од подобра организација на кодот. Во претходната лекција беше објаснето дека сложените проблеми може да се поделат на повеќе помали задачи. Таквиот начин на размислување претставува основа на модуларното програмирање. Сепак, за да може оваа поделба практично да се реализира во програмскиот јазик, потребен е механизам што ќе овозможи секоја задача да се смести во посебна програмска целина.

Во програмскиот јазик C++ таа улога ја имаат функциите. Функцијата е дел од програмата (потпрограма) и претставува именуван блок од инструкции што извршува определена задача. Наместо истиот код повеќепати да се пишува во програмата, тој може да се смести во функција и потоа да се повикува кога е потребно. Благодарение на тоа, програмите стануваат поуредни, пократки и полесни за одржување. Поради оваа причина, функциите претставуваат една од најважните алатки во современото програмирање.

Функцијата може да се замисли како мала програма во рамките на поголема програма. Секоја функција има име преку кое може да биде повикана и содржи инструкции што ќе се извршат кога ќе биде активирана. Во секоја C++ програма има барем една функција – `main()`. Таа претставува почетна точка за извршување на програмата. Освен `main()`, програмерот може да создава и свои функции за извршување различни задачи. Ваквиот пристап овозможува програмата да се организира во повеќе делови наместо целиот код да биде сместен во една функција. Кога секоја функција има јасно определена задача, програмата станува многу полесна за читање и разбирање. Дополнително, ако се појави грешка, вниманието може да се насочи кон конкретната функција наместо кон целата програма. Поради тоа, користењето функции е важен чекор во развојот на добри програмерски навики.

При дефинирање функција, пред нејзиното име се наведува типот на функцијата. Типот определува дали функцијата ќе враќа резултат по своето извршување и каков ќе биде тој резултат.

Во оваа лекција се користи типот:

`void`

Клучниот збор `void` означува дека функцијата не враќа никаква вредност. Таквите функции најчесто се користат за извршување определена задача, како што е прикажување порака на екранот, печатење наслов или извршување одредени инструкции, без враќање резултат во програмата.

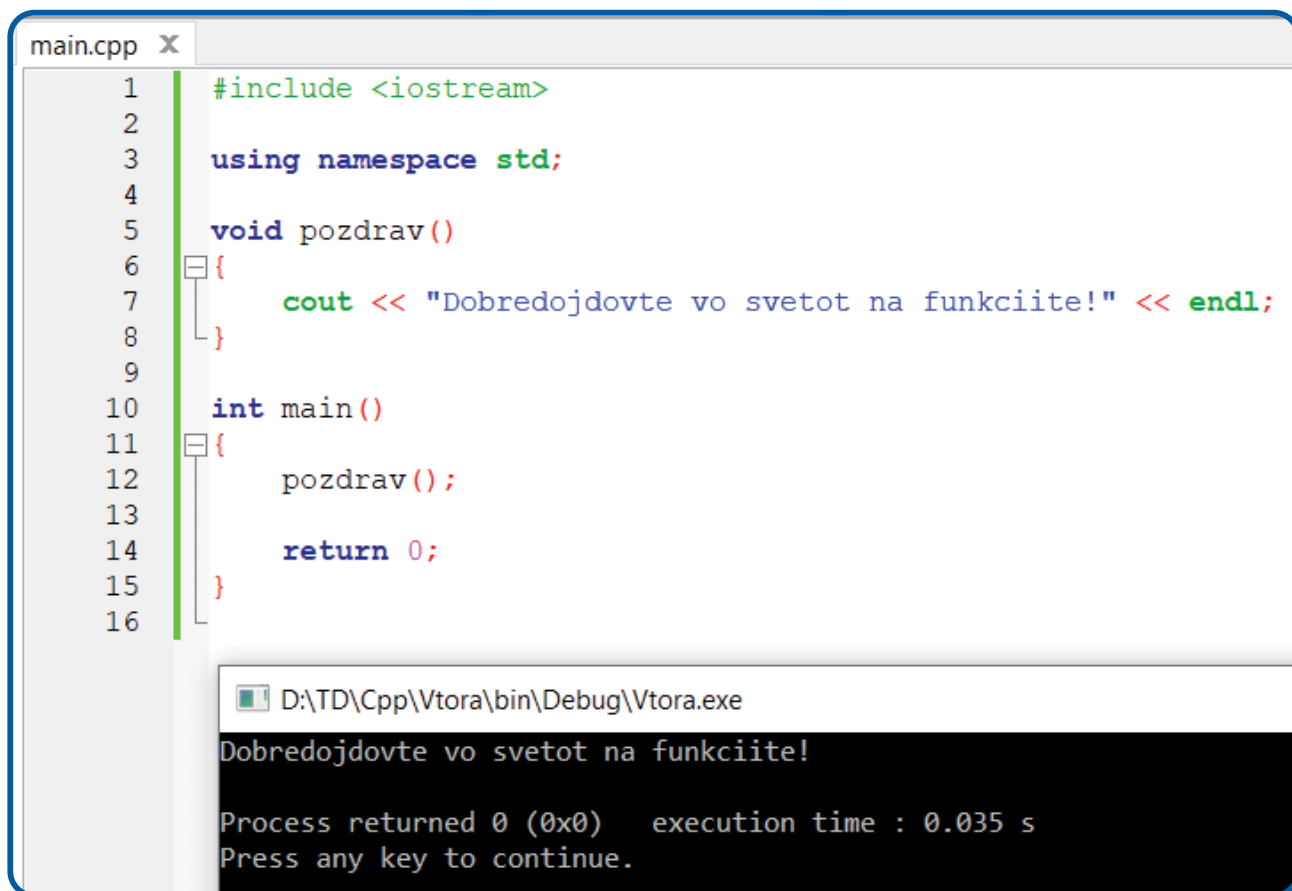
Општата форма на ваква функција е:

```
void imeNaFunkcija()
{
    инструкции
}
```

Во програмирањето се користат и функции што враќаат вредност, на пример:

```
int zbir()
{
    return 10;
}
```

Во овој случај функцијата враќа цел број (`int`), додека функцијата со тип `void` не враќа ништо.



```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  void pozdrav()
6  {
7      cout << "Dobredojdovte vo svetot na funkciite!" << endl;
8  }
9
10 int main()
11 {
12     pozdrav();
13
14     return 0;
15 }
16
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

Dobredojdovte vo svetot na funkciite!

Process returned 0 (0x0) execution time : 0.035 s
Press any key to continue.

Во примерот е дефинирана функција со име `pozdrav()`. Кога ќе биде повикана функцијата, ќе се испечати порака на екранот. Многу е важно да се забележи дека дефинирањето на функцијата не значи и нејзино автоматско извршување. За да се извршат инструкциите во функцијата, таа мора да биде повикана од програмата. Повикот се врши со наведување на името на функцијата и загради. Овој механизам овозможува истата функција да се користи повеќепати без повторно пишување на истиот код. Во реалните програми ова значително ја намалува должината на кодот и ја подобрува неговата читливост. Програмерите честопати создаваат функции за задачи што често се повторуваат. Така се избегнува непотребно дуплирање на инструкции и се подобрува организацијата на програмата.

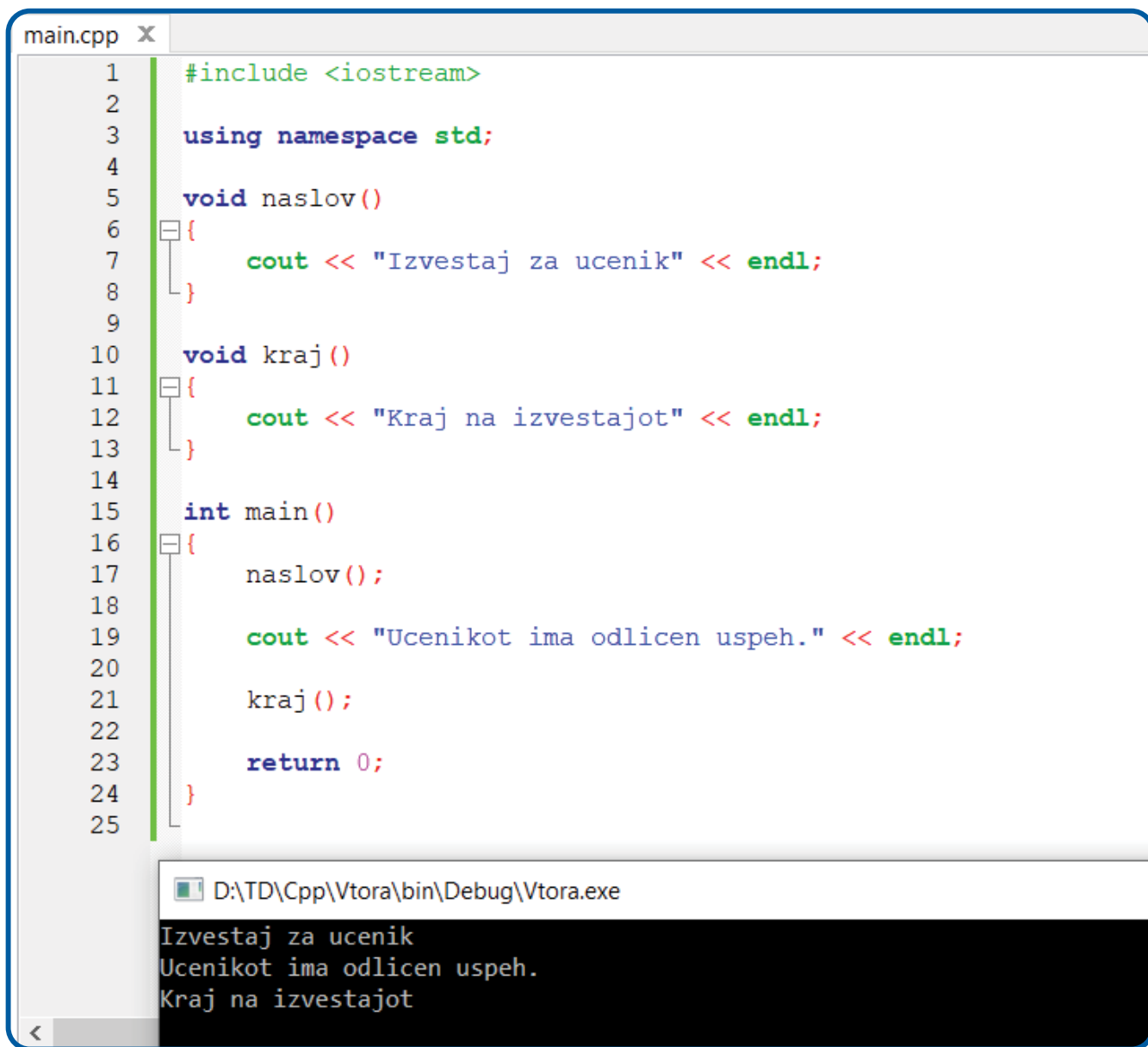
Пример 2: Повеќекратно повикување на функција

```

main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  void linija()
6  {
7      cout << "=====" << endl;
8  }
9
10 int main()
11 {
12     linija();
13     cout << "Prv del od programata" << endl;
14
15     linija();
16     cout << "Vtor del od programata" << endl;
17
18     linija();
19
20     return 0;
21 }
22
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
=====
Prv del od programata
=====
Vtor del od programata
=====

```

Во претходниот пример истата функција е повикана повеќепати. Наместо линијата од знаци секојпат повторно да се пишува во програмата, таа е сместена во функција. Благодарение на тоа, кодот е пократок и поуреден. Ова е една од најважните предности на функциите – можноста за повторна употреба на веќе напишан код. Во големите програми истата функција може да биде повикана десетици или стотици пати. Доколку е потребна промена во начинот на работа, доволно е да се измени кодот само во функцијата, без да се пребарува низ целата програма. Поради тоа, функциите значително ги олеснуваат развојот и одржувањето на софтверот.



```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  void naslov()
6  {
7      cout << "Izvestaj za ucenik" << endl;
8  }
9
10 void kraj()
11 {
12     cout << "Kraj na izvestajot" << endl;
13 }
14
15 int main()
16 {
17     naslov();
18
19     cout << "Ucenikot ima odlicen uspeh." << endl;
20
21     kraj();
22
23     return 0;
24 }
25
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

Izvestaj za ucenik
Ucenikot ima odlicen uspeh.
Kraj na izvestajot

Во овој пример се користат две различни функции. Едната е задолжена за прикажување наслов, а другата за прикажување завршна порака. На тој начин задачите во програмата се поделени на повеќе логички целини. Иако е мала програмата, лесно може да се забележи како функциите придонесуваат за подобра организација на кодот. Во следните лекции функциите ќе добијат дополнителни можности преку користење параметри и повратни вредности, со што ќе можат да извршуваат уште посложени задачи.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува функција?
2. Која е улогата на функцијата `main()`?
3. Што претставува повик на функција?
4. Која е разликата помеѓу дефиниција и повик на функција?
5. Зошто функциите ја подобруваат организацијата на програмите?



ЗАДАЧИ

1. Напиши функција што ќе прикажува порака „Добар ден!“.
2. Напиши функција што ќе прикажува линија составена од 30 знаци *.
3. Напиши програма што ќе користи две функции: една за прикажување наслов и една за прикажување завршна порака.
4. Напиши програма што трипати ќе повика иста функција.
5. Објасни како помагаат функциите при развој на големи програми.



ЗАПОМНИ ШТО НАУЧИ

- Функцијата претставува именуван блок од инструкции.
- Функциите овозможуваат поделба на програмата на помали целини.
- Дефиницијата ја опишува функцијата, а повикот ја активира.
- Истата функција може да биде повикана повеќепати.
- Функциите придонесуваат за подобра организација и одржување на програмите.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите програмски јазици постојат библиотеки што содржат илјадници веќе подготвени функции. Наместо програмерите секогаш да пишуваат код од почеток, тие можат да ги користат овие функции за математички пресметки, работа со текст, обработка на датотеки и многу други задачи. Благодарение на ваквиот пристап, развојот на софтвер станува побрз и поефикасен. Затоа функциите претставуваат една од најважните идеи во програмирањето и основа за развој на сложени софтверски системи.



2.9

ПАРАМЕТРИ И ВРАЌАЊЕ ВРЕДНОСТ

ВЕЌЕ ЗНАЕШ ДА...

- објасниш што претставува функција;
- дефинираш едноставни функции во C++;
- повикуваш функции од главната програма;
- разликуваш дефиниција и повик на функција;
- организираш програми во повеќе функции.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува функција?
2. Која е улогата на функцијата `main()`?
3. Што претставува повик на функција?
4. Напиши функција што ќе прикажува порака „Добредојде!“.
5. Зошто функциите придонесуваат за подобра организација на програмите?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- користиш параметри во функции;
- разликуваш параметри и аргументи;
- дефинираш функции што примаат податоци;
- користиш функции што враќаат вредност;
- ја применуваш наредбата `return`.

Функциите овозможуваат програмите да бидат организирани во повеќе логички целини и придонесуваат за подобра прегледност и одржување на кодот. При нивната употреба често се јавува потреба од обработка на различни податоци при секое повикување на функцијата. За таа цел во програмскиот јазик C++ се користат параметри преку кои до функцијата се предаваат вредности што треба да бидат

обработени. На овој начин се овозможува создавање пофлексибилни програмски решенија и повторна употреба на истиот код во различни ситуации.

Покрај примањето податоци, многу често е потребно функцијата да врати резултат што ќе може да се користи во друг дел од програмата. На пример, функцијата може да пресмета плоштина, квадрат на број или просек и добиениот резултат да го врати назад во програмата. Благодарение на оваа можност, функциите стануваат многу помоќни и пофлексибилни. Во оваа лекција ќе биде обработено користењето параметри и ќе започне изучувањето на функциите што враќаат вредност.

Кога функцијата треба да обработува различни податоци, тие податоци се испраќаат преку параметри. Параметрите претставуваат променливи што се наведуваат во заградите на функцијата. При секое повикување на функцијата, на параметрите им се доделуваат нови вредности. На тој начин истата функција може да се користи за обработка на различни податоци без повторно пишување на кодот. Ваквиот пристап значително ја зголемува повторната употребливост на програмите и овозможува создавање пофлексибилни решенија. Наместо да се создава посебна функција за секоја вредност, може да се создаде една функција што ќе работи со различни податоци. Поради тоа, параметрите се една од најважните карактеристики на функциите. Тие овозможуваат комуникација меѓу главната програма и функцијата и претставуваат основа за развој на посложени програмски решенија.

ПАРАМЕТРИ И АРГУМЕНТИ

При работа со функции често се користат поимите **параметар** и **аргумент**.

- Параметар е променлива наведена во дефиницијата на функцијата.
- Аргумент е вредност што се испраќа при повикување на функцијата.

Пример 1: Функција со еден параметар

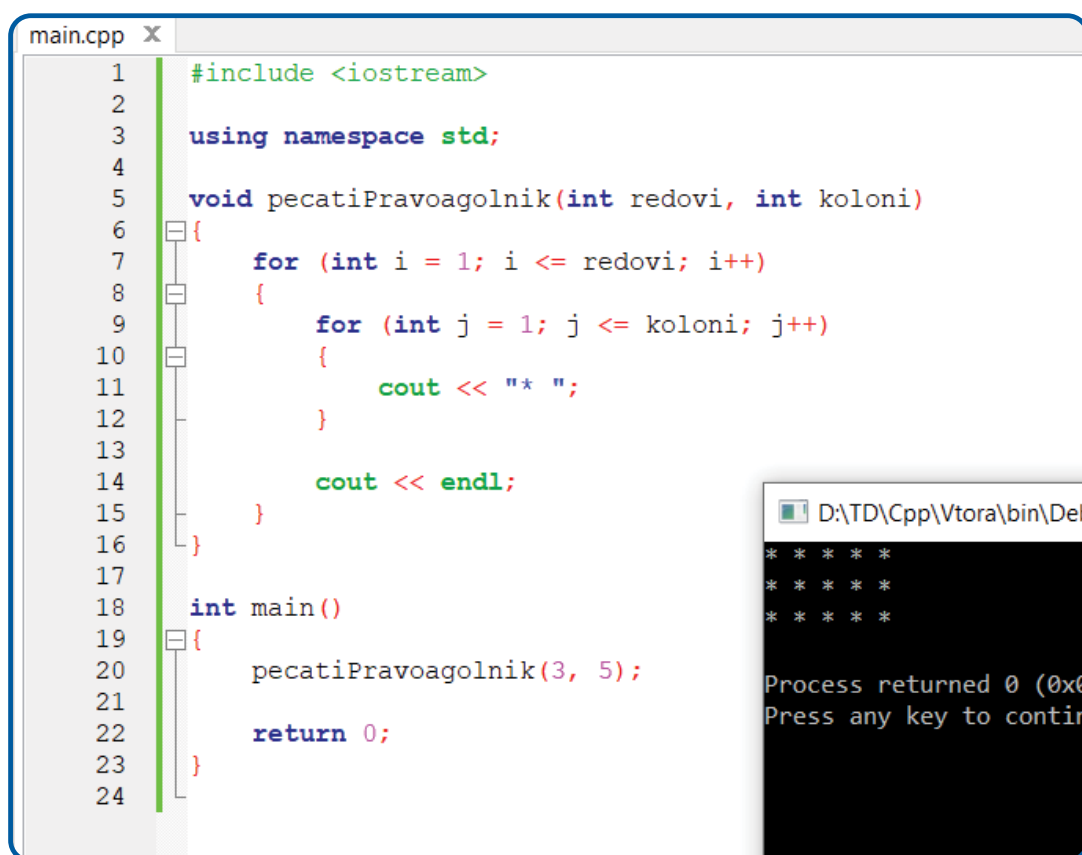
```

main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  void prikaziBroj(int broj)
6  {
7      cout << "Vneseniot broj e: " << broj << endl;
8  }
9
10 int main()
11 {
12     prikaziBroj(15);
13
14     return 0;
15 }
16
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Vneseniot broj e: 15
Process returned 0 (0x0)   execution time : 0.036 s
Press any key to continue.
  
```

Во примерот функцијата прима една вредност преку параметарот број. При повикувањето на функцијата, аргументот 15 се пренесува во параметарот и потоа се прикажува на екранот. Благодарение на параметрите, истата функција може да се користи со различни вредности без никакви измени во нејзиниот код. Ваквиот пристап значително ја зголемува флексибилноста на програмите и овозможува создавање пократок и поорганизиран код.

Функциите можат да примаат повеќе параметри истовремено. Во такви случаи секој параметар добива своја вредност при повикувањето на функцијата. Ова овозможува функцијата да работи со повеќе податоци и да извршува посложени задачи. Во практиката често се користат функции што примаат две, три или повеќе вредности. Притоа, редоследот на аргументите е многу важен, затоа што секоја вредност се доделува на соодветниот параметар според нејзината позиција. Ваквиот механизам претставува основа за комуникација на различните делови од програмата. Благодарение на него, функциите можат да се користат во голем број различни ситуации без потреба од создавање нови функции за секој поединечен случај.

Пример 2: Функција со два параметри



The screenshot shows a C++ IDE with a file named `main.cpp`. The code defines a function `pecatiPravoagolnik` that takes two integer parameters, `redovi` and `koloni`. The function uses nested `for` loops to print a right-angled triangle of asterisks. The first loop iterates over the number of rows (`redovi`), and the second loop iterates over the number of columns (`koloni`) for each row. The output of the program is a 3x5 grid of asterisks, as shown in the terminal window. The terminal window title is `D:\TD\Cpp\Vtora\bin\Det` and it shows the output: `* * * * *` on three lines. Below the output, it says `Process returned 0 (0x0)` and `Press any key to continue`.

```
1  #include <iostream>
2
3  using namespace std;
4
5  void pecatiPravoagolnik(int redovi, int koloni)
6  {
7      for (int i = 1; i <= redovi; i++)
8      {
9          for (int j = 1; j <= koloni; j++)
10         {
11             cout << "* ";
12         }
13
14         cout << endl;
15     }
16 }
17
18 int main()
19 {
20     pecatiPravoagolnik(3, 5);
21
22     return 0;
23 }
24
```

Во примерот функцијата прима два параметри. Првиот параметар го определува бројот на редови, а вториот бројот на колони. Благодарение на тоа, истата функција може да печати правоаголници со различни димензии, без потреба од промена на кодот во функцијата.

Ова се функции што примаат податоци преку параметри. Сепак, во многу ситуации не е доволно функцијата само да изврши определена задача. Често постои потреба резултатот од обработката да

биде вратен назад во програмата за да може понатаму да се користи. За таа цел се користат функции што враќаат вредност. За разлика од `void` функциите, кои не враќаат резултат, овие функции имаат определен тип податок што ќе биде вратен по нивното извршување. Враќањето на вредноста се реализира со помош на наредбата `return`. Кога ќе се изврши `return`, работата на функцијата завршува и добиената вредност се испраќа назад на местото од кое е повикана таа. Благодарение на оваа можност, функциите можат да се користат како дел од изрази, пресметки и други програмски конструкции. Ваквиот начин на работа значително ја зголемува нивната примена и овозможува изградба на покомплексни програмски решенија. Поради тоа, функциите што враќаат вредност се една од најважните алатки во програмирањето.

Функции што враќаат вредност

Општата форма на функција што враќа вредност е:

```
tip imeNaFunkcija(parametri)
{
    instrukcii

    return vrednost;
}
```

Наместо `void`, пред името на функцијата се наведува типот на податок што ќе биде вратен. На пример, ако функцијата враќа цел број, се користи типот `int`.

Пример 3: Функција што враќа вредност

The screenshot shows a C++ program in a text editor and its execution in a console window. The program defines a function `kvadrat` that takes an integer `broj` and returns its square. The `main` function calls `kvadrat(7)` and prints the result.

```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int kvadrat(int broj)
6  {
7      return broj * broj;
8  }
9
10 int main()
11 {
12     int rezultat;
13
14     rezultat = kvadrat(7);
15
16     cout << "Kvadratot e: " << rezultat << endl;
17
18     return 0;
19 }
20
```

The console window shows the output: "Kvadratot e: 49". Below the output, it says "Process returned 0 (0x0) execution time : 0.034 s" and "Press any key to continue.".

Во примерот функцијата `kvadrat()` прима еден параметар и враќа резултат со помош на наредбата `return`. Вредноста што се враќа потоа се зачувува во променливата `rezultat`. Благодарение на оваа можност, функциите можат да извршуваат пресметки и добиените резултати да ги предаваат на други делови од програмата. Ова претставува една од најчестите примени на функциите во современото програмирање.

Функциите што враќаат вредност често се користат за математички пресметки, анализа на податоци и решавање различни програмски проблеми. Наместо истата пресметка да се повторува повеќепати во програмата, таа може да се смести во функција и потоа да се користи секогаш кога ќе биде потребна. На овој начин кодот станува пократок и поразбирлив. Дополнително, ако е потребна промена во начинот на пресметување, доволно е да се измени само функцијата. Останатиот дел од програмата продолжува да работи без промени. Поради тоа, функциите придонесуваат за подобра организација, поголема повторна употребливост на кодот и полесно одржување на програмите. Во следните лекции овие концепти ќе бидат дополнително проширени преку изучување на различни начини на предавање податоци во функции и работа со променливи во нивниот опсег.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува параметар?
2. Што претставува аргумент?
3. Која е разликата помеѓу функција со тип `void` и функција што враќа вредност?
4. Која е улогата на наредбата `return`?
5. Зошто функциите со параметри се пофлексибилни од функциите без параметри?



ЗАДАЧИ

1. Напиши функција `recatilme()` што ќе прима име како параметар и ќе го прикажува на екранот.
2. Напиши функција `pogolemBroj()` што ќе прима два цели броја и ќе го враќа поголемиот број.
3. Напиши функција `plostinaPravoagolnik()` што ќе прима должина и ширина и ќе ја враќа плоштината.
4. Напиши функција `kub()` што ќе го враќа кубот на внесен број.
5. Напиши програма што ќе користи функција за пресметување на обем на квадрат и ќе го прикажува резултатот.



ЗАПОМНИ ШТО НАУЧИ

- Параметрите претставуваат променливи наведени во дефиницијата на функцијата.
- Аргументите се вредностите што се испраќаат при повикување на функцијата.
- Функциите можат да примаат еден или повеќе параметри.
- Наредбата `return` се користи за враќање вредност од функцијата.
- Функциите што враќаат вредност може да се користат во пресметки и во изрази.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во големите софтверски системи постојат функции што примаат десетици параметри и враќаат сложени резултати. Сепак, во професионалното програмирање се препорачува функциите да имаат јасна намена и да извршуваат една конкретна задача. Ваквиот пристап го прави кодот полесен за читање, тестирање и одржување. Многу современи програмски јазици содржат илјадници готови функции што програмерите секојдневно ги користат за работа со текст, датотеки, мрежи, графика и математички пресметки. Функциите се една од најважните основи на современото програмирање и нивното добро познавање значително го олеснува развојот на сложени програмски решенија.





2.10

ЛОКАЛНИ И ГЛОБАЛНИ ПРОМЕНЛИВИ КАЈ ФУНКЦИИ

ВЕЌЕ ЗНАЕШ ДА...

- дефинираш и повикуваш функции;
- користиш параметри во функции;
- разликуваш параметри и аргументи;
- користиш функции што враќаат вредност;
- ја применуваш наредбата return.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува параметар?
2. Што претставува аргумент?
3. Која е улогата на наредбата return?
4. Напиши функција што ќе прима број и ќе го прикажува на екранот.
5. Зошто функциите со параметри се пофлексибилни од функциите без параметри?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што претставува опсег на променлива;
- разликуваш локални и глобални променливи;
- анализираш каде може да се користи одредена променлива;
- користиш глобални променливи во повеќе функции;
- препознаваш предности и недостатоци на различни видови променливи.

При развојот на програмски решенија постојано се користат променливи за зачувување на податоци. Сепак, не сите променливи можат да бидат користени насекаде во програмата. Можноста за пристап до одредена променлива зависи од местото на кое е таа декларирана. Поради тоа, во програмскиот јазик C++ се користи поимот **опсег на променлива**.

Опсегот определува во кој дел од програмата може да биде користена определена променлива. Во зависност од местото на декларација, променливите може да бидат локални или глобални. Познавањето

на овие поими е многу важно бидејќи правилниот избор на видот на променлива придонесува за подобра организација на кодот и полесно откривање грешки во програмите.

Секоја променлива што се декларира во програмата има определен опсег на важност. Тоа значи дека постојат правила што определуваат каде може да биде користена и каде не може да биде пристапена. Кога променливата се декларира во рамките на функција, нејзината употреба е ограничена само на таа функција. Таквите променливи се нарекуваат локални променливи. Локалните променливи претставуваат најчесто користен вид променливи во програмите, затоа што овозможуваат подобра контрола врз податоците и ја намалуваат можноста за појава на несакани грешки. Секоја функција може да има свои локални променливи што не се видливи за останатите функции. Благодарение на тоа, различни функции можат независно да работат со сопствени податоци без да влијаат една на друга. Овој принцип придонесува за поголема сигурност и подобра организација на програмскиот код.

Пример 1: Локална променлива

```

main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  void poraka()
6  {
7      int broj = 10;
8
9      cout << broj << endl;
10 }
11
12 int main()
13 {
14     poraka();
15
16     return 0;
17 }
18

```

Execution output (D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe):

```

10

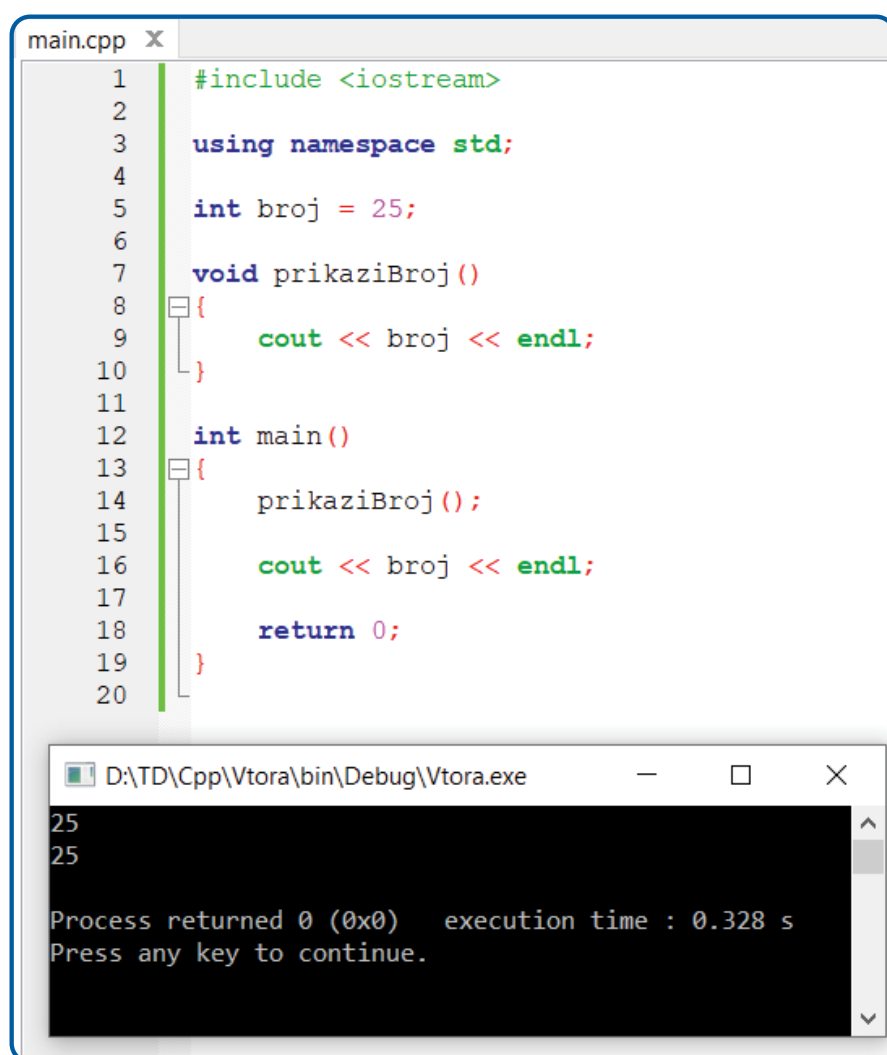
Process returned 0 (0x0)   executi
Press any key to continue.

```

Во примерот променливата `broj` е декларирана во функцијата `poraka()`. Поради тоа, таа може да биде користена само во рамките на таа функција. Ако се направи обид за нејзино користење во `main()` или во друга функција, програмата ќе пријави грешка. Ваквото однесување е резултат на ограничениот опсег на локалните променливи.

Покрај локалните променливи, во програмите може да се користат и глобални променливи. За разлика од локалните, глобалните променливи се декларираат надвор од сите функции. Поради тоа, тие можат да бидат пристапени од повеќе функции во истата програма. Ова овозможува одредени податоци да бидат споделени во различни делови на програмата. Сепак, ваквиот пристап треба да се користи внимателно. Кога голем број функции користат и менуваат иста глобална променлива, може да стане потешко да се следи текот на податоците и да се откријат евентуални грешки. Поради тоа, во современото програмирање најчесто им се дава предност на локалните променливи, а глобалните се користат само кога навистина има потреба за споделување на податоците.

Пример 2: Глобална променлива



```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int broj = 25;
6
7  void prikaziBroj()
8  {
9      cout << broj << endl;
10 }
11
12 int main()
13 {
14     prikaziBroj();
15
16     cout << broj << endl;
17
18     return 0;
19 }
20
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

```
25
25

Process returned 0 (0x0)   execution time : 0.328 s
Press any key to continue.
```

Во овој пример променливата `broj` е декларирана надвор од сите функции. Поради тоа, таа може да биде користена и во функцијата `prikaziBroj()` и во функцијата `main()`.

При работа со повеќе функции може да се појави ситуација во која локална и глобална променлива имаат исто име. Во таков случај, приоритет добива локалната променлива во рамките на функцијата во која е декларирана. Тоа значи дека во таа функција нема да биде користена глобалната променлива со истото име. Ваквата појава може да предизвика забуна кај програмерите, особено кај поголеми програми со голем број променливи. Поради тоа, при именување на променливите треба да се внимава да се избираат јасни и недвосмислени имиња. Добрата организација на променливите придонесува за подобра читливост на кодот и полесно откривање на логички грешки. Освен тоа, користењето локални променливи овозможува секоја функција да биде независна од останатите делови на програмата. На тој начин се намалува можноста ненамерно да биде променета вредноста што ја користи друга функција. Токму затоа локалните променливи најчесто претставуваат препорачан начин за работа со податоци во програмските решенија.

Пример 3: Локална и глобална променлива со исто име

```

main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int broj = 100;
6
7  void prikazi()
8  {
9      int broj = 20;
10
11     cout << "Lokalna promenлива: " << broj << endl;
12 }
13
14 int main()
15 {
16     prikazi();
17
18     cout << "Globalna promenлива: " << broj << endl;
19
20     return 0;
21 }
22

```

```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Lokalna promenлива: 20
Globalna promenлива: 100

Process returned 0 (0x0)   execution time : 0.323 s
Press any key to continue.

```

Во примерот има две променливи со исто име. Во функцијата prikazi() се користи локалната променлива со вредност 20, додека во main() се користи глобалната променлива со вредност 100. Овој пример покажува дека местото на декларација има големо влијание врз тоа која променлива ќе биде користена во програмата.

Изборот помеѓу локални и глобални променливи зависи од конкретниот проблем што се решава. Локалните променливи овозможуваат подобра контрола врз податоците и ја намалуваат можноста за појава на несакани промени во програмата. Од друга страна, глобалните променливи овозможуваат исти податоци да бидат користени во повеќе функции без потреба тие вредности постојано да се предаваат како параметри. Сепак, прекумерната употреба на глобални променливи може да ја направи програмата потешка за разбирање и одржување. Поради тоа, во практиката најчесто се препорачува прво да се користат локални променливи, а глобални променливи да се воведуваат само кога има јасна потреба за нивно користење во повеќе делови од програмата. Ваквиот пристап придонесува за создавање поуредни, посигурни и полесно одржливи програмски решенија.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува опсег на променлива?
2. Кога една променлива се смета за локална?
3. Кога една променлива се смета за глобална?
4. Која е основната разлика помеѓу локалните и глобалните променливи?
5. Зошто прекумерната употреба на глобални променливи може да создаде проблеми?



ЗАДАЧИ

1. Напиши програма што ќе содржи локална променлива во функција и ќе ја прикажува нејзината вредност.
2. Напиши програма што ќе користи една глобална променлива во две различни функции.
3. Напиши програма што ќе содржи локална и глобална променлива со различни имиња и ќе ги прикажува нивните вредности.
4. Анализирај во кои функции може да се користат глобална променлива декларирана надвор од сите функции.
5. Објасни во кои ситуации е подобро да се користат локална променлива наместо глобална.



ЗАПОМНИ ШТО НАУЧИ

- Опсегот на променливата определува каде може да биде користена.
- Локалните променливи се декларираат во рамките на функција и можат да се користат само во неа.
- Глобалните променливи се декларираат надвор од сите функции.
- Глобалните променливи може да бидат користени во повеќе функции.
- Локалните променливи најчесто овозможуваат подобра организација и поголема сигурност на програмата.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во големите програмски системи може да постојат илјадници променливи. Кога би биле сите тие глобални, следењето на нивните вредности би било исклучително тешко. Поради тоа, современите програмски практики препорачуваат ограничување на употребата на глобални променливи и користење локални променливи секогаш кога е можно. Ваквиот пристап овозможува подобра организација на програмскиот код, полесно тестирање на функциите и намалување на бројот на логички грешки при развојот на софтверски решенија.





ции се потребни посложени алгоритми. На пример, пресметувањето квадратен корен, степенување или тригонометриски функции би барало значително поголем број инструкции ако се реализираат рачно.

За олеснување на работата на програмерите, програмскиот јазик C++ содржи голем број готови функции сместени во различни библиотеки. Една од најчесто користените библиотеки е `cmath`, во која се наоѓаат функции за математички пресметки. Со нивна примена може брзо и едноставно да се извршуваат сложени операции без потреба од пишување дополнителен код. Благодарение на тоа, програмите стануваат пократки, поразбирливи и полесни за одржување.

Библиотеката `cmath` претставува збир од готови математички функции што може да се користат во програмите. За нивна употреба потребно е на почетокот од програмата да се вклучи библиотеката преку директивата `#include <cmath>`. Откако ќе биде вклучена библиотеката, програмата добива пристап до голем број функции што извршуваат различни математички операции. Наместо програмерот самостојно да развива алгоритам за пресметување квадратен корен или степен на број, може директно да ги искористи функциите што веќе се наоѓаат во библиотеката. Овој пристап овозможува намалување на времето потребно за развој на програмите и придонесува за поголема сигурност на резултатите. Функциите од `cmath` се темелно тестирани и се користат во голем број професионални софтверски решенија. Поради тоа, нивното познавање претставува важен дел од програмерските вештини.

Пример 1: Функцијата `sqrt()`

Функцијата `sqrt()` се користи за пресметување квадратен корен на број.

The image shows a code editor window titled 'main.cpp' with the following C++ code:

```

1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main()
7  {
8      double broj = 81;
9
10     double koren = sqrt(broj);
11
12     cout << "Kvadratniot koren e: " << koren << endl;
13
14     return 0;
15 }
16

```

Below the code editor, a console window titled 'D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe' displays the output:

```

Kvadratniot koren e: 9

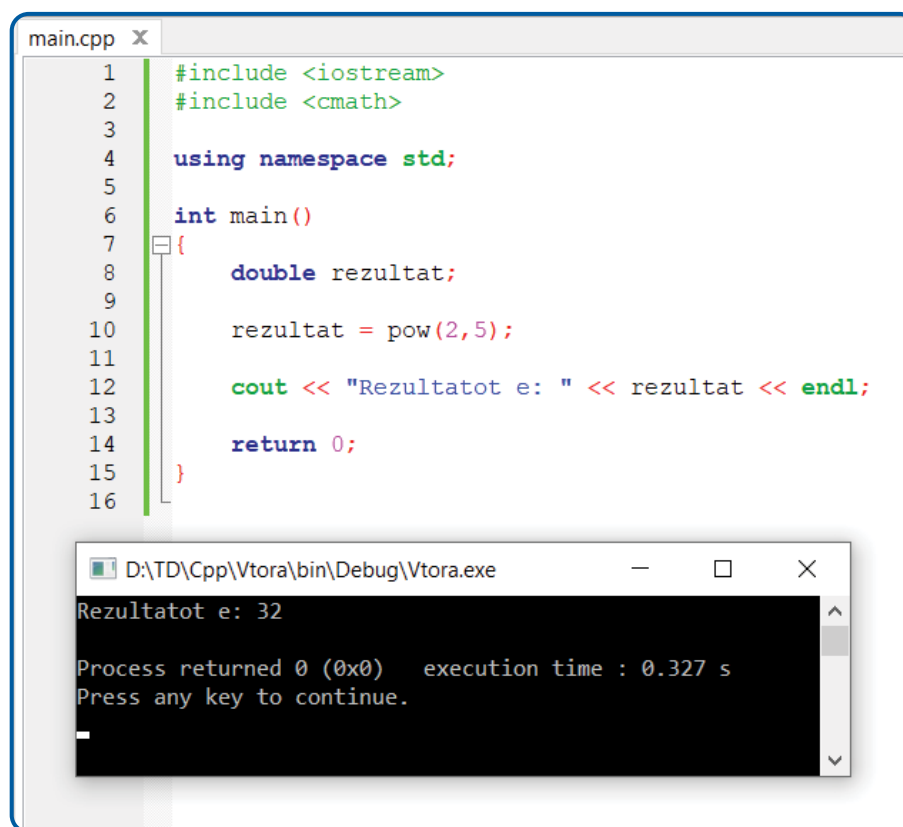
Process returned 0 (0x0)   execution time : 0.334 s
Press any key to continue.

```

Во примерот се пресметува квадратниот корен од бројот 81. Добиениот резултат е 9. Функцијата `sqrt()` прима број како аргумент и ја враќа вредноста на неговиот квадратен корен.

Покрај квадратниот корен, често се користи и степенување на броеви. Во математиката степенувањето претставува операција при која еден број се множи сам со себе определен број пати. Во библиотеката `cmath` оваа операција е реализирана преку функцијата `pow()`. Таа прима два аргументи. Првиот аргумент ја претставува основата, а вториот аргумент степенот на кој треба да биде кренат бројот. Благодарение на оваа функција, лесно може да се реализираат голем број математички формули и алгоритми. Употребата на готови функции придонесува кодот да биде пократок и поразбирлив во споредба со рачното применување на истите операции. Поради тоа, функцијата `pow()` е една од најчесто користените функции од библиотеката `cmath`.

Пример 2: Функцијата `pow()`



The screenshot shows a C++ IDE with a file named `main.cpp`. The code includes `<iostream>` and `<cmath>`, uses the `std` namespace, and defines a `main` function. Inside `main`, a `double` variable `rezultat` is declared and assigned the value of `pow(2, 5)`. The program then prints "Rezultatot e: " followed by the value of `rezultat` and ends with `endl`. The `main` function returns 0. Below the code editor, a console window titled `D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe` shows the output "Rezultatot e: 32". It also displays "Process returned 0 (0x0) execution time : 0.327 s" and "Press any key to continue."

```
1 #include <iostream>
2 #include <cmath>
3
4 using namespace std;
5
6 int main()
7 {
8     double rezultat;
9
10    rezultat = pow(2, 5);
11
12    cout << "Rezultatot e: " << rezultat << endl;
13
14    return 0;
15 }
16
```

Rezultatot e: 32

Process returned 0 (0x0) execution time : 0.327 s
Press any key to continue.

Во примерот се пресметува вредноста на изразот 2^5 . Добиениот резултат е 32. Првиот аргумент ја претставува основата, а вториот аргумент – степенот.

Покрај функциите за пресметување квадратен корен и степенување, библиотеката `cmath` содржи и функции за работа со апсолутни вредности и заокружување на броеви. Апсолутната вредност на еден број ја претставува неговата оддалеченост од нулата на бројната права и секогаш има ненегативна вредност. За нејзино пресметување се користи функцијата `fabs()`. При работа со реални броеви често се јавува потреба вредностите да бидат заокружени. За таа цел се користат функциите `ceil()` и `floor()`. Функцијата `ceil()` го заокружува бројот на првиот поголем цел број, додека функцијата `floor()` го заокружува бројот на првиот помал цел број. Овие функции имаат широка примена во различни програмски

решенија, особено кога е потребно добиените резултати да бидат претставени во форма на цели броеви. Благодарение на нив, програмите можат едноставно да обработуваат реални вредности и да ги приспособуваат на конкретните потреби за проблемот што се решава.

Некои често користени функции од библиотеката **cmath**

Функција	Намена
<code>sqrt(x)</code>	Квадратен корен
<code>pow(a,b)</code>	Степенување
<code>fabs(x)</code>	Апсолутна вредност
<code>ceil(x)</code>	Заокружување нагоре
<code>floor(x)</code>	Заокружување надолу
<code>sin(x)</code>	Синус
<code>cos(x)</code>	Косинус

Пример 3: Користење повеќе математички функции

```

main.cpp x
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main()
7  {
8      double broj = -15.7;
9
10     cout << "Apsolutna vrednost: "
11         << fabs(broj) << endl;
12
13     cout << "Zaokruzuvanje nagore: "
14         << ceil(broj) << endl;
15
16     cout << "Zaokruzuvanje nadolu: "
17         << floor(broj) << endl;
18
19     return 0;
20 }
21

```

```

D:\TD\C++\Vtora\bin\Debug\Vtora.exe
Apsolutna vrednost: 15.7
Zaokruzuvanje nagore: -15
Zaokruzuvanje nadolu: -16

Process returned 0 (0x0)   execution time : 1.645 s
Press any key to continue.

```

Во примерот се користат три различни функции од библиотеката `cmath`. Со функцијата `fabs()` се добива апсолутната вредност на бројот, додека функциите `ceil()` и `floor()` вршат различни видови заокружување. Ваквите операции често се користат при обработка на податоци и изработка на математички пресметки во програмите.

Во библиотеката `cmath` се наоѓаат и тригонометриски функции што имаат значајна примена во математиката, физиката, инженерството и во компјутерската графика. Најчесто користени се функциите `sin()` и `cos()`, кои ги пресметуваат синусот и косинусот на даден агол. При нивна употреба треба да се знае дека агли се задаваат во радијани. Иако во училишната математика најчесто се работи со степени, компјутерските програми вообичаено ги користат радијаните како мерна единица за агли. Тригонометриските функции се применуваат при изработка на симулации, игри, анимации, графички апликации и голем број научни програми. Благодарение на библиотеката `cmath`, овие пресметки може лесно да бидат реализирани без потреба од развивање сопствени математички алгоритми.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. За што се користи библиотеката `cmath`?
2. Која е намената на функцијата `sqrt()`?
3. Која е намената на функцијата `pow()`?
4. Која е разликата помеѓу `ceil()` и `floor()`?
5. За што се користат функциите `sin()` и `cos()`?



ЗАДАЧИ

1. Напиши програма што ќе го пресметува квадратниот корен на внесен број.
2. Напиши програма што ќе пресметува вредност на изразот 3^4 со користење на функцијата `pow()`.
3. Напиши програма што ќе ја прикажува апсолутната вредност на внесен реален број.
4. Напиши програма што ќе прикажува резултати од `ceil()` и `floor()` за внесен број.
5. Напиши програма што ќе пресметува синус и косинус на агол зададен во радијани.



ЗАПОМНИ ШТО НАУЧИ

- Библиотеката `cmath` содржи готови математички функции.
- Функцијата `sqrt()` се користи за пресметување квадратен корен.
- Функцијата `pow()` се користи за степенување.
- Функцијата `fabs()` ја пресметува апсолутната вредност на број.
- Функциите `ceil()` и `floor()` се користат за заокружување.
- Функциите `sin()` и `cos()` се користат за тригонометриски пресметки.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Библиотеката `cmath` содржи многу повеќе функции од оние што беа обработени во оваа лекција. Во неа може да се најдат функции за пресметување логаритми, експоненцијални функции, тригонометриски и инверзни тригонометриски функции, како и голем број други математички операции. Во професионалните програмски решенија овие функции се користат при развој на научни симулации, инженерски апликации, финансиски анализи, компјутерски игри и системи за обработка на податоци. Познавањето на библиотеката `cmath` му овозможува на програмерот значително побрзо да решава математички проблеми и да создава поефикасни програмски решенија.





2.12

РЕШАВАЊЕ ТЕКСТУАЛНИ ЗАДАЧИ СО ПРИМЕНА НА ФУНКЦИИ

ВЕЌЕ ЗНАЕШ ДА...

- дефинираш и повикуваш функции;
- користиш параметри и аргументи;
- користиш функции што враќаат вредност;
- разликуваш локални и глобални променливи;
- применуваш функции од библиотеката `cmath`.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. За што се користи библиотеката `cmath`?
2. Која е намената на функцијата `sqrt()`?
3. Која е разликата помеѓу `ceil()` и `floor()`?
4. Во кои ситуации се користат функции што враќаат вредност?
5. Која е улогата на наредбата `return`?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- анализираш текстуална задача;
- препознаваш кои делови од решението треба да бидат реализирани како функции;
- комбинираш повеќе функции во една програма;
- користиш параметри и повратни вредности при решавање проблеми;
- создаваш поорганизирани програмски решенија.

Функциите претставуваат една од најкорисните алатки за организација на програмскиот код. Нивната вистинска вредност најдобро се согледува при решавање реални проблеми, кога една програма треба да извршува повеќе различни задачи. Наместо сите инструкции да бидат сместени во функцијата `main()`, програмата може да биде поделена на повеќе помали целини, при што секоја функција добива јасно определена задача.

При решавање текстуални задачи најпрво треба да биде извршена анализа на проблемот. Потоа треба да биде утврдено кои операции се повторуваат и кои делови од решението може да бидат издвоени во посебни функции. Ваквиот пристап овозможува програмите да бидат поуредни, полесни за разбирање и полесни за одржување.

При решавање на програмски проблем најпрво треба да биде утврдено кои податоци се внесуваат, кои пресметки треба да бидат извршени и каков резултат треба да биде прикажан. Откако ќе биде направена оваа анализа, решението може да биде поделено на повеќе логички целини. Секоја целина може да биде реализирана како посебна функција. На тој начин се добива програма составена од повеќе помали делови наместо една голема и сложена целина. Ваквиот пристап го следи принципот „раздели па владеј“ и претставува основа на модуларното програмирање.

Пример 1: Пресметување цена на билети

Во кино се продаваат билети по цена од 250 денари. Да се пресмета вкупниот приход за внесен број продадени билети.

```

main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int prihod(int brojBileti)
6  {
7      return brojBileti * 250;
8  }
9
10 int main()
11 {
12     int brojBileti;
13
14     cout << "Vnesi broj na bileti: ";
15     cin >> brojBileti;
16
17     cout << "Prihod: "
18          << prihod(brojBileti)
19          << " denari" << endl;
20
21     return 0;
22 }
23
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Vnesi broj na bileti: 15
Prihod: 3750 denari

Process returned 0 (0x0)   execution time : 1.607 s
Press any key to continue.

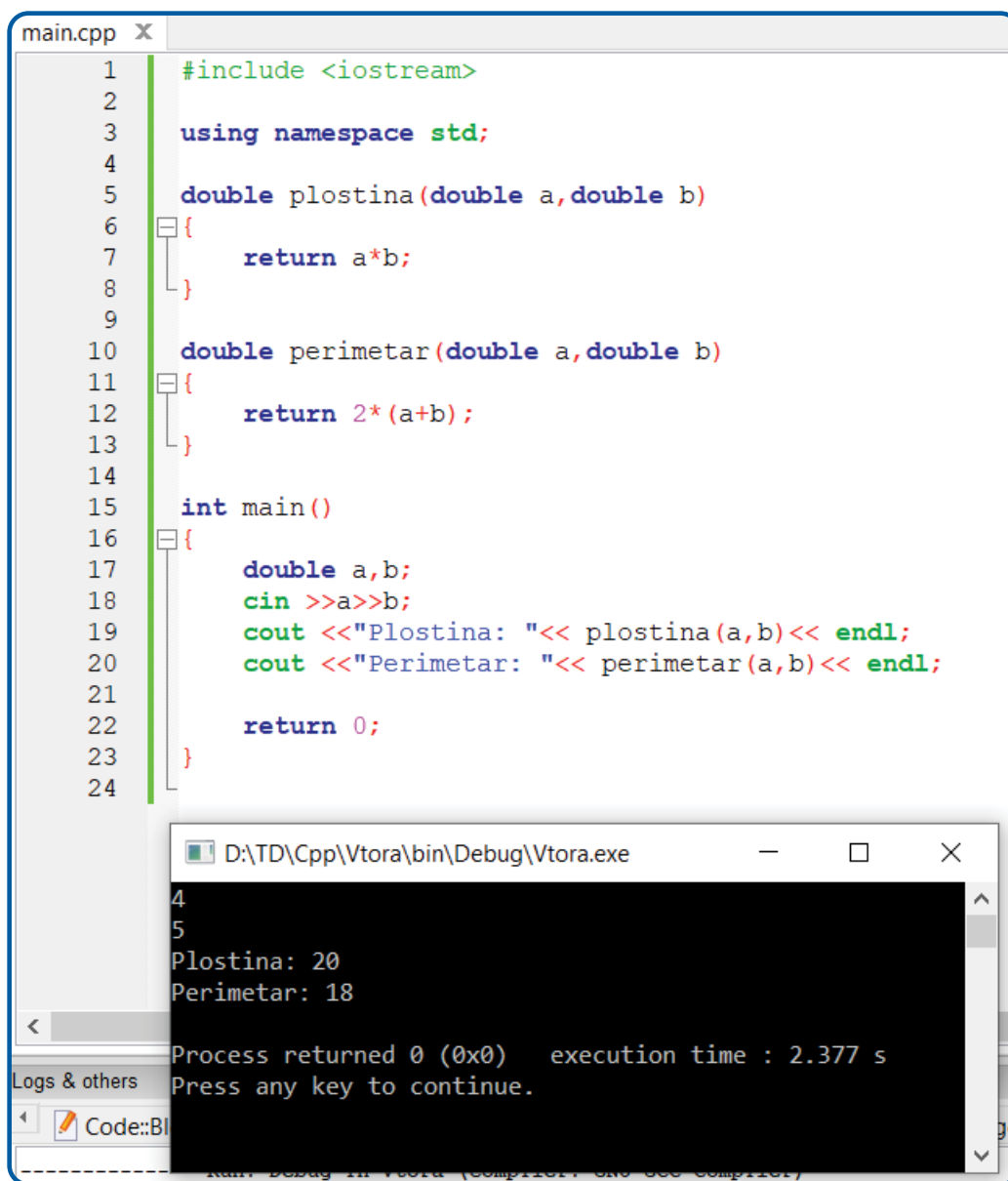
```

Во примерот пресметувањето на приходот е издвоено во посебна функција. На тој начин решението е поуредно и полесно за одржување.

Во многу ситуации е потребно една програма да користи повеќе функции. Секоја функција може да извршува различна задача, а потоа добиените резултати да се комбинираат во главната програма. Ваквиот пристап е особено корисен кога проблемот содржи повеќе пресметки што може независно да бидат реализирани.

Пример 2: Пресметување плоштина и периметар

Да се внесат должина и ширина на правоаголник. Да се пресметаат плоштината и периметарот.



```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  double plostina(double a, double b)
6  {
7      return a*b;
8  }
9
10 double perimetar(double a, double b)
11 {
12     return 2*(a+b);
13 }
14
15 int main()
16 {
17     double a, b;
18     cin >> a >> b;
19     cout << "Plostina: " << plostina(a, b) << endl;
20     cout << "Perimetar: " << perimetar(a, b) << endl;
21
22     return 0;
23 }
24
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

```
4
5
Plostina: 20
Perimetar: 18

Process returned 0 (0x0)   execution time : 2.377 s
Press any key to continue.
```

Во примерот секоја функција има јасно определена задача. Едната ја пресметува плоштината, а другата периметарот.

Во практичните програмски решенија често не е доволно да биде употребена само една функција. Поголемите проблеми најчесто се составени од повеќе помали задачи што треба да бидат решени независно една од друга. Поради тоа, при анализата на проблемот најпрво треба да биде утврдено кои активности ќе бидат извршувани, а потоа секоја од нив да биде реализирана во посебна функција. На овој начин програмата добива јасна структура, а секоја функција добива конкретна одговорност. Кога програмскиот код е организиран на ваков начин, многу полесно се откриваат грешки и се додаваат нови можности. Дополнително, функциите може повторно да бидат искористени и во други програми. Токму

затоа анализата на проблемот и правилната поделба на задачите претставуваат важен чекор во развојот на секое програмско решение.

Пример 3: Истражувачка станица на Марс

Во истражувачка станица на Марс секој ден се регистрира количината на произведена електрична енергија од соларните панели и количината на потрошена енергија од системите за одржување на животот. Да се пресмета дали е остварен енергетски вишок или дефицит.

```

1  #include <iostream>
2  using namespace std;
3  double proizvedenaEnergija(double paneli, double energijaPoPanel)
4  {
5      return paneli * energijaPoPanel;
6  }
7  double bilans(double proizvedeno, double potroseno)
8  {
9      return proizvedeno - potroseno;
10 }
11 int main()
12 {
13     double paneli;
14     double energijaPoPanel;
15     double potroseno;
16     cout << "Broj na paneli: ";
17     cin >> paneli;
18     cout << "Energija od eden panel: ";
19     cin >> energijaPoPanel;
20     cout << "Potrosena energija: ";
21     cin >> potroseno;
22     double proizvedeno=proizvedenaEnergija(paneli, energijaPoPanel);
23     double rezultat=bilans(proizvedeno, potroseno);
24     cout << "Energetski bilans: "
25          << rezultat
26          << endl;
27     return 0;
28 }
29

```

Output window (D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe):

```

Broj na paneli: 25
Energija od eden panel: 22
Potrosena energija: 473
Energetski bilans: 77
Press any key to continue.

```

Process returned 0 (0x0) execution time : 12.698 s

Во примерот проблемот е поделен на две независни задачи. Во првата функција се пресметува вкупната произведена енергија, а во втората се пресметува енергетскиот биланс. Благодарение на ваквата организација, секоја функција има јасно определена намена и програмата е полесна за разбирање.

Функциите претставуваат важна алатка за развој на квалитетни програмски решенија. Со нивна примена сложените проблеми може да бидат поделени на повеќе помали и полесно управливи целини.

На тој начин се овозможува подобра организација на програмскиот код, полесно тестирање на поединечните делови од решението и поефикасно откривање грешки. При анализа на текстуална задача секогаш треба да биде утврдено кои активности може да бидат издвоени во функции и кои податоци треба да бидат предавани преку параметри. Ваквиот пристап претставува основа за развој на поголеми програмски проекти и современи софтверски системи. Поради тоа, функциите се вбројуваат меѓу најважните концепти во програмскиот јазик C++ и претставуваат основа за понатамошно изучување на структурите за складирање и обработка на податоци.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Зошто е корисно сложените проблеми да се поделат на повеќе функции?
2. Кога е оправдано да се креира нова функција?
3. Која е улогата на параметрите при решавање текстуални задачи?
4. Која е предноста на функциите што враќаат вредност?
5. Како придонесуваат функциите за подобра организација на програмскиот код?



ЗАДАЧИ

1. Во зоолошка градина се води евиденција за дневна потрошувачка на храна. Напиши програма во која ќе бидат употребени две функции: едната да ја пресметува храната потребна за лавовите, а другата – храната потребна за тигрите.
2. Во аеродром се следи количината на гориво во авионите. Напиши програма во која една функција ќе ја пресметува вкупната количина натоварено гориво, а друга ќе го пресметува преостанатото гориво по летот.
3. Во ботаничка градина се засадуваат кружни цветни површини. Напиши програма во која една функција ќе ја пресметува плоштината, а друга – цената за уредување ако цената по квадратен метар се внесува од тастатура.
4. Во подводна истражувачка база се следи потрошувачката на кислород. Напиши програма во која ќе биде употребена функција што ја пресметува количината преостанат кислород по одреден број часови работа.
5. Во астрономска опсерваторија се пресметува времето потребно за сигнал да стигне од сателит до Земјата. Напиши програма во која ќе биде употребена функција што ја пресметува оваа вредност врз основа на внесено растојание.



ЗАПОМНИ ШТО НАУЧИ

- Текстуалната задача најпрво треба да биде анализирана.
- Сложените проблеми може да бидат поделени на повеќе помали задачи.
- Секоја задача може да биде реализирана во посебна функција.
- Функциите овозможуваат подобра организација и повторна употреба на кодот.
- Параметрите и повратните вредности овозможуваат размена на податоци меѓу функциите и програмата.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Современите компјутерски игри, мобилни апликации и оперативни системи содржат милиони линии програмски код. Таквите системи не би можеле да бидат развиени кога целиот код би бил сместен во една единствена функција. Поради тоа, програмерите ги организираат програмите во голем број функции и модули, при што секоја функција решава мал дел од вкупниот проблем. На овој начин се овозможува повеќе програмери истовремено да работат на ист проект, а програмите стануваат полесни за одржување и надградување. Со тоа завршува темата за функции и се создава основа за изучување на структури што овозможуваат обработка на поголем број податоци, како што се еднодимензионалните низи.





2.13

ЗАДАЧИ ЗА ВЕЖБАЊЕ ОД ФУНКЦИИ

ЗАДАЧИ ЗА ФУНКЦИИ ВО C++

Лесни задачи

1. Во метеоролошка станица секојдневно се следи температурата на воздухот. Напиши програма во која ќе вметнеш функција што прима температура во Целзиусови степени и ја пресметува соодветната температура во Фаренхајтови степени.
2. Во градежна компанија се изработуваат квадратни бетонски плочи. Напиши програма во која ќе вметнеш функција што прима должина на страна на плочата и го пресметува нејзиниот периметар.
3. Туристичка агенција организира патување што трае повеќе денови. Напиши програма во која ќе вметнеш функција што прима број на денови и пресметува колку цели недели содржи тој период.
4. Во продавница се организира сезонска распродажба. Напиши програма во која ќе вметнеш функција што прима цена на производ и процент на попуст и ја пресметува новата цена.
5. Во здравствена анкета се внесува возраста на испитаниците. Напиши програма во која ќе вметнеш функција што прима возраст и пресметува уште колку години недостасуваат до 100 години.
6. Во парк се планира изградба на кружна фонтана. Напиши програма во која ќе вметнеш функција што прима радиус на фонтаната и ја пресметува должината на нејзиниот раб.
7. Во спортска сала се води евиденција за времетраење на тренинзите. Напиши програма во која ќе вметнеш функција што прима број на минути и пресметува колку часови претставуваат.
8. Во лабораторија се споредуваат резултати од две мерења. Напиши програма во која ќе вметнеш функција што прима две вредности и ја пресметува нивната апсолутна разлика. Во функцијата да се користи локална променлива со име `razlika`.
9. На бензинска пумпа се врши анализа на количината на гориво. Напиши програма во која ќе вметнеш функција што прима количина гориво изразена во литри и ја пресметува истата количина во милилитри.
10. Во банка се анализира состојба на сметка. Напиши програма во која ќе вметнеш функција што прима број и утврдува дали е вредноста позитивна.

Средно тешки задачи

1. Во средновековен замок секој жител плаќа месечен данок од 12 монети. Напиши програма во која ќе вметнеш функција што прима број на жители и го пресметува вкупниот месечен данок.
2. Во работилница се проверуваат три дрвени летви за изработка на триаголна рамка. Напиши програма во која ќе вметнеш функција што прима три должини и проверува дали од нив може да се состави триаголник.
3. Во училиштен натпревар се доделува специјална ознака за броеви што се деливи и со 3 и со 5. Напиши програма во која ќе вметнеш функција што прима број и проверува дали го исполнува овој услов.
4. На мапа се означува положба на објект со координати x и y . Напиши програма во која ќе вметнеш функција што ги прима координатите и го пресметува растојанието на објектот од координатниот почеток.
5. Во домаќинство се пресметува сметка за електрична енергија. Напиши програма во која ќе има глобална променлива $\text{senaKwh} = 7$ и функција што прима потрошени киловат-часови и го пресметува износот за плаќање.
6. Во планинарски дом едно ноќевање чини 850 денари. Напиши програма во која ќе има глобална променлива $\text{senaNokevanje} = 850$ и функција што прима број на ноќевања и го пресметува вкупниот износ.
7. Во информатички систем се проверува должината на внесен идентификациски број. Напиши програма во која ќе вметнеш функција што прима природен број и пресметува колку цифри содржи.
8. Во спортска дисциплина се споредуваат резултатите од тројца натпреварувачи. Напиши програма во која ќе вметнеш функција што прима три резултати и го враќа најголемиот. Во функцијата да се користи локална променлива najgolem .
9. Во училишна лабораторија се пресметува синус на агол зададен во степени. Напиши програма во која ќе вметнеш функција што прима агол во степени, го претвора во радијани и го пресметува неговиот синус со користење на библиотеката cmath .
10. Во финансиска апликација се обработуваат реални вредности што треба да се заокружат. Напиши програма во која ќе вметнеш функција што прима реален број и ги прикажува резултатите од $\text{ceil}()$ и $\text{floor}()$.

Тешки задачи

1. Во музеј се продаваат билети за возрасни и за деца. Напиши програма во која ќе има глобални променливи $\text{senaVozrasen} = 200$ и $\text{senaDete} = 120$, како и функција што прима број на возрасни и број на деца и го пресметува вкупниот приход.
2. Во календарска апликација треба да се провери дали одредена година е престапна. Напиши програма во која ќе вметнеш функција што прима година и враќа дали е таа година престапна.
3. Во безбедносен систем се анализира број преку збирот на неговите цифри. Напиши програма во која ќе вметнеш функција што прима природен број и го пресметува збирот на неговите цифри.
4. Во математичка апликација се пресметуваат производи од последователни броеви. Напиши програма во која ќе вметнеш функција што прима природен број и го пресметува неговиот факториел.
5. Во видеоигра играчот отклучува ниво според бројот на собрани златници. Напиши програма во која ќе вметнеш функција што прима број на златници и одредува кое ниво е отклучено: до 100 златници – ниво 1, од 101 до 300 – ниво 2, а над 300 – ниво 3.

6. Во фабрика се пакуваат две серии производи во еднакви групи без остаток. Напиши програма во која ќе вметнеш функција што прима два броја и го пресметува нивниот најголем заеднички делител.
7. Во систем за мерење време се внесува времетраење во секунди. Напиши програма во која ќе вметнеш функција што прима број на секунди и ги прикажува часовите, минутите и секундите. Во функцијата да се користат локални променливи `casovi`, `minuti` и `sekundi`.
8. Во метеоролошка станица се мерат температурата и влажноста. Напиши програма во која ќе вметнеш функција што прима две вредности и враќа дали има можност за магла ако е температурата под 5°C , а влажноста е над 85%.
9. Во навигациска апликација се пресметува растојанието помеѓу две точки на мапа. Напиши програма во која ќе вметнеш функција што ги прима координатите на двете точки и го пресметува растојанието помеѓу нив.
10. Во астрономска опсерваторија се следи движење на комета. Напиши програма во која ќе вметнеш функција што прима изминато растојание и време и ја пресметува просечната брзина. Во функцијата да се користи локална променлива `brzina`.

2.14



ЕДНОДИМЕНЗИОНАЛНА НИЗА КАКО СТРУКТУРА НА ПОДАТОЦИ

ВЕЌЕ ЗНАЕШ ДА...

- користиш променливи од различен тип;
- внесуваш и да прикажуваш податоци;
- користиш условни конструкции;
- користиш циклуси;
- дефинираш и да повикуваш функции.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува функција?
2. Која е улогата на параметрите?
3. За што се користи наредбата return?
4. Зошто е корисно една програма да биде поделена на повеќе функции?
5. Наведи една функција од библиотеката smath.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што претставува низа;
- препознаеш ситуации во кои е потребна низа;
- разликуваш променлива од низа;
- објасниш како се чуваат елементите на низата во меморијата;
- користиш еднодимензионални низи во едноставни програмски решенија.

При развој на програмски решенија често се јавува потреба од обработка на голем број податоци од ист вид. На пример, може да биде потребно да се зачуваат оценките на учениците во едно одделение, дневните температури во текот на еден месец или резултатите на натпреварувачите на спортски турнир. Доколку за секој податок се користи посебна променлива, програмата брзо станува сложена и тешка за одржување.

За поефикасна организација на податоците во програмскиот јазик C++ се користат низи. Низата овозможува повеќе податоци од ист тип да бидат зачувани под едно заедничко име. Благодарение на тоа, податоците може полесно да бидат внесувани, обработувани и анализирани. Поради оваа причина, низите претставуваат една од најважните структури за складирање податоци во програмирањето.

Променливата овозможува зачувување на една вредност во меморијата на компјутерот. Кога е потребно да бидат зачувани десет, дваесет или сто вредности од ист вид, користењето голем број поединечни променливи не претставува практично решение. Во такви ситуации се користи низа. Низата претставува збир од елементи што имаат ист тип на податок и се зачувани еден до друг во меморијата. Сите елементи се поврзани со едно заедничко име, а секој елемент има своја позиција. Благодарение на оваа организација, голем број податоци може да се обработуваат како една целина. Низите се користат во речиси сите современи програмски решенија, затоа што овозможуваат ефикасна обработка на податоци и претставуваат основа за многу посложени структури во програмирањето.

Во програмскиот јазик C++ низата се декларира со наведување на типот на податокот, името на низата и бројот на елементи што ќе ги содржи. Општата форма е:

`тип имеНаНиза[големина];`

На пример: `int ocenki[5];`

Во овој пример клучниот збор `int` покажува дека во низата ќе бидат зачувани цели броеви. Името на низата е `ocenki`, а бројот 5 означува дека низата содржи пет елементи. Иако низата има пет елементи, нивните позиции не започнуваат од 1, туку од 0. Поради тоа, првиот елемент се наоѓа на позиција 0, вториот на позиција 1, третиот на позиција 2, четвртиот на позиција 3 и петтиот на позиција 4. Општо правило е дека кај низа со големина `n`, првиот елемент се наоѓа на позиција 0, а последниот на позиција `n - 1`.

За низата:

`int ocenki[5];`

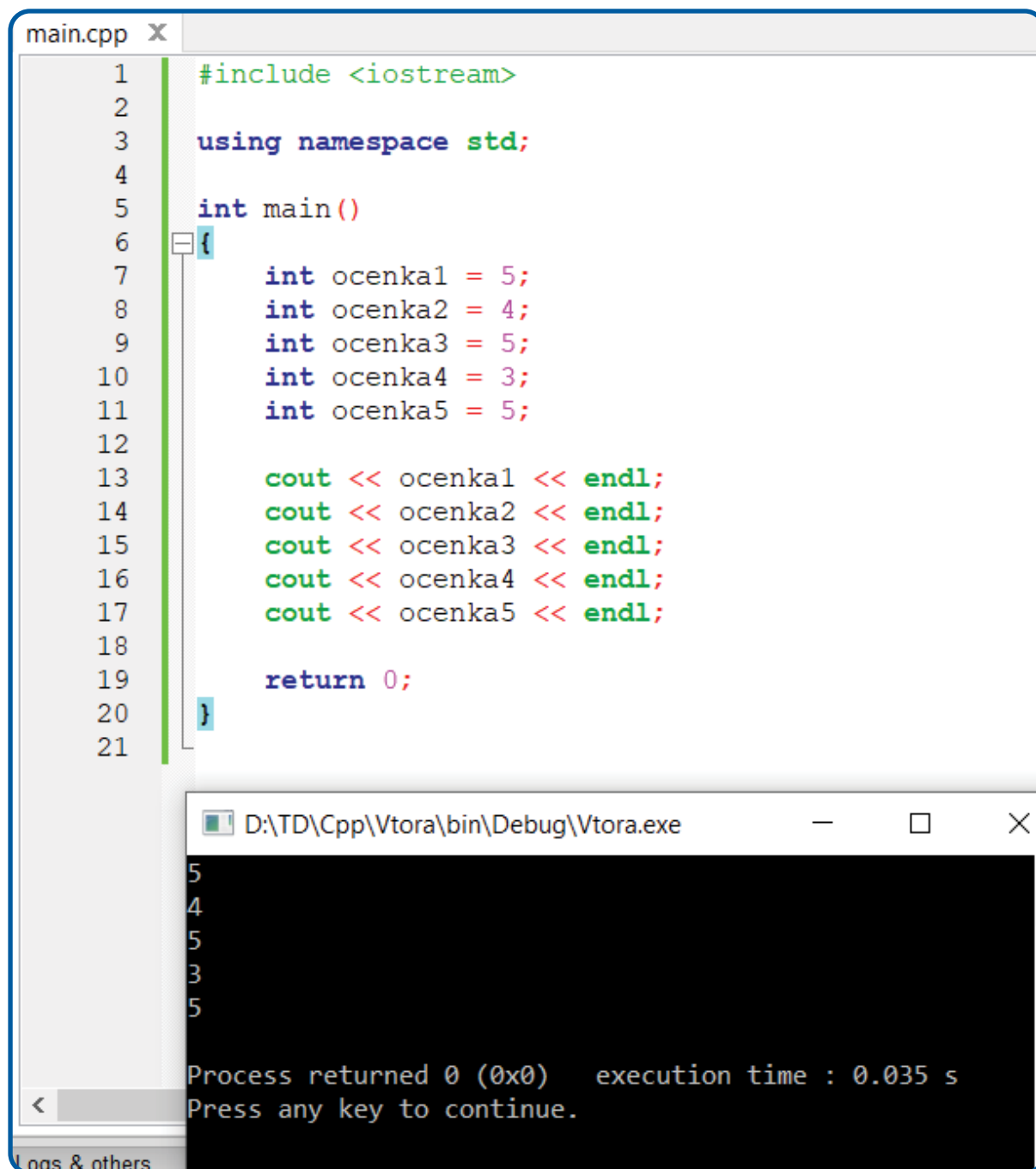
позициите на елементите се:

`ocenki[0] ocenki[1] ocenki[2] ocenki[3] ocenki[4]`

Поради тоа, при пристапување до елементите секогаш треба да се внимава да не се користи позиција поголема од последната дозволена вредност, бидејќи тоа може да предизвика неправилна работа на програмата.

Пример 1: Проблем без користење низа

Да се прикажат оценките на пет ученици.



```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int oценка1 = 5;
8      int oценка2 = 4;
9      int oценка3 = 5;
10     int oценка4 = 3;
11     int oценка5 = 5;
12
13     cout << oценка1 << endl;
14     cout << oценка2 << endl;
15     cout << oценка3 << endl;
16     cout << oценка4 << endl;
17     cout << oценка5 << endl;
18
19     return 0;
20 }
21
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

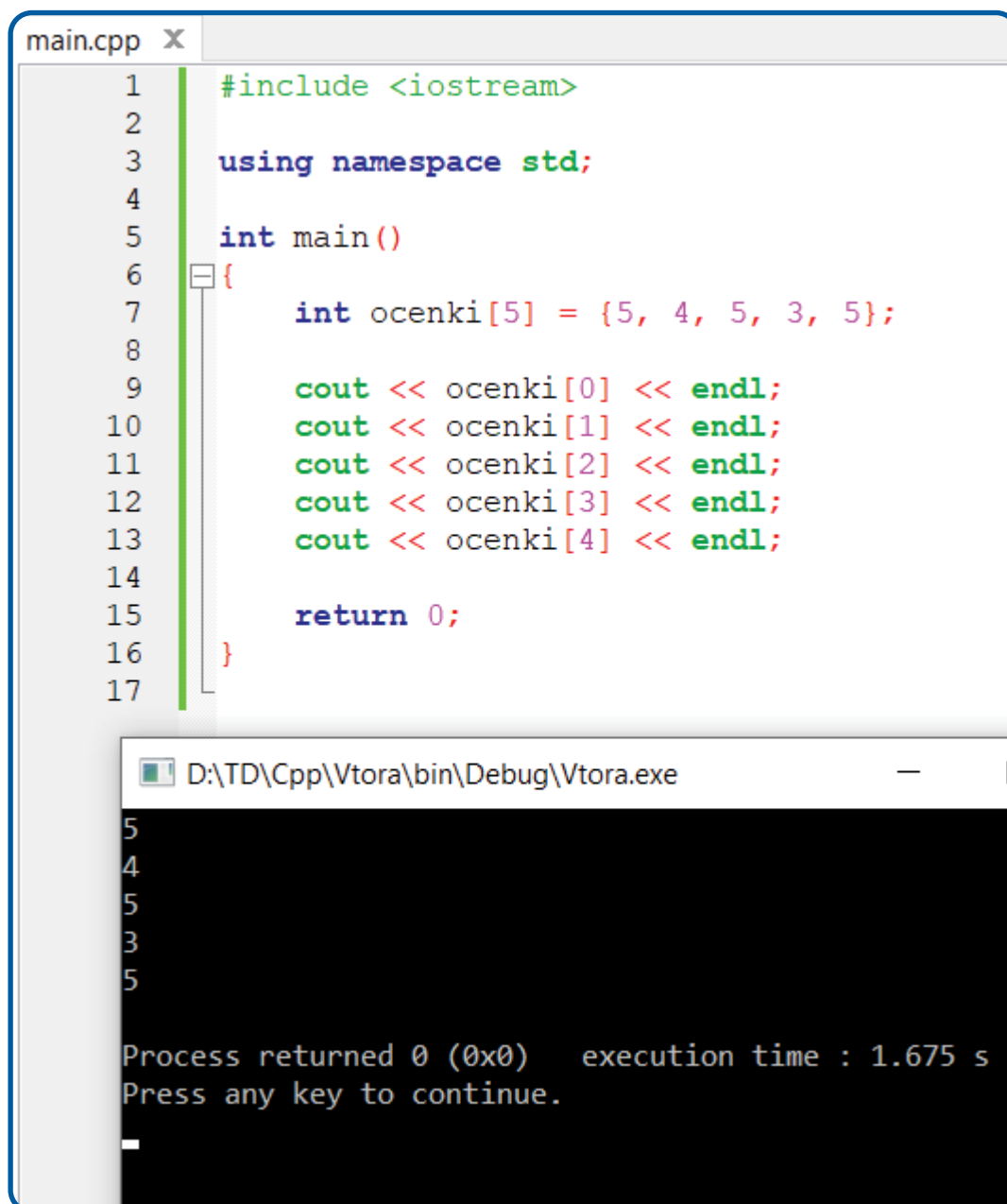
5
4
5
3
5

Process returned 0 (0x0) execution time : 0.035 s
Press any key to continue.

Во примерот за секоја оценка е употребена посебна променлива. Решението е прифатливо кога бројот на податоци е мал, но станува непрактично ако треба да се обработуваат десетици или стотици вредности.

Кога повеќе податоци имаат ист тип и иста намена, тие може да бидат организирани во низа. Наместо да се користат многу различни имиња на променливи, сите вредности се групираат под едно заедничко име. На тој начин програмата станува пократка и поуредна. Дополнително, податоците може лесно да се обработуваат со помош на циклуси. Токму оваа можност претставува една од најголемите предности на низите и причина за нивната широка примена во програмирањето.

Пример 2: Истите податоци организирани во низа



```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int ocenki[5] = {5, 4, 5, 3, 5};
8
9      cout << ocenki[0] << endl;
10     cout << ocenki[1] << endl;
11     cout << ocenki[2] << endl;
12     cout << ocenki[3] << endl;
13     cout << ocenki[4] << endl;
14
15     return 0;
16 }
17
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

```
5
4
5
3
5

Process returned 0 (0x0)   execution time : 1.675 s
Press any key to continue.
_
```

Во примерот сите оценки се зачувани во една низа со име `ocenki`. На тој начин повеќе вредности се организирани под едно заедничко име, што ја прави програмата поедноставна и полесна за одржување.

Елементите на една низа се сместуваат во последователни мемориски локации. Тоа значи дека компјутерот ги чува еден до друг во меморијата. Благодарение на оваа организација, пристапот до елементите е многу брз и ефикасен. Иако сите елементи се поврзани со едно заедничко име, секој од нив претставува посебна вредност што може независно да биде прочитана или изменета. Ваквиот начин на организирање на податоците овозможува голем број вредности да бидат обработувани како една целина. Токму поради оваа причина низите имаат значајна улога во програмирањето и претставуваат основа за голем број посложени структури на податоци. При работа со низи секогаш треба да се има предвид дека сите елементи мора да бидат од ист тип. Не е дозволено во иста низа истовремено да се чуваат цели броеви, реални броеви и текстуални вредности.

Пример 3: Број на посетители во музеј

Во текот на пет дена бил евидентиран бројот на посетители во еден музеј. Податоците треба да бидат зачувани во низа и прикажани на екранот.

```
#include <iostream>

using namespace std;

int main()
{
    int posetiteli[5] = {120, 145, 132, 160, 151};

    cout << "Den 1: " << posetiteli[0] << endl;
    cout << "Den 2: " << posetiteli[1] << endl;
    cout << "Den 3: " << posetiteli[2] << endl;
    cout << "Den 4: " << posetiteli[3] << endl;
    cout << "Den 5: " << posetiteli[4] << endl;

    return 0;
}
```

Во примерот податоците за сите пет дена се сместени во една низа. Наместо да се користат пет различни променливи, сите вредности се организирани под едно заедничко име. На овој начин програмата станува поуредна и полесна за одржување.

Низите се користат секогаш кога треба да биде обработен поголем број податоци од ист вид. Тие може да се применат за чување оценки, температури, финансиски податоци, спортски резултати, мерења од сензори и многу други информации. Без нив, развојот на поголеми програмски решенија би бил значително потешок. Поради тоа, изучувањето на низите претставува важен чекор во развојот на програмерските вештини. Откако ќе биде совладан основниот концепт, може да се премине кон внесување, обработка и анализа на елементите во низите. Токму овие активности претставуваат основа за голем број алгоритми и програмски решенија што се користат во практиката.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува еднодимензионална низа?
2. Кога е оправдано користење низа наместо повеќе поединечни променливи?
3. Каков тип податоци може да содржи една низа?
4. Како се организирани елементите на низата во меморијата?
5. Кои се предностите на користењето низи?



ЗАДАЧИ

1. Во една библиотека треба да бидат зачувани броевите на позајмени книги во текот на седум дена. Напиши програма што ќе ги смести податоците во низа и ќе ги прикаже на екранот.
2. Во спортски клуб се евидентира бројот на освоени поени на пет натпревари. Напиши програма што ќе ги зачува податоците во низа и ќе ги прикаже.
3. Во метеоролошка станица се бележат шест дневни температури. Напиши програма што ќе ги зачува температурите во низа и ќе ги испечати.
4. Во продавница се следи продажбата на четири производи. Напиши програма што ќе ги зачува количините во низа и ќе ги прикаже на екранот.
5. Во училиште се води евиденција за бројот на ученици во пет паралелки. Напиши програма што ќе ги зачува податоците во низа и ќе ги испечати.



ЗАПОМНИ ШТО НАУЧИ

- Низата претставува збир од елементи од ист тип.
- Сите елементи на низата имаат заедничко име.
- Елементите се сместени во последователни мемориски локации.
- Низите овозможуваат организирано чување на поголем број податоци.
- Низите претставуваат една од најважните структури на податоци во програмирањето.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите компјутерски системи огромни количини податоци се организираат со користење структури што се базираат на низи. Листите на корисници во социјалните мрежи, резултатите од пребарување на Интернет, податоците од научни експерименти и статистичките анализи често започнуваат токму со едноставни низи. Иако на прв поглед изгледаат како едноставна структура, низите претставуваат темел врз кој се изградени многу посложени структури што се користат во современото програмирање и во развојот на софтверски системи.

2.15



ДЕКЛАРИРАЊЕ И ИНИЦИЈАЛИЗАЦИЈА НА ЕДНОДИМЕНЗИОНАЛНИ НИЗИ

ВЕЌЕ ЗНАЕШ ДА...

- објасниш што претставува еднодимензионална низа;
- препознаеш ситуации во кои е потребно користење низа;
- разликуваш променлива од низа;
- објасниш што претставува големина на низа;
- препознаеш дека елементите на низата се нумерираат почнувајќи од 0.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува еднодимензионална низа?
2. Зошто е подобро да се користи низа наместо голем број поединечни променливи?
3. Што означува бројот во заградите при декларација на низа?
4. Која е позицијата на првиот елемент во низа?
5. Која е позицијата на последниот елемент во низа со големина 8?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

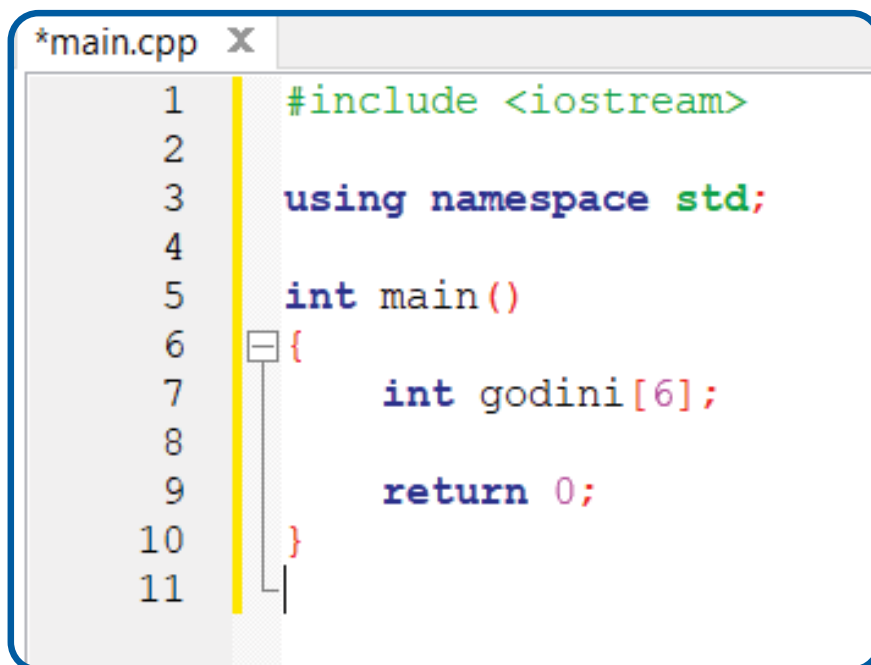
- декларираш еднодимензионални низи;
- иницијализираш низи при декларација;
- разликуваш декларација од иницијализација;
- користиш различни начини на иницијализирање на низи;
- препознаваш најчести грешки при работа со низи.

Пред да бидат податоците сместени во низа, потребно е најпрво да биде создаден простор во меморијата на компјутерот. Оваа постапка се нарекува декларирање на низа. При декларацијата се определуваат типот на податоци што ќе бидат зачувани и бројот на елементи што ќе ги содржи низата.

По декларацијата, елементите на низата може да добијат почетни вредности. Оваа постапка се нарекува иницијализација. Во програмскиот јазик C++ постојат повеќе начини за иницијализирање низи, а нивниот правилен избор придонесува за подобра прегледност и поедноставна работа со податоците.

Декларирањето низа претставува постапка со која се резервира простор во меморијата за определен број елементи од ист тип. При декларацијата задолжително се наведуваат типот на податокот, името на низата и нејзината големина. Големината определува колку елементи ќе може да содржи низата. Откако ќе биде декларирана, низата може да биде користена за зачувување и обработка на податоци. Доколку при декларацијата не им бидат доделени вредности на елементите, тие ќе содржат неопределени вредности. Поради тоа, во многу ситуации е препорачливо низата веднаш да биде иницијализирана. На овој начин програмата станува појасна, а можноста за појава на грешки значително се намалува.

Пример 1: Декларирање низа



```
*main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int godini[6];
8
9      return 0;
10 }
11
```

Во примерот е декларирана низа со име `godini`. Во неа може да бидат сместени шест цели броеви. Во овој момент е резервиран простор во меморијата, но на елементите сè уште не им се доделени вредности.

Кога се вредностите познати однапред, тие може да бидат доделени уште при декларацијата на низата. Ваквата постапка се нарекува иницијализација. При иницијализацијата вредностите се запишуваат во виткасти загради и се раздвојуваат со запирки. Бројот на вредностите најчесто е еднаков на големината на низата. Ваквиот начин на работа често се користи при тестирање на програми или кога подато-

ците се однапред познати. Иницијализираните низи овозможуваат програмата веднаш да располага со податоци што може да бидат обработувани и анализирани.

Пример 2: Иницијализација на низа

```

main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int temperaturi[7] =
8      {
9          18, 20, 21, 19, 22, 24, 23
10     };
11
12     cout << temperaturi[0] << endl;
13
14     return 0;
15 }
16
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
18

Process returned 0 (0x0)   execution time : 0.008 s
Press any key to continue.
  
```

Во примерот низата е иницијализирана уште при декларацијата. На секој елемент му е доделена почетна вредност и низата веднаш е подготвена за употреба.

При иницијализација на низи не е задолжително да бидат зададени вредности за сите елементи. Ако се наведат помалку вредности од големината на низата, првите елементи ќе ги добијат зададените вредности, а останатите ќе бидат поставени на нула. Овој начин на иницијализација се нарекува делумна иницијализација. Ваквиот пристап може да биде корисен кога само дел од податоците се познати во моментот на создавање на низата. Сепак, при работа со вакви низи треба внимателно да биде проверено кои елементи содржат реални податоци, а кои имаат автоматски доделена вредност. Познавањето на начинот на иницијализација е важно бидејќи директно влијае врз резултатите што ќе бидат добиени по извршување на програмата.

```

main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int poeni[6] =
8      {
9          12, 15, 18
10     };
11
12     cout << poeni[0] << endl;
13     cout << poeni[1] << endl;
14     cout << poeni[2] << endl;
15     cout << poeni[3] << endl;
16     cout << poeni[4] << endl;
17     cout << poeni[5] << endl;
18
19     return 0;
20
21

```

```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
12
15
18
0
0
0

Process returned 0 (0x0)   execution time : 0.476 s
Press any key to continue.

```

Во примерот првите три елементи добиваат вредности 12, 15 и 18. Останатите три елементи автоматски добиваат вредност 0. Благодарение на оваа можност, не е потребно секогаш да бидат наведени вредности за сите елементи од низата.

Во програмскиот јазик C++ постои можност големината на низата автоматски да биде определена врз основа на бројот на внесени вредности. Во такви ситуации не се наведува бројот на елементи во квадратните загради. Компилаторот самостојно го пресметува бројот на елементи според бројот на зададени вредности. Ваквиот пристап ја подобрува читливоста на програмата и ја намалува можноста за грешки при броење на елементите. Сепак, овој начин на работа е погоден само кога се познати сите вредности уште при декларацијата на низата. Ако се внесуваат податоците подоцна, големината мора однапред да биде определена. Затоа изборот на начинот на декларација зависи од конкретниот проблем што се решава и од достапноста на податоците во моментот на креирање на низата.

Автоматско определување на големина

```

int meseci[] =
{
    31, 28, 31, 30,
    31, 30, 31, 31,
    30, 31, 30, 31
};

```

Во овој пример не е наведена големината на низата. Бидејќи се внесени дванаесет вредности, компајлерот автоматски создава низа со дванаесет елементи.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува декларација на низа?
2. Што претставува иницијализација на низа?
3. Која е разликата помеѓу декларација и иницијализација?
4. Што се случува со елементите што не добиле вредност при делумна иницијализација?
5. Во кои ситуации може да се користи автоматско определување на големината на низата?



ЗАДАЧИ

1. Напиши програма во која ќе биде декларирана низа со десет цели броеви без иницијализација.
2. Напиши програма во која ќе биде иницијализирана низа со седум температури измерени во текот на една седмица.
3. Напиши програма во која ќе биде креирана низа со пет вредности што ги претставуваат резултатите на еден спортист на пет натпревари.
4. Напиши програма во која ќе биде употребена делумно иницијализирана низа со осум елементи, при што ќе бидат зададени само првите четири вредности.
5. Напиши програма во која големината на низата ќе биде автоматски определена врз основа на бројот на внесени вредности.



ЗАПОМНИ ШТО НАУЧИ

- Декларацијата резервира простор во меморијата за елементите на низата.
- Иницијализацијата им доделува почетни вредности на елементите.
- Големината на низата го определува бројот на елементи што може да бидат зачувани.
- При делумна иницијализација преостанатите елементи добиваат вредност 0.
- Големината на низата може автоматски да биде определена кога се наведени сите вредности при декларацијата.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите програмски јазици постојат структури што можат автоматски да ја менуваат својата големина во текот на извршувањето на програмата. Таквите структури овозможуваат додавање и отстранување елементи без однапред да биде определен нивниот број. Сепак, класичните низи остануваат една од најбрзите и најефикасни структури за складирање податоци. Поради нивната едноставност и ефикасност, тие претставуваат основа за голем број посложени структури што се користат во современото програмирање.



2.16

ИНДЕКС НА ЕЛЕМЕНТ И ПРИСТАПУВАЊЕ ДО ЕЛЕМЕНТИ НА НИЗА

ВЕЌЕ ЗНАЕШ ДА...

- објасниш што претставува еднодимензионална низа;
- декларираш низа;
- иницијализираш низа;
- разликуваш декларација од иницијализација;
- препознаеш дека елементите на низата се нумерираат почнувајќи од 0.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува декларација на низа?
2. Што претставува иницијализација на низа?
3. Колку елементи содржи низата `int broevi[8];`?
4. Која е позицијата на последниот елемент во низа со големина 10?
5. Што се случува со останатите елементи при делумна иницијализација?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што претставува индекс на елемент;
- пристапуваш до поединечни елементи на низа;
- менуваш вредности на елементи во низа;
- препознаваш невалидни индекси;
- избегнуваш најчести грешки при работа со индекси.

Иако сите елементи на една низа се поврзани со заедничко име, секој елемент мора да може поединечно да биде пристапен и обработен. За таа цел се користат индекси. Индексот ја определува позицијата на елементот во низата и овозможува програмата да пристапи до конкретна вредност.

Правилното користење на индексите претставува основа за успешна работа со низи. Голем број програмски грешки се појавуваат токму поради неправилно користење на индексите. Поради тоа, потребно е внимателно да биде разбрано како функционира нумерирањето на елементите и кои вредности претставуваат дозволени индекси.

Секој елемент во низата има своја позиција што се нарекува индекс. Индексот овозможува да биде одреден конкретниот елемент со кој ќе се работи. Во програмскиот јазик C++ нумерирањето започнува од 0. Поради тоа, првиот елемент има индекс 0, вториот индекс 1, третиот индекс 2 и така натаму. Доколку низата содржи пет елементи, последниот елемент нема индекс 5, туку индекс 4. Ова правило важи за сите низи, без разлика на нивната големина. Пристапувањето до елементите се врши со наведување на името на низата и индексот во квадратни загради. На овој начин програмата точно знае со кој елемент треба да работи.

Пример 1: Пристапување до елемент од низа

```
#include <iostream>

using namespace std;

int main()
{
    int poeni[5] =
    {
        10, 15, 20, 25, 30
    };

    cout << poeni[2] << endl;

    return 0;
}
```

Во примерот се прикажува елементот со индекс 2. Бидејќи нумерирањето започнува од 0, тоа е третиот елемент во низата и неговата вредност е 20.

Покрај читање вредности, индексите овозможуваат и промена на елементите во низата. За да биде променета вредноста на одреден елемент, потребно е да биде наведен неговиот индекс и да му се додели нова вредност. На овој начин поединечните елементи може независно да бидат менувани без да влијаат врз останатите елементи во низата. Оваа можност е особено важна при обработка на податоци, затоа што овозможува ажурирање на конкретни информации без потреба од повторно креирање на целата низа.

Пример 2: Промена на вредност во низа

```
#include <iostream>

using namespace std;

int main()
{
    int zaliha[4] =
    {
        12, 18, 25, 30
    };

    zaliha[1] = 22;

    cout << zaliha[1] << endl;

    return 0;
}
```

Во примерот вредноста на елементот со индекс 1 се менува од 18 на 22.

При работа со низи особено важно е да се користат дозволени индекси. Секој елемент може да биде пристапен само преку индекс што припаѓа на низата. Доколку низата содржи пет елементи, дозволени индекси се 0, 1, 2, 3 и 4. Обидот за пристапување до елемент со индекс 5 или поголем – претставува грешка. Исто така, не е дозволено користење негативни индекси. Ваквите грешки може да предизвикаат неправилна работа на програмата или прикажување неочекувани резултати. Поради тоа, секогаш треба внимателно да биде проверено дали користениот индекс се наоѓа во дозволениот опсег. Разбирањето на ова правило претставува еден од најважните предуслови за успешна работа со низи.

Пример 3: Пристапување до повеќе елементи

Во спортски клуб се евидентирани поените освоени на пет натпревари. Да се прикажат првиот, третиот и последниот резултат.

```
#include <iostream>

using namespace std;

int main()
{
    int poeni[5] =
    {
        12, 18, 15, 20, 25
    };

    cout << "Prv rezultat: "
         << poeni[0]
         << endl;
```

```
cout << "Tret rezultat: "
    << poeni[2]
    << endl;

cout << "Posleden rezultat: "
    << poeni[4]
    << endl;

return 0;
}
```

Во примерот се пристапува до три различни елементи од истата низа. Првиот елемент има индекс 0, третиот индекс 2, а последниот индекс 4. Ова покажува дека позицијата на елементот не е секогаш еднаква на неговиот реден број во секојдневното броење.

Индексите претставуваат основен механизам преку кој се пристапува до елементите на низите. Без нив не би било можно поединечните вредности да бидат читани, менувани или обработувани. Поради тоа, при работа со низи секогаш треба да биде позната нивната големина и дозволеният опсег на индекси. Откако ќе биде совладано пристапувањето до поединечни елементи, може да се премине кон внесување и обработка на поголем број податоци, што претставува следен чекор во изучувањето на еднодимензионалните низи.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува индекс на елемент?
2. Од кој број започнува нумерирањето на елементите во низа?
3. Кој е индексот на последниот елемент во низа со големина 12?
4. Како се пристапува до елемент од низа?
5. Зошто е важно да се користат само дозволени индекси?



ЗАДАЧИ

1. Во една библиотека се евидентирани пет дневни посети, и тоа од учениците со број на членска карта: 35, 42, 38, 50 и 47. Напиши програма што ќе го прикажува првиот и последниот елемент од низата.
2. Во продавница се следи продажбата на шест производи. Напиши програма што ќе ги зачува податоците во низа и ќе го прикажува четвртиот елемент.
3. Во спортски турнир се запишани освоените поени на една екипа на пет натпревари. Напиши програма што ќе ја промени вредноста на вториот натпревар и ќе ја прикаже новата вредност.
4. Во метеоролошка станица се зачувани седум температури во низа. Напиши програма што ќе ги прикаже температурата од првиот, третиот и шестиот ден.
5. Во училишна евиденција се зачувани бројот на ученици во пет паралелки. Напиши програма што ќе ја промени вредноста на последниот елемент и ќе го прикаже резултатот.



ЗАПОМНИ ШТО НАУЧИ

- Индексот ја определува позицијата на елементот во низата.
- Нумерирањето на елементите започнува од 0.
- Последниот елемент има индекс еднаков на големината на низата намалена за 1.
- Индексите се користат за читање и измена на елементите.
- Треба да се користат само индекси што припаѓаат на низата.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во многу програмски јазици, како C++, Java и C#, нумерирањето на елементите во низите започнува од 0. Ова правило потекнува од начинот на кој компјутерот ги пресметува мемориските адреси на елементите. Иако на почетокот може да изгледа необично, ваквиот начин на нумерирање овозможува поедноставни и побрзи пресметки при пристапување до податоците. Токму затоа индексирањето од 0 е прифатено како стандард во голем број современи програмски јазици.

2.17



ВНЕСУВАЊЕ И ПЕЧАТЕЊЕ НА ЕЛЕМЕНТИ ОД НИЗА

ВЕЌЕ ЗНАЕШ ДА...

- декларираш еднодимензионална низа;
- иницијализираш низа;
- пристапуваш до елементи преку индекс;
- менуваш вредности на елементи во низа;
- користиш циклус `for`.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува индекс на елемент?
2. Од кој број започнува нумерирањето на елементите во низа?
3. Кој е индексот на последниот елемент во низа со големина 15?
4. Како се менува вредноста на еден елемент во низа?
5. Зошто не треба да се користат индекси надвор од дозволениот опсег?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- внесуваш вредности во низа;
- користиш циклус за внесување на елементи;
- прикажуваш елементи од низа;
- комбинираш низи и циклуси;
- избегнуваш најчести грешки при внесување податоци.

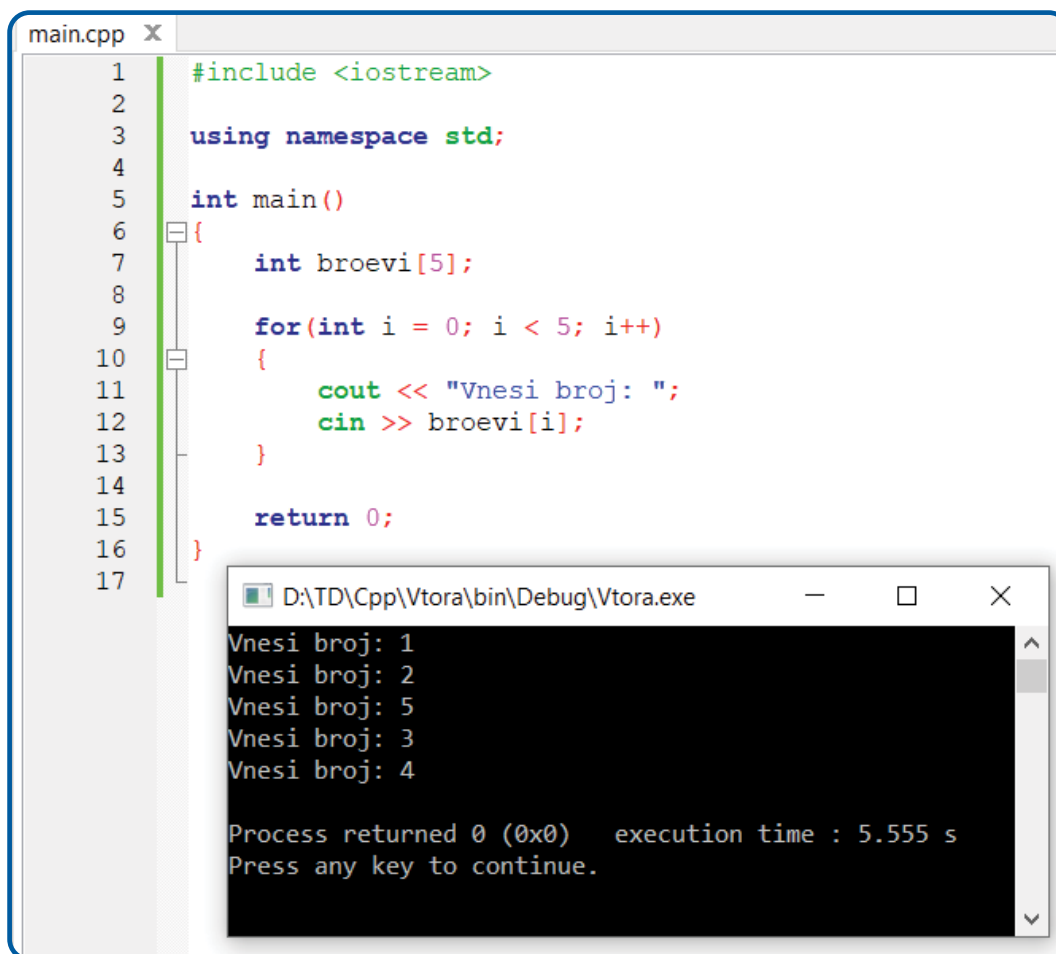
Една од најважните предности на низите е можноста за обработка на голем број податоци со мал број програмски инструкции. Доколку е потребно да бидат внесени десет, дваесет или сто вредности, не

е практично за секоја вредност да биде користена посебна променлива. Наместо тоа, податоците може да бидат сместени во низа.

При работа со низи многу често е потребно податоците да бидат внесени од корисникот. За таа цел се користат циклуси што овозможуваат внесување повеќе елементи со повторување на исти инструкции. На сличен начин може да биде извршено и прикажување на содржината на низата.

При внесување податоци во низа секој елемент треба да добие вредност. Наместо да се пишува посебна наредба за секој елемент, најчесто се користи циклус `for`. Бројачот на циклусот ја претставува позицијата на елементот што се обработува во моментот. На овој начин може лесно да биде извршено внесување на сите елементи, без разлика дали низата содржи пет, десет или сто вредности. Комбинацијата на низи и циклуси претставува една од најважните техники во програмирањето бидејќи овозможува ефикасна обработка на поголем број податоци.

Пример 1: Внесување на пет броја во низа



```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int broevi[5];
8
9      for(int i = 0; i < 5; i++)
10     {
11         cout << "Vnesi broj: ";
12         cin >> broevi[i];
13     }
14
15     return 0;
16 }
17
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

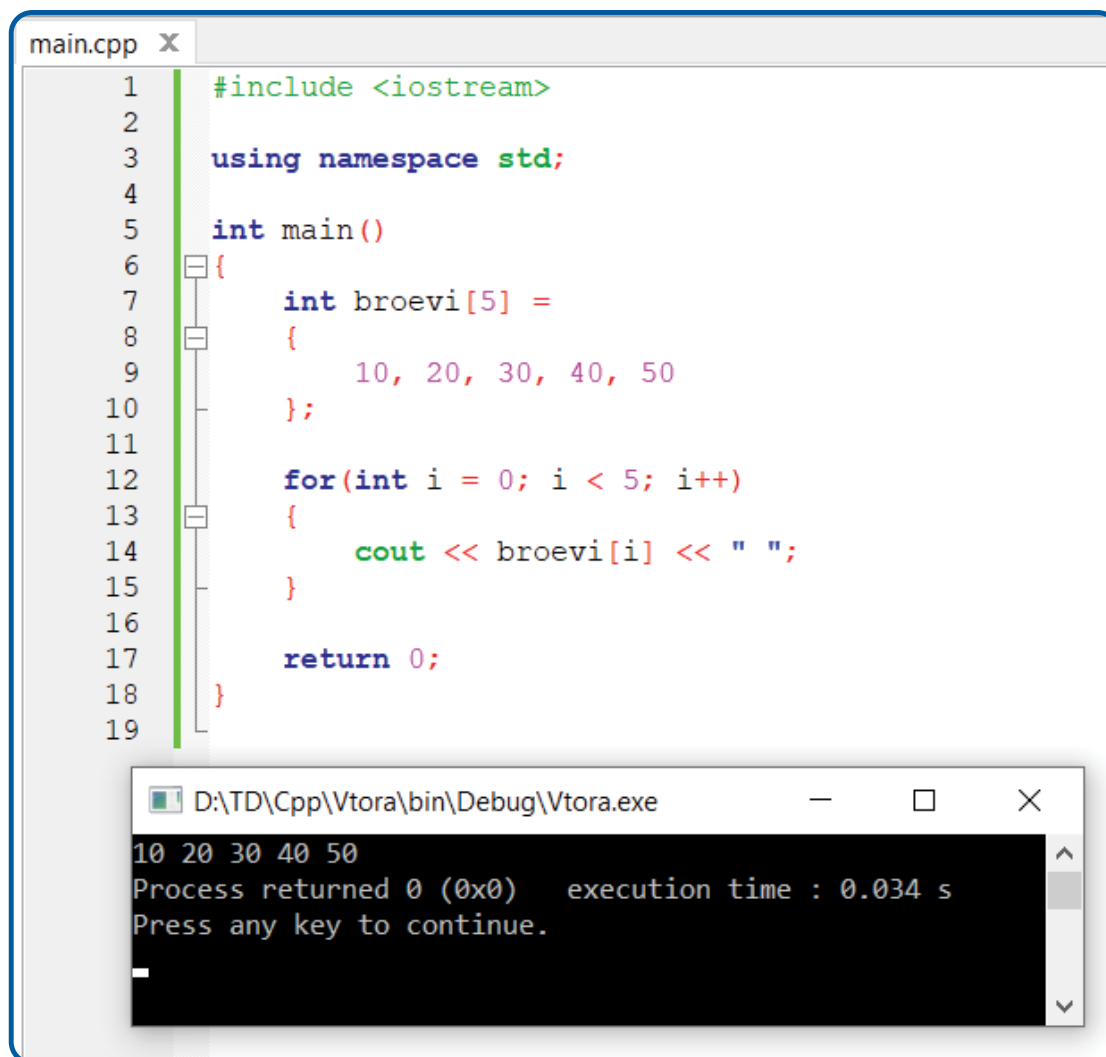
Vnesi broj: 1
Vnesi broj: 2
Vnesi broj: 5
Vnesi broj: 3
Vnesi broj: 4

Process returned 0 (0x0) execution time : 5.555 s
Press any key to continue.

Во примерот циклусот се извршува пет пати. При секое извршување се внесува вредност во различен елемент од низата.

Откако ќе бидат внесени податоците, често е потребно нивно прикажување. И во оваа ситуација се користи циклус `for`. Наместо да се пишува посебна наредба за секој елемент, циклусот последователно пристапува до сите позиции во низата. Овој пристап ја прави програмата пократка и поразбирлива. Дополнително, истиот код може да биде применет и кај многу поголеми низи без значителни промени.

Пример 2: Прикажување на елементите од низа



```

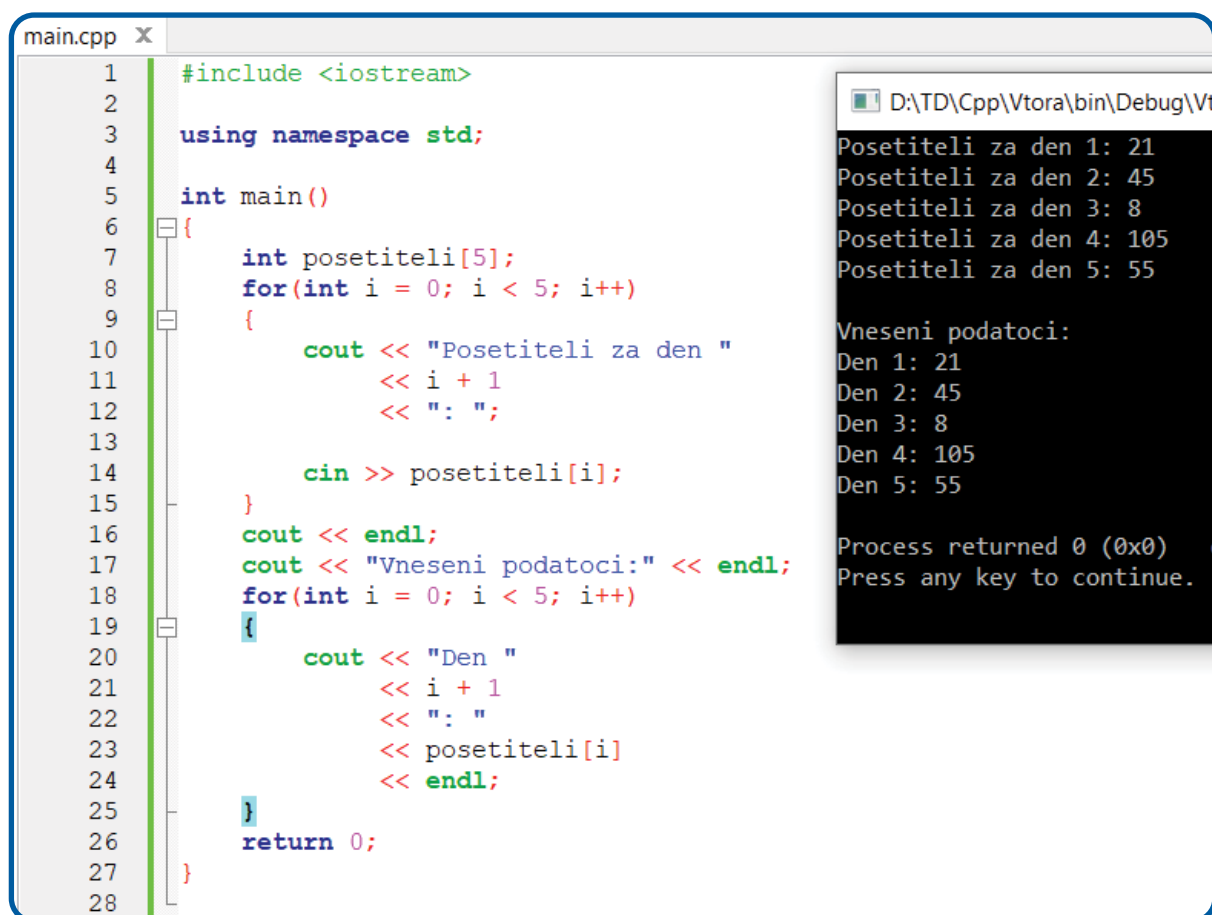
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int broevi[5] =
8      {
9          10, 20, 30, 40, 50
10     };
11
12     for(int i = 0; i < 5; i++)
13     {
14         cout << broevi[i] << " ";
15     }
16
17     return 0;
18 }
19
D:\TD\C++\Vtora\bin\Debug\Vtora.exe
10 20 30 40 50
Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
  
```

Во примерот сите елементи од низата се прикажуваат со помош на циклус. Без циклус би биле потребни пет посебни наредби `cout`.

Во практичните програмски решенија најчесто не е доволно само да бидат внесени податоците. По внесувањето обично следува нивна обработка, проверка или прикажување. Поради тоа, многу често во една програма се користат два циклуси. Првиот циклус служи за внесување на податоците во низата, а вториот за нивно прикажување. Ваквиот пристап овозможува сите податоци најпрво да бидат зачувани во меморијата, а потоа да бидат обработени на различни начини. Со зголемување на бројот на елементи, предностите на ваквата организација стануваат уште поизразени. Наместо да се пишуваат десетици или стотици наредби за внесување и прикажување, потребни се само два кратки циклуси. Токму затоа комбинацијата на низи и циклуси претставува една од најкористените техники во програмирањето.

Пример 3: Внесување и прикажување на број на посетители

Во музеј се евидентира бројот на посетители во текот на пет дена. Да се внесат податоците и потоа да се прикажат.



The screenshot shows a C++ IDE with a file named `main.cpp`. The code defines an array `posetiteli` of size 5 and uses two loops. The first loop prompts the user to enter the number of visitors for each day. The second loop prints the results. A console window on the right shows the program's execution, including the input values and the final output.

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int posetiteli[5];
8     for(int i = 0; i < 5; i++)
9     {
10         cout << "Posetiteli za den "
11             << i + 1
12             << ": ";
13
14         cin >> posetiteli[i];
15     }
16     cout << endl;
17     cout << "Vneseni podatoci:" << endl;
18     for(int i = 0; i < 5; i++)
19     {
20         cout << "Den "
21             << i + 1
22             << ": "
23             << posetiteli[i]
24             << endl;
25     }
26     return 0;
27 }
28
```

Output from the console window:

```
Posetiteli za den 1: 21
Posetiteli za den 2: 45
Posetiteli za den 3: 8
Posetiteli za den 4: 105
Posetiteli za den 5: 55

Vneseni podatoci:
Den 1: 21
Den 2: 45
Den 3: 8
Den 4: 105
Den 5: 55

Process returned 0 (0x0)
Press any key to continue.
```

Во примерот првиот циклус ги внесува податоците во низата, а вториот циклус ги прикажува. На овој начин сите вредности може лесно да се проверат по завршувањето на внесувањето.

При работа со низи и циклуси особено важно е правилно да биде определен бројот на повторувања. Доколку циклусот се извршува повеќе пати од бројот на елементи во низата, ќе се појави обид за пристапување до непостоечки елементи. Од друга страна, доколку циклусот се извршува помалку пати, дел од елементите нема да бидат обработени. Поради тоа, горната граница на циклусот треба да биде усогласена со големината на низата. Ова правило ќе биде применувано понатаму за да се вршат различни пресметки и анализи врз елементите на низите.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Зошто најчесто се користи циклус при внесување на елементи во низа?
2. Која е улогата на бројачот во циклусот?
3. Зошто е важно бројот на повторувања да одговара на големината на низата?
4. Кои се предностите на користење циклус за прикажување на елементи?
5. Зошто низите и циклусите често се користат заедно?



ЗАДАЧИ

1. Во археолошко наоѓалиште биле пронајдени седум предмети со различна старост изразена во години. Напиши програма што ќе ги внесе старостите во низа и потоа ќе ги прикаже по редоследот на внесување.
2. Во астрономска опсерваторија се бележи бројот на забележани метеори во текот на десет ноќи. Напиши програма што ќе ги внесе податоците во низа и потоа ќе ги испечати сите вредности.
3. Во фабрика за велосипеди се евидентира бројот на произведени велосипеди во текот на шест работни смени. Напиши програма што ќе ги внесе податоците во низа и ќе ги прикаже на екранот.
4. Во национален парк се следи бројот на посетители на пет различни туристички патеки. Напиши програма што ќе ги внесе податоците во низа и потоа ќе ги испечати со порака за секоја патека.
5. Во вселенска станица се мерат количините на вода (во литри) што се потрошени во текот на осум последователни дена. Напиши програма што ќе ги внесе вредностите во низа и ќе ги прикаже по завршувањето на внесувањето.



ЗАПОМНИ ШТО НАУЧИ

- Внесувањето на елементи во низа најчесто се врши со циклус `for`.
- Бројачот на циклусот се користи како индекс на елементите.
- Прикажувањето на елементите исто така може да се реализира со циклус.
- Големината на низата треба да биде усогласена со бројот на повторувања на циклусот.
- Комбинацијата на низи и циклуси овозможува ефикасна обработка на голем број податоци.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во голем број современи апликации податоците пристигнуваат од различни извори како што се сензори, интернет-услуги, бази на податоци или мобилни уреди. Без разлика на нивното потекло, тие најчесто се зачувуваат во структури што овозможуваат обработка на повеќе вредности истовремено. Низите претставуваат една од наједноставните такви структури. Благодарение на циклусите, илјадници податоци може да бидат обработени со релативно мал број програмски инструкции. Оваа можност претставува основа за развој на системи за статистичка анализа, научни истражувања, обработка на слики, машинско учење и многу други современи информатички области.



2.18

ОСНОВНИ ОПЕРАЦИИ СО ЕЛЕМЕНТИ НА НИЗА

ВЕЌЕ ЗНАЕШ ДА...

- декларираш и да иницијализираш низа;
- внесуваш и прикажуваш елементи од низа;
- пристапуваш до елементите преку индекс;
- користиш циклус `for`;
- комбинираш низи и циклуси.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува индекс на елемент?
2. Од кој број започнува нумерирањето на елементите?
3. Зошто при работа со низи се користи циклус `for`?
4. Како се внесуваат повеќе елементи во низа?
5. Што ќе се случи ако се употреби индекс надвор од границите на низата?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- пресметуваш сума на елементите од низа;
- пресметуваш производ на елементите од низа;
- користиш дополнителни променливи при обработка на податоци;
- комбинираш аритметички операции и низи;
- решаваш практични задачи со обработка на сите елементи.

Во голем број програмски решенија не е доволно податоците само да бидат внесени и прикажани. Често има потреба да биде пресметана нивната вкупна вредност, производот на сите вредности или некој друг заеднички резултат. На пример, може да биде потребно да се пресмета вкупната количина произведен мед, вкупниот број патници или вкупната должина на патување составено од повеќе делници.

За решавање на вакви проблеми најчесто се користи циклус што последователно ги обработува сите елементи на низата. При секое поминување низ циклусот се ажурира резултатот од пресметката.

На овој начин голем број податоци може лесно да бидат обработени со релативно мал број програмски инструкции.

При обработка на елементите од една низа често има потреба да биде пресметана нивната вкупна вредност. За таа цел се користи дополнителна променлива што на почетокот добива вредност 0. Потоа, при секое поминување низ циклусот, кон неа се додава тековниот елемент од низата. По обработката на последниот елемент, променливата ја содржи вкупната сума на сите вредности. Овој начин на работа има примена во голем број практични ситуации и претставува една од основните техники за обработка на податоци во низи.

Пример 1: Производство на мед

Во пчеларник се измерени количините мед произведени од пет кошници. Да се пресмета вкупното производство.

```

main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int med[5]={18,22,20,25,19};
8
9      int suma=0;
10
11     for(int i=0;i<5;i++)
12     {
13         suma=suma+med[i];
14     }
15     cout <<"Vкупno proizvodstvo: "<<suma<<" kg";
16     return 0;
17 }
18
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Vкупno proizvodstvo: 104 kg
Process returned 0 (0x0)   execution time : 0.164 s
Press any key to continue.

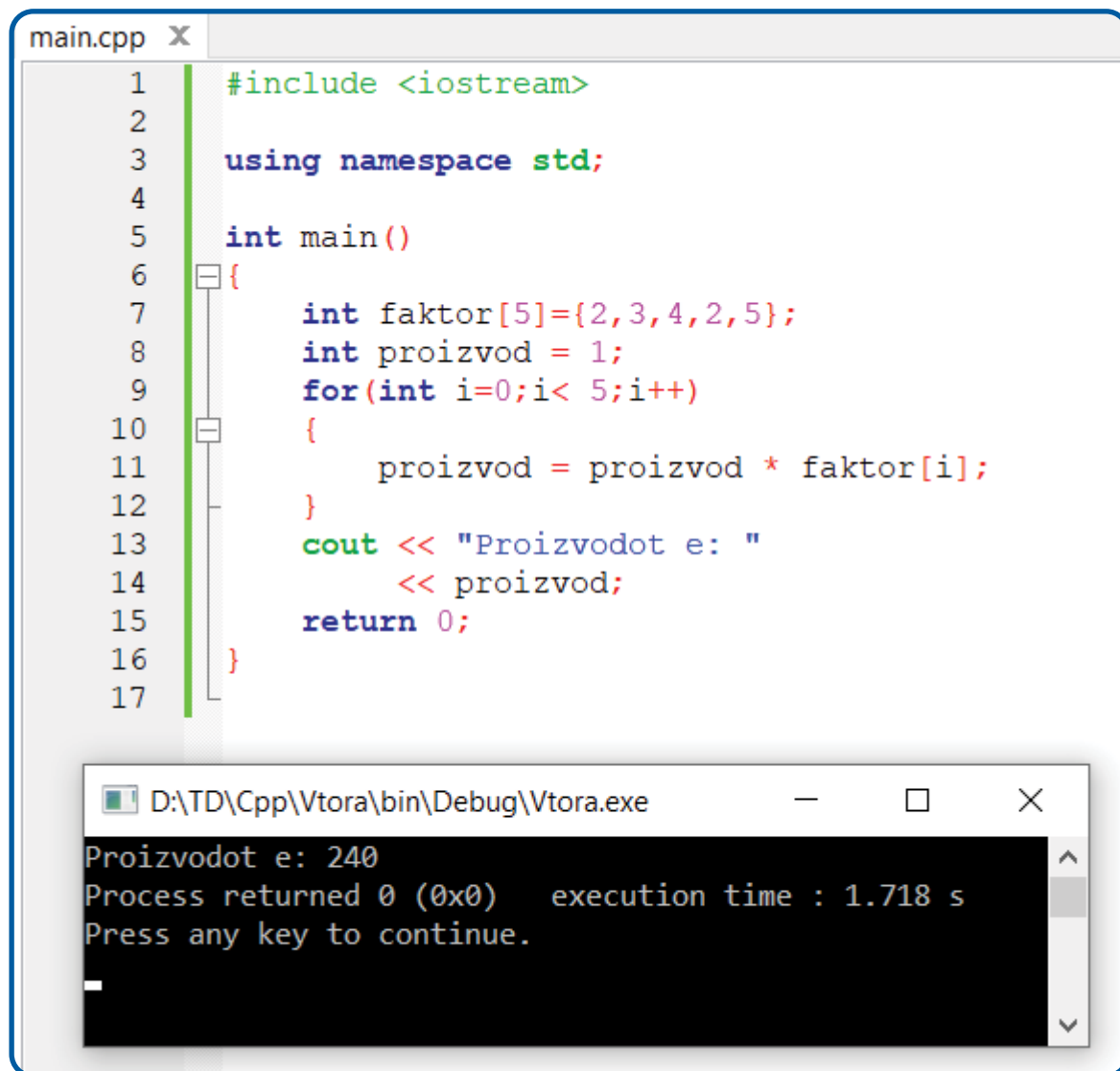
```

Во примерот сите количини произведен мед постепено се додаваат една на друга. По завршувањето на циклусот се добива вкупното производство.

Не секогаш има потреба да биде пресметана сума. Во одредени ситуации треба да биде пресметан производ на повеќе вредности. Во такви задачи се користи дополнителна променлива што на почетокот добива вредност 1. Причината за ова е што множењето со 1 не ја менува вредноста на броевите, додека множењето со 0 секогаш дава резултат 0. При секое поминување низ циклусот тековниот елемент се множи со претходниот резултат.

Пример 2: Вкупен коефициент на пренос

Во научен експеримент се измерени пет коефициенти на засилување. Да се пресмета нивниот производ.



The image shows a screenshot of a C++ IDE with a file named `main.cpp`. The code defines an array `faktor` with values {2, 3, 4, 2, 5} and calculates their product in a variable `proizvod`. The output window shows the result 240 and execution time 1.718 s.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int faktor[5]={2,3,4,2,5};
8      int proizvod = 1;
9      for(int i=0;i< 5;i++)
10     {
11         proizvod = proizvod * faktor[i];
12     }
13     cout << "Proizvodot e: "
14          << proizvod;
15     return 0;
16 }
17
```

Output window (D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe):

```
Proizvodot e: 240
Process returned 0 (0x0)   execution time : 1.718 s
Press any key to continue.
```

Во примерот променливата `proizvod` постепено се ажурира при обработката на секој елемент од низата. По завршувањето на циклусот се добива производот на сите вредности.

При обработка на податоците зачувани во една низа не секогаш има потреба да биде пресметана само сума или производ. Во голем број практични ситуации потребно е врз секој елемент да биде извршена одредена пресметка, а добиените резултати постепено да бидат комбинирани. Ваквиот начин

на работа овозможува решавање на разновидни проблеми од секојдневниот живот и од различни научни области. Без разлика на видот на пресметката, основната идеја останува иста. Циклусот последователно ги обработува сите елементи од низата, а врз секој елемент се применува потребната операција. На овој начин може да биде пресметана вкупна потрошувачка, произведена енергија, финансиски износ или многу други вредности.

Пример 3: Потрошувачка на вода

Во истражувачки камп се евидентира дневната потрошувачка на вода на пет експедиции. Цената на еден литар вода изнесува 2 денари. Да се пресмета вкупниот износ што треба да биде платен.

The image shows a code editor window titled 'main.cpp' with the following C++ code:

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int voda[5]={120,150,110,140,130};
8      int vkupno = 0;
9      for(int i=0;i<5;i++)
10     {
11         vkupno+=voda[i]*2;
12     }
13     cout << "Vкупен iznos: "<<vkupno<<" denari"<<endl;
14     return 0;
15 }
16

```

Below the code editor is a console window titled 'D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe' showing the output:

```

Vкупен iznos: 1300 denari

Process returned 0 (0x0)   execution time : 1.682 s
Press any key to continue.

```

Во примерот не се собираат само вредностите од низата. За секој елемент најпрво се пресметува неговата цена, а потоа таа вредност се додава кон вкупниот износ. На овој начин се комбинираат аритметички операции и обработка на елементите од низата.

При обработка на низи изборот на операцијата што ќе биде извршена зависи од проблемот што се решава. Во некои ситуации се пресметува сума, во други производ, а во трети одреден математички израз за секој елемент. Заедничка карактеристика на сите овие задачи е тоа што секој елемент од низата се обработува точно еднаш. Благодарение на ова, програмите остануваат едноставни и лесни за

разбирање, дури и кога работат со поголем број податоци. Ваквиот начин на обработка претставува основа за голем број алгоритми што ќе бидат изучувани во наредните лекции.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Зошто при обработка на низа најчесто се користи циклус `for`?
2. Со која почетна вредност најчесто започнува променливата за сума?
3. Зошто променливата за производ најчесто започнува со вредност 1?
4. Дали врз секој елемент од низата може да биде извршена различна аритметичка операција?
5. Која е предноста на обработката на сите елементи со еден циклус?



ЗАДАЧИ

1. Во ветерен парк се измерени количините произведена електрична енергија од шест ветерници. Напиши програма што ќе ја пресметува вкупната произведена енергија.
2. Во фабрика се произведуваат делови што се пакуваат во групи. Напиши програма што ќе го пресметува производот на бројот на групи зачувани во една низа.
3. Во еколошка акција се евидентира бројот на засадени дрвја во пет населени места. За секое дрво се издвојуваат по 50 денари за одржување. Напиши програма што ќе го пресметува вкупниот потребен износ.
4. Во роботска лабораторија се бележи бројот на успешно извршени задачи на четири работи. Напиши програма што ќе го пресметува вкупниот број успешно извршени задачи.
5. Во истражувачка станица се мери количината собрани примероци во текот на седум дена. За секој примерок се пресметуваат по 3 поени. Напиши програма што ќе го пресметува вкупниот број освоени поени.



ЗАПОМНИ ШТО НАУЧИ

- Елементите на една низа може последователно да бидат обработени со циклус `for`.
- За пресметување на вкупна вредност најчесто се користи променлива за сума.
- За пресметување производ се користи променлива што започнува со вредност 1.
- Врз секој елемент од низата може да биде применета аритметичка операција.
- Обработката на сите елементи со еден циклус овозможува едноставни и ефикасни програмски решенија.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите информатички системи често се обработуваат големи количини податоци. На пример, може да бидат анализирани мерења од сензори, финансиски трансакции, научни експерименти или статистички податоци. Иако таквите системи работат со илјадници или милиони вредности, основната идеја останува иста – податоците последователно се обработуваат, а врз секој елемент се извршуваат потребните операции. Познавањето на овие основни техники претставува важна основа за изучување на посложени алгоритми и структури на податоци.





2.19

ВНЕСУВАЊЕ НА ЕЛЕМЕНТИ ВО НИЗА И ОСНОВНИ ПРЕСМЕТКИ

ВЕЌЕ ЗНАЕШ ДА...

- декларираш и да иницијализираш еднодимензионална низа;
- пристапуваш до елементите на низата;
- користиш циклус for;
- пресметуваш сума и производ на однапред зададени елементи;
- обработуваш елементи на низа со циклус.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Како се декларира еднодимензионална низа?
2. Од кој број започнува нумерирањето на елементите?
3. Зошто при работа со низи најчесто се користи циклус for?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- внесуваш елементи во низа;
- обработуваш внесени податоци;
- пресметуваш сума на внесени вредности;
- пресметуваш производ на внесени вредности;
- пресметуваш аритметичка средина на внесени податоци.

Во голем број програмски решенија податоците не се познати однапред. Наместо да бидат внесени во програмата при нејзиното пишување, тие се внесуваат во текот на извршувањето. На овој начин една иста програма може да биде употребена за обработка на различни множества податоци.

При работа со низи најчесто податоците се внесуваат од тастатура со помош на циклус. Откако ќе бидат внесени, врз нив може да бидат извршени различни пресметки како што се определување на сума, производ или аритметичка средина. Ваквиот начин на работа претставува основа за голем број практични програмски задачи.

При внесување на елементите во една низа најчесто се користи циклус `for`. Бројачот на циклусот ја определува позицијата на елементот што во моментот се внесува. По внесувањето на сите вредности, тие може да бидат обработени со втор циклус или во текот на самото внесување. Благодарение на оваа можност, голем број податоци може брзо и ефикасно да бидат обработени. Ваквиот пристап има широка примена во различни информатички системи и претставува една од основните техники при работа со еднодимензионални низи.

Пример 1: Вкупна должина на велосипедски патеки

Во национален парк се внесуваат должините на пет велосипедски патеки. Да се пресмета нивната вкупна должина.

The image shows a C++ program in a code editor and its execution output in a console window.

Code Editor (main.cpp):

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int pateki[5];
8      int suma = 0;
9
10     for(int i = 0; i < 5; i++)
11     {
12         cout << "Dolzina na pateka "<<i+1<< " ": ";
13         cin >> pateki[i];
14         suma += pateki[i];
15     }
16     cout << "Vkupna dolzina: "<<suma<<" km"<< endl;
17     return 0;
18 }
19

```

Console Window (D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe):

```

Dolzina na pateka 1: 14
Dolzina na pateka 2: 7
Dolzina na pateka 3: 2
Dolzina na pateka 4: 9
Dolzina na pateka 5: 18
Vkupna dolzina: 50 km

Process returned 0 (0x0)   execution time : 8.285 s
Press any key to continue.

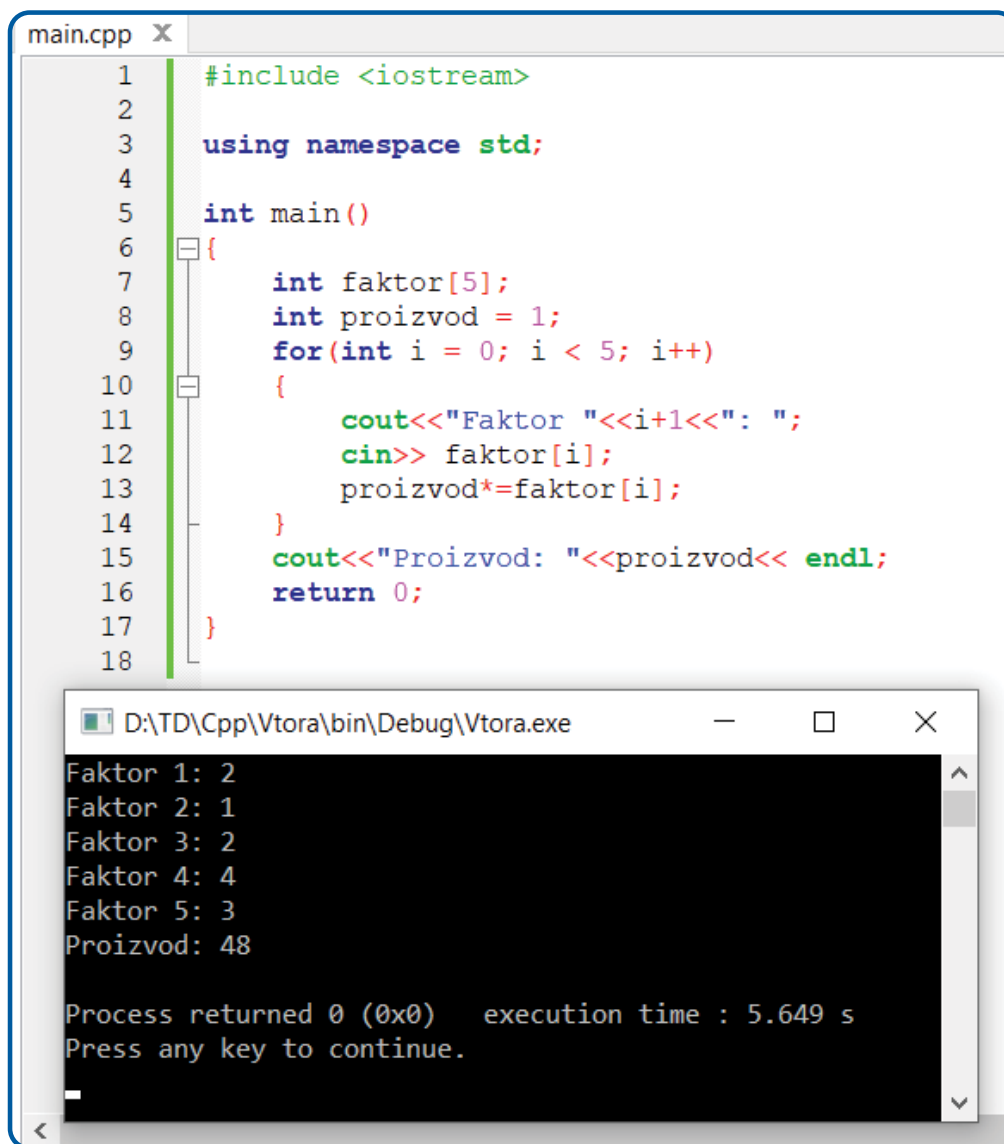
```

Во примерот податоците се внесуваат од тастатура, а при секое внесување новата вредност се додава кон вкупната сума. По завршувањето на циклусот се добива вкупната должина на сите патеки.

Во практичните програмски решенија не секогаш има потреба да биде пресметана сума. Во некои ситуации е потребно да биде определен производот на повеќе вредности. Во такви случаи се користи дополнителна променлива што на почетокот добива вредност 1. При секое внесување новата вредност се множи со претходниот резултат. На овој начин постепено се добива производот на сите внесени елементи.

Пример 2: Производ на фактори

Во научна лабораторија се внесуваат пет фактори на засилување. Да се пресмета нивниот производ.



The image shows a C++ program in a code editor and its execution in a console window. The code calculates the product of five factors entered by the user. The console output shows the factors and the final product.

```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int faktor[5];
8      int proizvod = 1;
9      for(int i = 0; i < 5; i++)
10     {
11         cout<<"Faktor "<<i+1<<" : ";
12         cin>> faktor[i];
13         proizvod*=faktor[i];
14     }
15     cout<<"Proizvod: "<<proizvod<< endl;
16     return 0;
17 }
18
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Faktor 1: 2
Faktor 2: 1
Faktor 3: 2
Faktor 4: 4
Faktor 5: 3
Proizvod: 48

Process returned 0 (0x0)   execution time : 5.649 s
Press any key to continue.
```

При обработка на внесени податоци често има потреба да биде определена нивната просечна вредност. За таа цел најпрво треба да биде пресметана вкупната сума на сите елементи од низата. Потоа добиениот резултат се дели со бројот на внесените елементи. Бидејќи резултатот од делењето може да содржи децимален дел, при пресметувањето на аритметичката средина најчесто се користи променлива од тип `double`. На овој начин се добива попрецизен резултат и се избегнува губење на децималните вредности. Ваквиот начин на работа има широка примена при статистички анализи, научни истражувања и обработка на различни видови податоци.

Пример 3: Просечна дневна изложеност на сонце

Во астрономска опсерваторија се бележи бројот на часови сончева светлина во текот на пет дена. Да се пресмета вкупниот број часови и просечната дневна изложеност.

The screenshot shows a C++ program in a text editor and its execution in a console window. The program calculates the average daily solar exposure over five days.

```

main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int casovi[5];
8      int suma = 0;
9      double prosek;
10     for(int i = 0; i < 5; i++)
11     {
12         cout<<"Casovi za den "<<i+1<<": ";
13         cin >> casovi[i];
14         suma += casovi[i];
15     }
16     prosek = (double)suma/5;
17     cout << "Vkupno casovi: "<<suma<<endl;
18     cout << "Prosecna vrednost: "<<prosek<<endl;
19     return 0;
20 }
21
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Casovi za den 1: 2
Casovi za den 2: 3
Casovi za den 3: 4
Casovi za den 4: 1
Casovi za den 5: 3
Vkupno casovi: 13
Prosecna vrednost: 2.6
Process returned 0 (0x0)   execution time : 6.037 s
Press any key to continue.

```

Во примерот податоците најпрво се внесуваат во низа. При внесувањето постепено се пресметува нивната сума. По завршувањето на циклусот се определува аритметичката средина со делење на вкупната сума со бројот на елементите. За зачувување на резултатот се користи променлива од тип `double`, што овозможува прикажување и на децималниот дел.

При обработка на податоци внесени од корисникот може да бидат извршени различни пресметки во зависност од поставениот проблем. Во некои ситуации се пресметува сума, во други производ, а во трети аритметичка средина. Заедничка карактеристика на сите овие задачи е тоа што податоците најпрво се внесуваат во низа, а потоа со помош на циклус се обработуваат. Благодарение на оваа постапка, една програма може да биде употребена за обработка на различни множества податоци без потреба од нејзино менување. Ваквиот пристап претставува основа за голем број програмски решенија што ќе бидат изучувани во наредните лекции.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Зошто при внесување на повеќе елементи во низа се користи циклус `for`?
2. Зошто при пресметување аритметичка средина често се користи типот `double`?
3. Која е предноста на внесување податоци од тастатура наместо нивно однапред задавање?



ЗАДАЧИ

1. Во ботаничка градина се внесуваат височините на шест новозасадени дрвја. Напиши програма што ќе ја пресметува нивната вкупна височина.
2. Во центар за роботика се внесува бројот на успешно завршени задачи на пет роботи. Напиши програма што ќе го пресметува производот на внесените вредности.
3. Во океанографска станица се внесуваат пет дневни мерења на висината на брановите. Напиши програма што ќе ја пресметува нивната аритметичка средина.
4. Во археолошка експедиција се внесува бројот на пронајдени предмети на седум локалитети. Напиши програма што ќе ја пресметува вкупната бројност на предметите.
5. Во центар за обновлива енергија се внесува произведената енергија од пет соларни системи. Напиши програма што ќе ги пресметува вкупната и просечната произведена енергија.



ЗАПОМНИ ШТО НАУЧИ

- Елементите на низа можат да бидат внесени од тастатура со помош на циклус `for`.
- Внесените податоци може веднаш да бидат обработени.
- За пресметување на вкупна вредност се користи променлива за сума.
- За пресметување на производ се користи променлива со почетна вредност 1.
- Аритметичката средина се добива со делење на вкупната сума со бројот на елементите.
- При пресметување на просек најчесто се користи типот `double`.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите информатички системи голем дел од податоците не се однапред познати, туку се внесуваат во текот на работата на програмата. На пример, мерењата од сензори, финансиските трансакции, медицинските анализи и спортските резултати постојано се надополнуваат со нови информации. Поради тоа, програмите треба да бидат способни да примаат податоци од корисникот и веднаш да ги обработуваат. Токму оваа можност претставува една од најважните примени на еднодимензионалните низи и циклусите во современото програмирање.



ОПРЕДЕЛУВАЊЕ НА НАЈГОЛЕМ И НАЈМАЛ ЕЛЕМЕНТ ВО НИЗА

ВЕЌЕ ЗНАЕШ ДА...

- декларираш и да иницијализираш еднодимензионална низа;
- внесуваш елементи во низа;
- обработуваш податоци со циклус for;
- пресметуваш сума, производ и аритметичка средина;
- комбинираш внесување и обработка на податоци.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Како се внесуваат повеќе елементи во низа?
2. Зошто при работа со низи најчесто се користи циклус for?
3. Со која почетна вредност започнува променливата за сума?
4. Како се пресметува аритметичка средина?
5. Зошто е корисно податоците да бидат обработени веднаш по нивното внесување?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- определуваш најголем елемент во низа;
- определуваш најмал елемент во низа;
- споредуваш елементи од низа;
- користиш дополнителни променливи при споредување;
- обработуваш податоци со примена на условни конструкции.

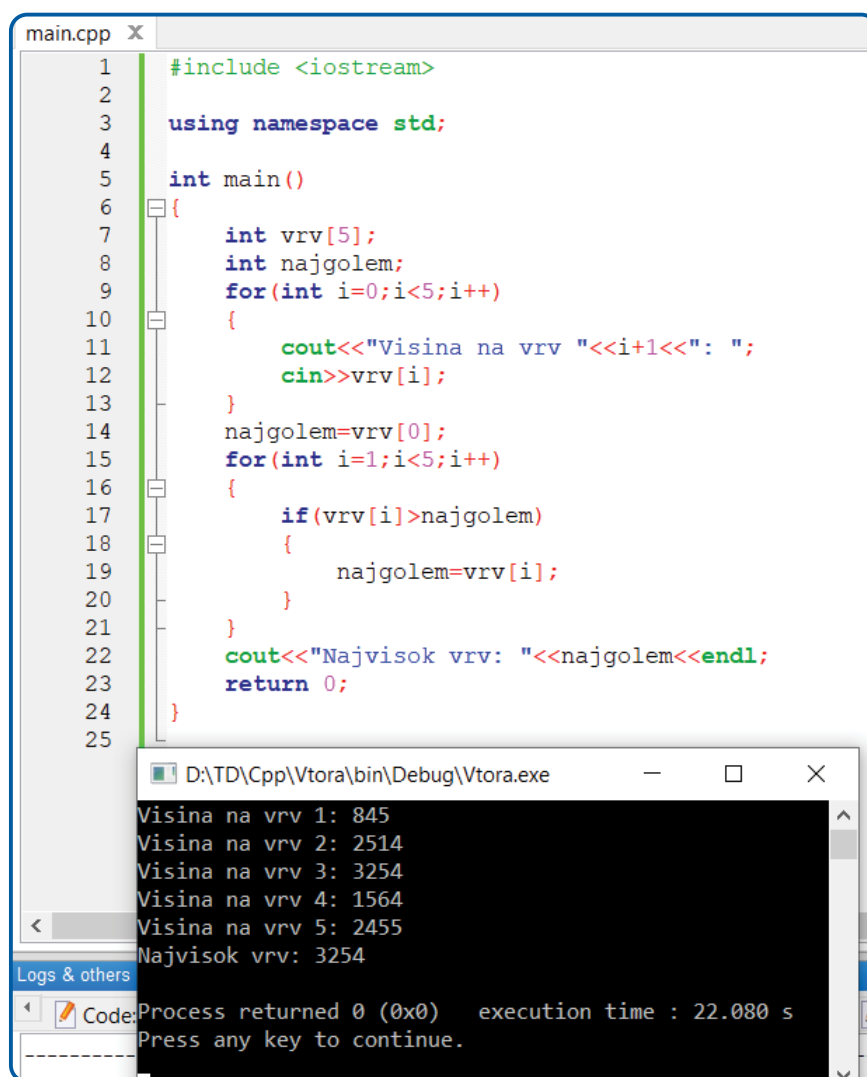
При обработка на податоци често има потреба да биде определена најголемата или најмалата вредност од одредено множество податоци. На пример, може да биде потребно да биде определена најголемата измерена височина, најмалата количина врнежи или највисоката постигната брзина.

За решавање на вакви задачи се користи споредување на елементите од низата. Најпрво една вредност се избира како почетна, а потоа останатите елементи последователно се споредуваат со неа. Доколку се пронајде поголема или помала вредност, почетната вредност се заменува со новата.

При определување на најголем или најмал елемент потребно е најпрво една вредност од низата да биде избрана како почетна. Најчесто тоа е првиот елемент од низата. Потоа со помош на циклус последователно се разгледуваат останатите елементи. Доколку некој од нив е поголем од моментално запаметената најголема вредност, таа се заменува. На ист начин може да биде определена и најмалата вредност. Благодарение на оваа постапка, сите податоци се обработуваат со едно поминување низ низата.

Пример 1: Највисок планински врв

Во планинарски водич се внесуваат надморските височини на пет планински врвови. Да се определи највисокиот врв.



The image shows a C++ program in a code editor and its execution output in a console window. The code defines an array of 5 integers, reads 5 values from the user, and finds the maximum value.

```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int vrv[5];
8      int najgolem;
9      for(int i=0;i<5;i++)
10     {
11         cout<<"Visina na vrv "<<i+1<<" : ";
12         cin>>vrv[i];
13     }
14     najgolem=vrv[0];
15     for(int i=1;i<5;i++)
16     {
17         if(vrv[i]>najgolem)
18         {
19             najgolem=vrv[i];
20         }
21     }
22     cout<<"Najvisok vrv: "<<najgolem<<endl;
23     return 0;
24 }
25
```

The console window shows the following output:

```
D:\TD\C++\Vtora\bin\Debug\Vtora.exe
Visina na vrv 1: 845
Visina na vrv 2: 2514
Visina na vrv 3: 3254
Visina na vrv 4: 1564
Visina na vrv 5: 2455
Najvisok vrv: 3254
Process returned 0 (0x0)   execution time : 22.080 s
Press any key to continue.
```

Во примерот првиот елемент се избира како почетна вредност. Потоа сите останати елементи се споредуваат со него. Доколку се пронајде поголема вредност, таа станува нова најголема вредност.

При определување на најмал елемент се користи истата постапка. Разликата е во тоа што наместо да се бара поголема вредност, се проверува дали тековниот елемент е помал од моментално запаметената најмала вредност. На овој начин може да биде определена најмалата вредност во едно множество податоци без дополнителни сложени пресметки.

Пример 2: Најмала количина гориво

Во истражувачка експедиција се внесуваат количините гориво во шест резервоари. Да се определи резервоарот со најмала количина гориво.

The image shows a C++ program in a code editor and its execution output in a console window.

Code Editor (main.cpp):

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int gorivo[6];
8      int najmal;
9      for(int i=0;i<6;i++)
10     {
11         cout<<"Rezervoar "<<i+1<<" : ";
12         cin>>gorivo[i];
13     }
14     najmal=gorivo[0];
15     for(int i=1;i<6;i++)
16     {
17         if(gorivo[i]<najmal)
18         {
19             najmal=gorivo[i];
20         }
21     }
22     cout<<"Najmala kolicina: "<<najmal<<endl;
23     return 0;
24 }
25

```

Console Output (D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe):

```

Rezervoar 1: 230
Rezervoar 2: 540
Rezervoar 3: 155
Rezervoar 4: 652
Rezervoar 5: 145
Rezervoar 6: 211
Najmala kolicina: 145
Process returned 0 (0x0)   execution time : 15.757 s
Press any key to continue.

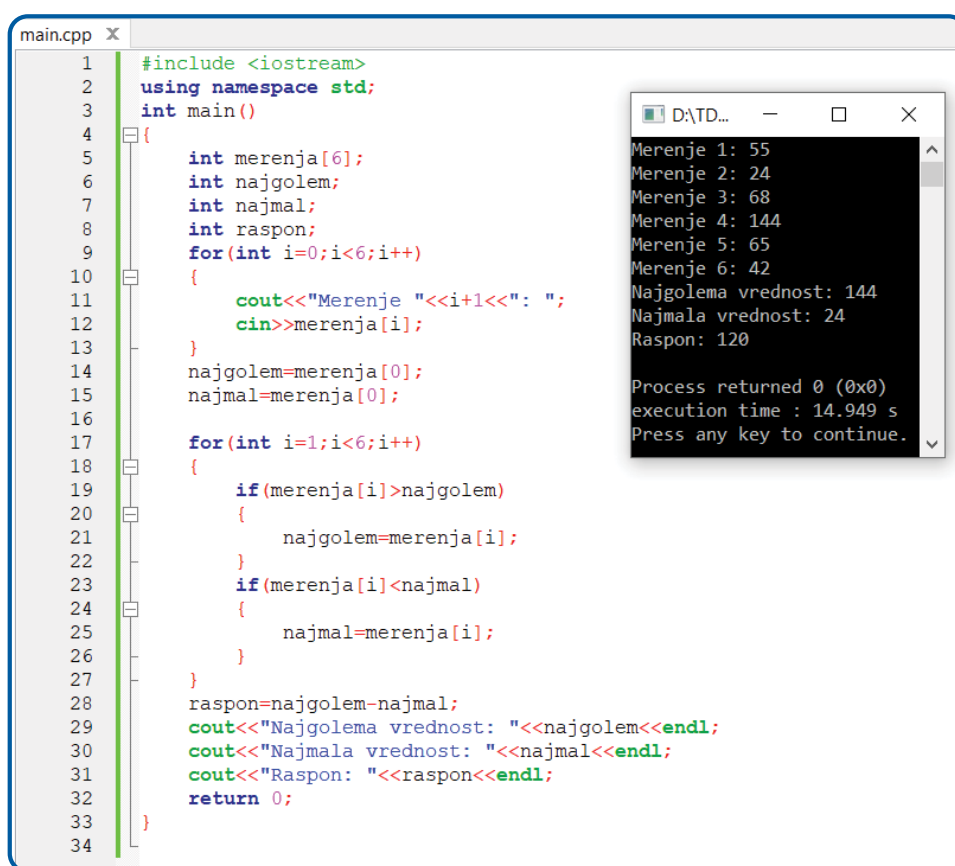
```

Во примерот се применува истата програмска идеја, но овојпат се определува најмалата вредност од внесените податоци.

Во практичните програмски задачи често има потреба истовремено да бидат определени и најголемата и најмалата вредност од едно множество податоци. Притоа не е потребно низата да биде обработувана двапати. Во текот на едно поминување низ елементите може истовремено да бидат ажурирани и најголемата и најмалата вредност. По завршувањето на обработката, врз добиените резултати може да се извршат и дополнителни пресметки. Една од нив е определувањето на распонот на податоците, кој се добива како разлика помеѓу најголемата и најмалата вредност.

Пример 3: Распон на измерени вредности

Во научна станица се внесуваат шест измерени вредности. Да се определат најголемата, најмалата вредност и распонот на мерењата.



The image shows a C++ program in a code editor and its execution output in a terminal window. The code defines an array of six measurements, finds the maximum and minimum values, and calculates the range (difference between max and min). The output displays each measurement, the maximum value (144), the minimum value (24), and the range (120).

```
main.cpp x
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int merenja[6];
6      int najgolem;
7      int najmal;
8      int raspon;
9      for(int i=0;i<6;i++)
10     {
11         cout<<"Merenje "<<i+1<<" ";
12         cin>>merenja[i];
13     }
14     najgolem=merenja[0];
15     najmal=merenja[0];
16
17     for(int i=1;i<6;i++)
18     {
19         if(merenja[i]>najgolem)
20         {
21             najgolem=merenja[i];
22         }
23         if(merenja[i]<najmal)
24         {
25             najmal=merenja[i];
26         }
27     }
28     raspon=najgolem-najmal;
29     cout<<"Najgolema vrednost: "<<najgolem<<endl;
30     cout<<"Najmala vrednost: "<<najmal<<endl;
31     cout<<"Raspon: "<<raspon<<endl;
32     return 0;
33 }
34
```

Output (D:\TD...):

```
Merenje 1: 55
Merenje 2: 24
Merenje 3: 68
Merenje 4: 144
Merenje 5: 65
Merenje 6: 42
Najgolema vrednost: 144
Najmala vrednost: 24
Raspon: 120

Process returned 0 (0x0)
execution time : 14.949 s
Press any key to continue.
```

Во примерот со едно поминување низ низата се определуваат и најголемата и најмалата вредност. Потоа со нивна помош се пресметува распонот на податоците.

При обработка на податоци определувањето на најголемите и најмалите вредности има широка примена. На пример, може да биде определена највисоката температура, најмалата потрошувачка на енергија или најголемата измерена брзина. Независно од видот на податоците, основната постапка останува иста. Најпрво се избира почетна вредност, потоа сите останати елементи последователно се споредуваат со неа, а по потреба се ажурираат резултатите од споредувањето.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Зошто при определување најголема вредност како почетна најчесто се избира првиот елемент?
2. Како се определува најмал елемент во низа?
3. Дали најголемата и најмалата вредност може да се определат со едно поминување низ низата?
4. Што претставува распон на податоците?
5. Зошто при овие задачи најчесто се користи циклус for?



ЗАДАЧИ

1. Во метеоролошка станица се внесуваат количините снег измерени во пет планински центри. Напиши програма што ќе ја определува најголемата измерена количина.
2. Во центар за заштита на животни се внесуваат тежините на шест млади мечки. Напиши програма што ќе ја определува најмалата измерена тежина.
3. Во океанографска експедиција се внесуваат длабочините на седум мерења. Напиши програма што ќе ги определува најголемата и најмалата длабочина.
4. Во истражување на квалитетот на воздухот се внесуваат дневни вредности од пет мерни станици. Напиши програма што ќе го пресметува распонот на измерените вредности.
5. Во археолошка експедиција се внесуваат старостите на шест пронајдени предмети. Напиши програма што ќе ги определува најголемата, најмалата старост и разликата помеѓу нив.



ЗАПОМНИ ШТО НАУЧИ

- Најголемата и најмалата вредност може да се определат со споредување на елементите од низата.
- Почетната вредност најчесто е првиот елемент од низата.
- При пронаоѓање на поголема или помала вредност резултатот се ажурира.
- Најголемата и најмалата вредност може да се определат со едно поминување низ низата.
- Распонот на податоците се добива како разлика помеѓу најголемата и најмалата вредност.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Определувањето на најголема и најмала вредност претставува една од најчесто користените постапки при анализа на податоци. Во науката може да биде определена највисоката измерена температура, во медицината најголемата измерена вредност на одреден показател, а во спортот најдобриот постигнат резултат. Иако современите информатички системи обработуваат огромни количества податоци, основната идеја останува иста – податоците последователно се споредуваат, а добиените резултати се ажурираат кога ќе биде пронајдена поголема или помала вредност.



2.21

ПРЕБРОЈУВАЊЕ НА ЕЛЕМЕНТИ ВО НИЗА

ВЕЌЕ ЗНАЕШ ДА...

- внесуваш елементи во еднодимензионална низа;
- обработуваш елементи со циклус for;
- пресметуваш сума и производ;
- определуваш најголем и најмал елемент;
- користиш условни конструкции при обработка на податоци.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Зошто при обработка на низа најчесто се користи циклус for?
2. Како се определува најголем елемент во низа?
3. Како се определува најмал елемент?
4. Дали е можно повеќе вредности да бидат обработени со едно поминување низ низата?
5. Што претставува распон на податоците?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- пребројуваш елементи што исполнуваат зададен услов;
- користиш променлива за броење;
- комбинираш циклуси и условни конструкции;
- применуваш логички изрази при обработка на низи;
- анализираш податоци според различни критериуми.

При обработка на податоците често нема потреба само да биде определена нивната сума или најголемата вредност. Во многу практични ситуации потребно е да биде определено колку елементи имаат одредено својство. На пример, може да биде потребно да биде определен бројот на денови со врнежи, бројот на возила што ја надминале дозволената брзина или бројот на успешно извршени мерења.

За решавање на вакви задачи се користи променлива за броење. На почетокот таа добива вредност 0. При обработката на секој елемент се проверува дали е исполнет зададениот услов. Ако е исполнет

условот, вредноста на променливата се зголемува за 1. По обработката на сите елементи, таа го содржи вкупниот број на елементи што го исполнуваат условот.

При пребројување на елементите од една низа најпрво се создава променлива за броење што добива почетна вредност 0. Потоа сите елементи последователно се обработуваат со помош на циклус. За секој елемент се проверува дали го исполнува зададениот услов. Ако е исполнет условот, вредноста на променливата за броење се зголемува за 1. По обработката на последниот елемент, променливата го содржи конечниот резултат. Ваквиот начин на работа има широка примена при статистичка обработка и анализа на различни видови податоци.

Пример 1: Високи растенија

Во ботаничка градина се внесуваат височините на седум растенија. Да се определи колку растенија се повисоки од 100 cm.

The image shows a C++ program in a code editor and its execution output in a console window.

Code Editor (main.cpp):

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int visina[7];
8      int broj = 0;
9      for(int i=0; i<7; i++)
10     {
11         cout<<"Visina "<<i+1<<": ";
12         cin>>visina[i];
13         if(visina[i] > 100)
14         {
15             broj++;
16         }
17     }
18     cout << "Broj na rastenija: "<<broj<< endl;
19     return 0;
20 }
21

```

Console Window (D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe):

```

Visina 1: 105
Visina 2: 92
Visina 3: 56
Visina 4: 213
Visina 5: 84
Visina 6: 155
Visina 7: 142
Broj na rastenija: 4

```

Logs:

```

Process returned 0 (0x0)   execution time : 13.936 s
Press any key to continue.

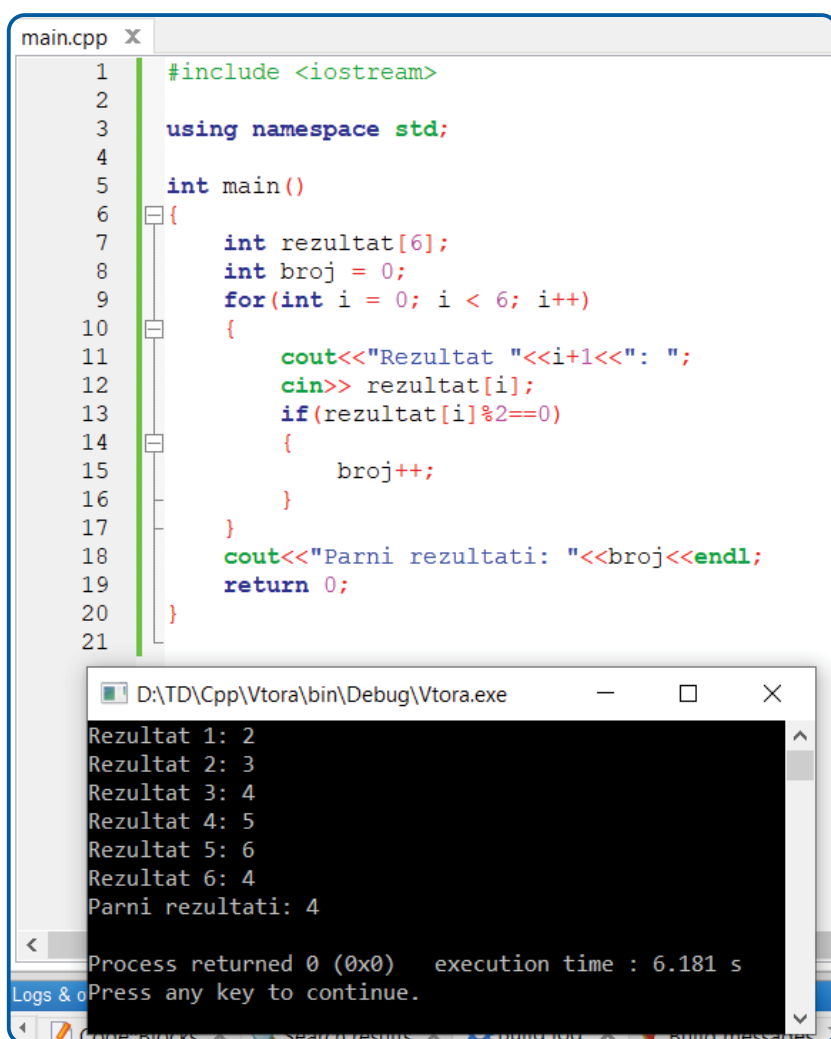
```

Во примерот секоја внесена вредност се проверува. Ако е растението повисоко од 100 cm, променливата за броене се зголемува за 1.

Во практичните задачи условот не мора да биде поврзан со големината на вредностите. Често има потреба да бидат преброени елементи што имаат одредено математичко својство. Едно од најчестите својства е парноста на броевите. За таа цел се користи операторот %, кој овозможува да биде определен остатокот при делење.

Пример 2: Парни резултати

Во спортски центар се внесуваат шест резултати. Да се определи колку од нив се парни броеви.



```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int rezultat[6];
8      int broj = 0;
9      for(int i = 0; i < 6; i++)
10     {
11         cout<<"Rezultat "<<i+1<<" : ";
12         cin>> rezultat[i];
13         if(rezultat[i]%2==0)
14         {
15             broj++;
16         }
17     }
18     cout<<"Parni rezultati: "<<broj<<endl;
19     return 0;
20 }
21
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

```
Rezultat 1: 2
Rezultat 2: 3
Rezultat 3: 4
Rezultat 4: 5
Rezultat 5: 6
Rezultat 6: 4
Parni rezultati: 4

Process returned 0 (0x0)   execution time : 6.181 s
Press any key to continue.
```

Во примерот за секоја внесена вредност се проверува дали е парен број. Ако е исполнет условот, се зголемува бројот на парни вредности.

При обработка на податоците честопати има потреба да биде определен бројот на елементи што се наоѓаат во одреден интервал. На пример, може да биде потребно да биде определен бројот на температури помеѓу две вредности, бројот на ученици со оценки во одреден опсег или бројот на производи

што ги исполнуваат поставените услови. Во вакви ситуации за секој елемент се проверуваат повеќе услови истовремено. Ако се сите услови исполнети, променливата за броене се зголемува за 1. На овој начин лесно може да биде определен бројот на елементи што припаѓаат на зададениот интервал.

Пример 3: Вредности во зададен интервал

Во истражувачка станица се внесуваат седум измерени вредности. Да се определи колку од нив се наоѓаат во интервалот од 20 до 50.

```

main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int vrednost[7];
8      int broj = 0;
9      for(int i=0;i<7;i++)
10     {
11         cout<<"Vrednost "<<i+1<<": ";
12         cin >> vrednost[i];
13         if(vrednost[i]>=20&&vrednost[i]<=50)
14         {
15             broj++;
16         }
17     }
18     cout<<"Broj na elementi: "<<broj<<endl;
19     return 0;
20 }
21
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Vrednost 1: 18
Vrednost 2: 22
Vrednost 3: 87
Vrednost 4: 54
Vrednost 5: 35
Vrednost 6: 40
Vrednost 7: 11
Broj na elementi: 3
Process returned 0 (0x0)   execution time : 16.578 s
Press any key to continue.

```

Во примерот секоја внесена вредност се проверува дали припаѓа на зададениот интервал. Ако е исполнет условот, променливата за броене се зголемува за 1. По обработката на сите елементи се добива конечниот резултат.

При пребројување на елементите условите што треба да бидат исполнети може да бидат многу различни. Во некои задачи се проверува дали е вредноста поголема од зададен број, во други дали е парна, а во трети дали припаѓа на определен интервал. Независно од поставениот проблем, основната постап-

ка останува иста. Елементите последователно се обработуваат, условот се проверува за секој елемент, а променливата за броење се зголемува кога е исполнет условот. Благодарение на оваа техника може да бидат решени голем број практични програмски задачи.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. За што служи променливата за броење?
2. Со која почетна вредност најчесто започнува променливата за броење?
3. Кога се зголемува нејзината вредност?
4. Како се проверува дали е бројот парен?
5. Како може да биде проверено дали елементот припаѓа на определен интервал?



ЗАДАЧИ

1. Во национален парк се внесуваат височините на осум водопади. Напиши програма што ќе определи колку водопади се повисоки од 50 метри.
2. Во роботска лабораторија се внесуваат броевите на успешно завршени задачи на шест роботи. Напиши програма што ќе определи колку од резултатите се непарни броеви.
3. Во центар за обновлива енергија се внесува дневното производство на седум соларни системи. Напиши програма што ќе определи колку вредности се наоѓаат помеѓу 100 и 300.
4. Во океанографска експедиција се внесуваат длабочините на шест мерења. Напиши програма што ќе определи колку мерења се поголеми од 500 метри.
5. Во астрономска опсерваторија се внесуваат броевите на забележани метеори во текот на пет ноќи. Напиши програма што ќе определи во колку ноќи се регистрирани повеќе од 20 метеори.



ЗАПОМНИ ШТО НАУЧИ

- За пребројување на елементи се користи променлива за броење.
- Променливата за броење најчесто започнува со вредност 0.
- Нејзината вредност се зголемува кога е исполнет зададениот услов.
- Условот може да биде едноставен или составен од повеќе проверки.
- Со едно поминување низ низата може да биде определен бројот на елементи што имаат определено својство.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Пребројувањето на елементите претставува една од најчесто применуваните техники при анализа на податоци. Во медицината може да биде определен бројот на пациенти што имаат определен симптом, во екологијата бројот на денови со врнежи, а во спортот бројот на успешно реализирани обиди. Иако современите информатички системи обработуваат огромни количества податоци, основната идеја останува непроменета – секој податок се проверува, а ако го исполнува зададениот услов, резултатот од пребројувањето се ажурира.



НИЗИ ОД КАРАКТЕРИ И ПРЕБАРУВАЊЕ НА ЗНАЦИ

ВЕЌЕ ЗНАЕШ ДА...

- декларираш и да користиш еднодимензионални низи од цели броеви;
- внесуваш и да обработуваш елементи во низа;
- користиш циклус for;
- применуваш условни конструкции;
- пребројуваш елементи што исполнуваат определен услов.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. Што претставува еднодимензионална низа?
2. Од кој број започнува нумерирањето на елементите?
3. Зошто при работа со низи се користи циклус for?
4. За што служи променливата за броење?
5. Како може да биде проверено дали елемент исполнува определен услов?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- користиш низи од карактери;
- внесуваш знаци во низа;
- пребаруваш знак во низа;
- определуваш дали има знак;
- пребројуваш појавувања на знак.

Освен броеви, во програмите често има потреба да бидат обработувани и знаци. На пример, може да биде потребно да биде проверено дали одредена буква се наоѓа во код, дали определен симбол се појавува во текст или колку пати е употребен одреден знак.

За зачувување на поединечни знаци во програмскиот јазик C++ се користи типот `char`. Повеќе знаци може да бидат зачувани во низа од карактери. На овој начин секој знак има своја позиција во низата и може поединечно да биде обработен.

Низата од карактери се декларира на ист начин како и низите од цели броеви. Разликата е во тоа што наместо типот `int` се користи типот `char`. Секој елемент од низата содржи по еден знак. Елементите може да се букви, цифри или други симболи. При обработката на низата секој знак може да биде спореден со определена вредност и врз основа на резултатот да биде донесена одлука. Благодарение на оваа можност може да се решат различни практични задачи поврзани со обработка на текстуални податоци.

Пример 1: Проверка дали има знак

Во контролен систем се внесуваат шест ознаки. Да се провери дали меѓу нив се наоѓа буквата А.



```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      char oznaka[6];
8      bool pronajden=false;
9      for(int i=0;i<6;i++)
10     {
11         cout<<"Oznaka "<<i+1<<" : ";
12         cin>>oznaka[i];
13     }
14     for(int i=0;i<6;i++)
15     {
16         if(oznaka[i]=='A')
17         {
18             pronajden=true;
19         }
20     }
21     if(pronajden)
22     {
23         cout<<"Znakot postoi.";
24     }
25     else
26     {
27         cout<<"Znakot ne postoi.";
28     }
29     return 0;
30 }
31
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

Oznaka 1: A
Oznaka 2: B
Oznaka 3: A
Oznaka 4: 3
Oznaka 5: c
Oznaka 6: a

Znakot postoi.

Process returned 0 (0x0) execution time : 16.964 s
Press any key to continue.

Во примерот сите знаци најпрво се внесуваат во низа. Потоа последователно се проверува дали некој од нив е еднаков на буквата А. Ако биде пронајдена, резултатот се запаметува.

При обработка на знаци често не е доволно да биде определено дали има одреден знак. Во многу ситуации има потреба да биде определено колку пати се појавува. За таа цел се користи променлива за броење што се зголемува секогаш кога ќе биде пронајден бараниот знак.

Пример 2: Пребројување на појавувања

Во комуникациски систем се внесуваат осум ознаки. Да се определи колку пати се појавува буквата К.

```

main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      char znak[8];
8      int broj = 0;
9
10     for(int i=0;i<8;i++)
11     {
12         cout<<"Znak "<<i+1<<": ";
13         cin>>znak[i];
14     }
15     for(int i=0;i<8;i++)
16     {
17         if(znak[i]=='K')
18         {
19             broj++;
20         }
21     }
22     cout<<"Broj na pojavuvanja: "<<broj<<endl;
23     return 0;
24 }
25
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Znak 1: K
Znak 2: A
Znak 3: k
Znak 4: R
Znak 5: x
Znak 6: X
Znak 7: K
Znak 8: k
Broj na pojavuvanja: 2

Logs & others Process returned 0 (0x0)   execution time : 24.749 s
Press any key to continue.
Code::
Executing:

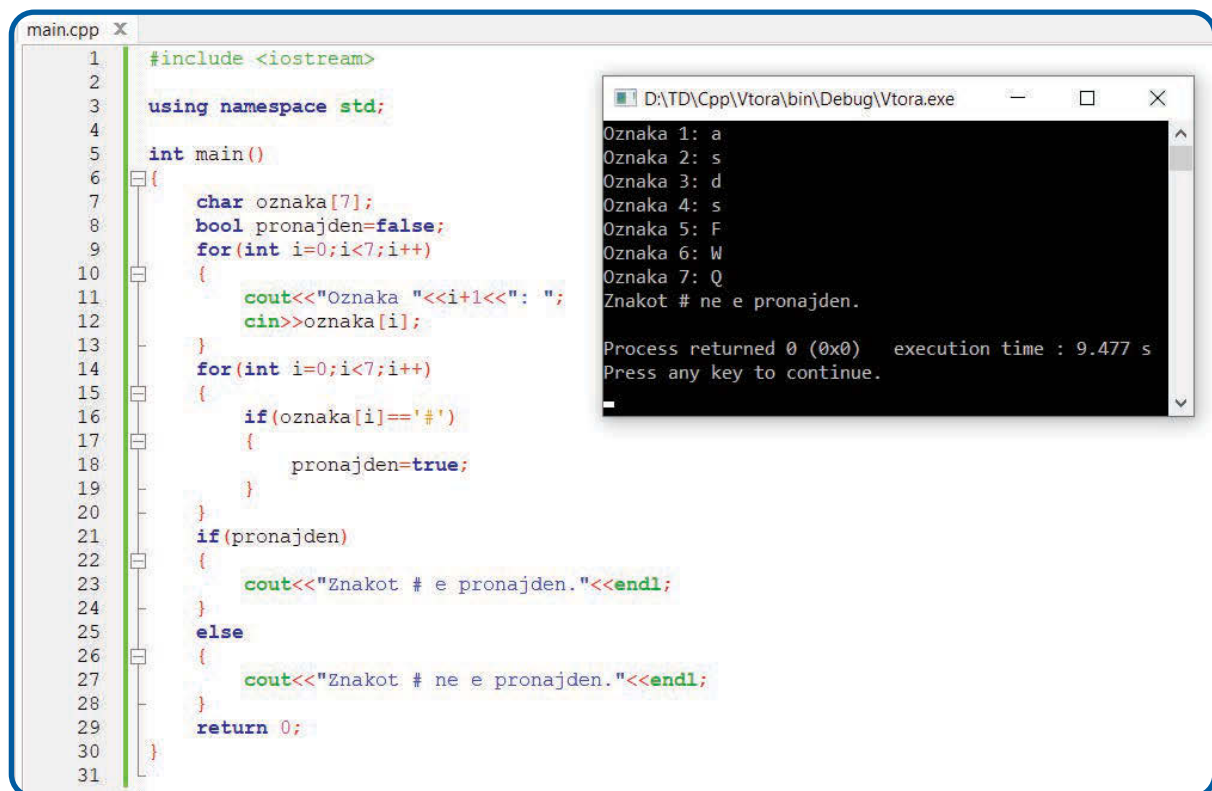
```

Во примерот се применува техниката на пребројување. Секојпат кога ќе биде пронајдена буква-та K, променливата за броене се зголемува за 1.

При работа со низи од карактери не секогаш има потреба да биде пребарана само определена буква. Во практичните програмски задачи често се проверува дали се појавува определен симбол. Такви симболи може да се знаците #, @, * или други ознаки што имаат посебно значење во одредени системи. Постапката на обработка останува непроменета. Секој елемент од низата последователно се споредува со бараниот знак, а ако биде пронајден, се донесува соодветен заклучок.

Пример 3: Проверка на специјален знак

Во систем за пристап се внесуваат седум ознаки. Да се провери дали меѓу нив се наоѓа знакот #.



```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      char oznaka[7];
8      bool pronajden=false;
9      for(int i=0;i<7;i++)
10     {
11         cout<<"Oznaka "<<i+1<<": ";
12         cin>>oznaka[i];
13     }
14     for(int i=0;i<7;i++)
15     {
16         if(oznaka[i]=='#')
17         {
18             pronajden=true;
19         }
20     }
21     if(pronajden)
22     {
23         cout<<"Znakot # e pronajden."<<endl;
24     }
25     else
26     {
27         cout<<"Znakot # ne e pronajden."<<endl;
28     }
29     return 0;
30 }
31
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Oznaka 1: a
Oznaka 2: s
Oznaka 3: d
Oznaka 4: s
Oznaka 5: F
Oznaka 6: W
Oznaka 7: Q
Znakot # ne e pronajden.
Process returned 0 (0x0)   execution time : 9.477 s
Press any key to continue.
```

Во примерот сите внесени знаци последователно се споредуваат со знакот #. Ако биде пронајден, програмата го известува корисникот дека знакот го има во низата.

При обработка на низи од карактери може да бидат поставени различни услови за пребарување. Во некои задачи се бара присуство на определена буква, во други се пребројува бројот на појавувања на одреден знак, а во трети се проверува дали се појавува определен симбол. Независно од видот на условот, основната постапка останува иста. Секој елемент од низата се обработува, се врши споредување и врз основа на резултатот се донесува заклучок. Оваа техника претставува основа за обработката на текстуални податоци во програмските јазици.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Што претставува низа од карактери?
2. Кој тип на податок се користи за зачувување на еден знак?
3. Како може да биде проверено дали определен знак го има во низа?
4. Како може да биде определено колку пати се појавува одреден знак?
5. Дали во низа од карактери може да бидат зачувани букви, цифри и специјални симболи?



ЗАДАЧИ

1. Во контролен систем се внесуваат осум ознаки. Напиши програма што ќе провери дали меѓу нив се наоѓа буквата М.
2. Во комуникациски центар се внесуваат шест знаци. Напиши програма што ќе определи колку пати се појавува цифрата '5'.
3. Во информатичка лабораторија се внесуваат десет знаци. Напиши програма што ќе провери дали е внесен знакот @.
4. Во систем за евиденција се внесуваат седум ознаки. Напиши програма што ќе определи колку пати се појавува буквата S.
5. Во истражувачка станица се внесуваат девет симболи. Напиши програма што ќе провери дали се појавува знакот *.



ЗАПОМНИ ШТО НАУЧИ

- Низата од карактери претставува множество од поединечни знаци.
- За зачувување на еден знак се користи типот char.
- Знаците може да се букви, цифри или специјални симболи.
- Пребарувањето на знак се врши со споредување на елементите од низата.
- Со помош на променлива за броење може да биде определен бројот на појавувања на определен знак.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите информатички системи обработката на текстуални податоци претставува една од најважните програмски активности. При внесување кориснички имиња, лозинки, ознаки на производи или различни кодови, програмите често проверуваат дали се користени определени знаци или симболи. Иако во оваа лекција се обработуваат поединечни знаци, истите принципи подоцна се применуваат и при работа со зборови, реченици и поголеми текстуални податоци.



2.23

ПРИМЕНА НА МАТЕМАТИЧКИ ФУНКЦИИ ВРЗ ЕЛЕМЕНТИТЕ НА НИЗА

ВЕЌЕ ЗНАЕШ ДА...

- внесуваш елементи во еднодимензионална низа;
- обработуваш елементи со циклус `for`;
- пресметуваш сума, производ и просек;
- определуваш најголем и најмал елемент;
- користиш функции од математичката библиотека.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

1. За што служи библиотеката `cmath`?
2. Која функција го пресметува квадратниот корен?
3. Која функција се користи за степенување?
4. За што служи функцијата `abs()`?
5. Дали функциите од математичката библиотека може да се применуваат врз елементите на низа?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- применуваш математички функции врз елементите на низа;
- комбинираш низи и библиотеката `cmath`;
- обработуваш внесени податоци со математички пресметки;
- споредуваш добиени резултати;
- решаваш практични задачи со комбинирана примена на изучените техники.

При обработка на податоци често има потреба врз секој елемент од една низа да биде извршена определена математичка операција. Во практичните програмски решенија може да биде потребно да биде пресметан квадратен корен, квадрат на број или апсолутна вредност на внесените податоци. За таа цел во програмскиот јазик C++ се користи математичката библиотека `cmath`.

Функциите од оваа библиотека може да се применат врз секој елемент од низата. Со помош на циклус последователно се обработуваат сите вредности, а врз секоја од нив се извршува потребната математичка пресметка. На овој начин може да се решат различни практични проблеми од науката, техниката и од секојдневниот живот.

При примена на математички функции врз елементите од една низа, најпрво се внесуваат податоците, а потоа секој елемент последователно се обработува. За секој елемент може да биде пресметан квадратен корен, квадрат, апсолутна вредност или друга математичка функција. Добиените резултати може веднаш да бидат прикажани или дополнително обработени. Благодарение на оваа постапка, голем број податоци може брзо да бидат анализирани и обработени со релативно мал број програмски инструкции.

Пример 1: Квадратен корен на измерени растојанија

Во научна лабораторија се внесуваат пет измерени растојанија. Да се пресмета квадратниот корен на секое растојание.

```

main.cpp x
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main()
7  {
8      double rastojanie[5];
9      for(int i=0;i<5;i++)
10     {
11         cout<<"Rastojanie "<<i+1<<": ";
12         cin>>rastojanie[i];
13     }
14     cout<<endl;
15     for(int i=0;i<5;i++)
16     {
17         cout<<"Kvadraten koren: "<<sqrt(rastojanie[i])<<endl;
18     }
19     return 0;
20 }
21
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Rastojanie 1: 5
Rastojanie 2: 4
Rastojanie 3: 3
Rastojanie 4: 81
Rastojanie 5: 6

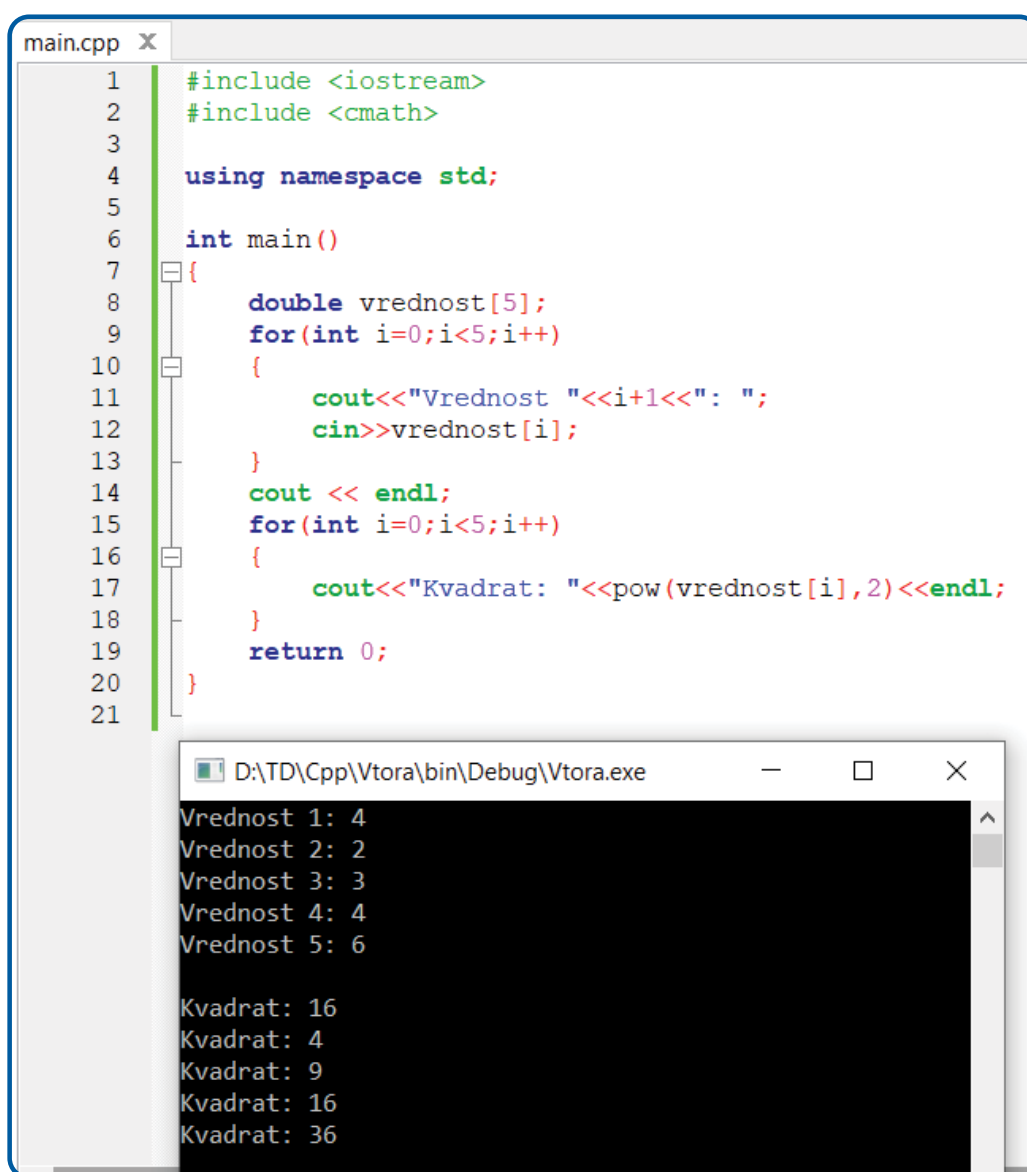
Kvadraten koren: 2.23607
Kvadraten koren: 2
Kvadraten koren: 1.73205
Kvadraten koren: 9
Kvadraten koren: 2.44949
  
```

Во примерот секој внесен податок се обработува независно. Со помош на функцијата `sqrt()` се пресметува квадратниот корен на секоја вредност и резултатот веднаш се прикажува.

При обработка на податоци често има потреба врз сите елементи од една низа да биде применета иста математичка операција. На пример, може да биде потребно да бидат пресметани квадратите на сите измерени вредности или да биде определена апсолутната вредност на секое мерење. Во вакви ситуации секој елемент последователно се обработува, а врз него се применува соодветната математичка функција. Благодарение на овој пристап, голем број податоци може брзо да бидат обработени со едноставна програмска постапка.

Пример 2: Квадрати на измерени вредности

Во истражувачки центар се внесуваат пет измерени вредности. Да се пресмета квадратот на секоја вредност.



```
main.cpp X
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main()
7  {
8      double vrednost[5];
9      for(int i=0;i<5;i++)
10     {
11         cout<<"Vrednost "<<i+1<<": ";
12         cin>>vrednost[i];
13     }
14     cout << endl;
15     for(int i=0;i<5;i++)
16     {
17         cout<<"Kvadrat: "<<pow(vrednost[i],2)<<endl;
18     }
19     return 0;
20 }
21
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

```
Vrednost 1: 4
Vrednost 2: 2
Vrednost 3: 3
Vrednost 4: 4
Vrednost 5: 6

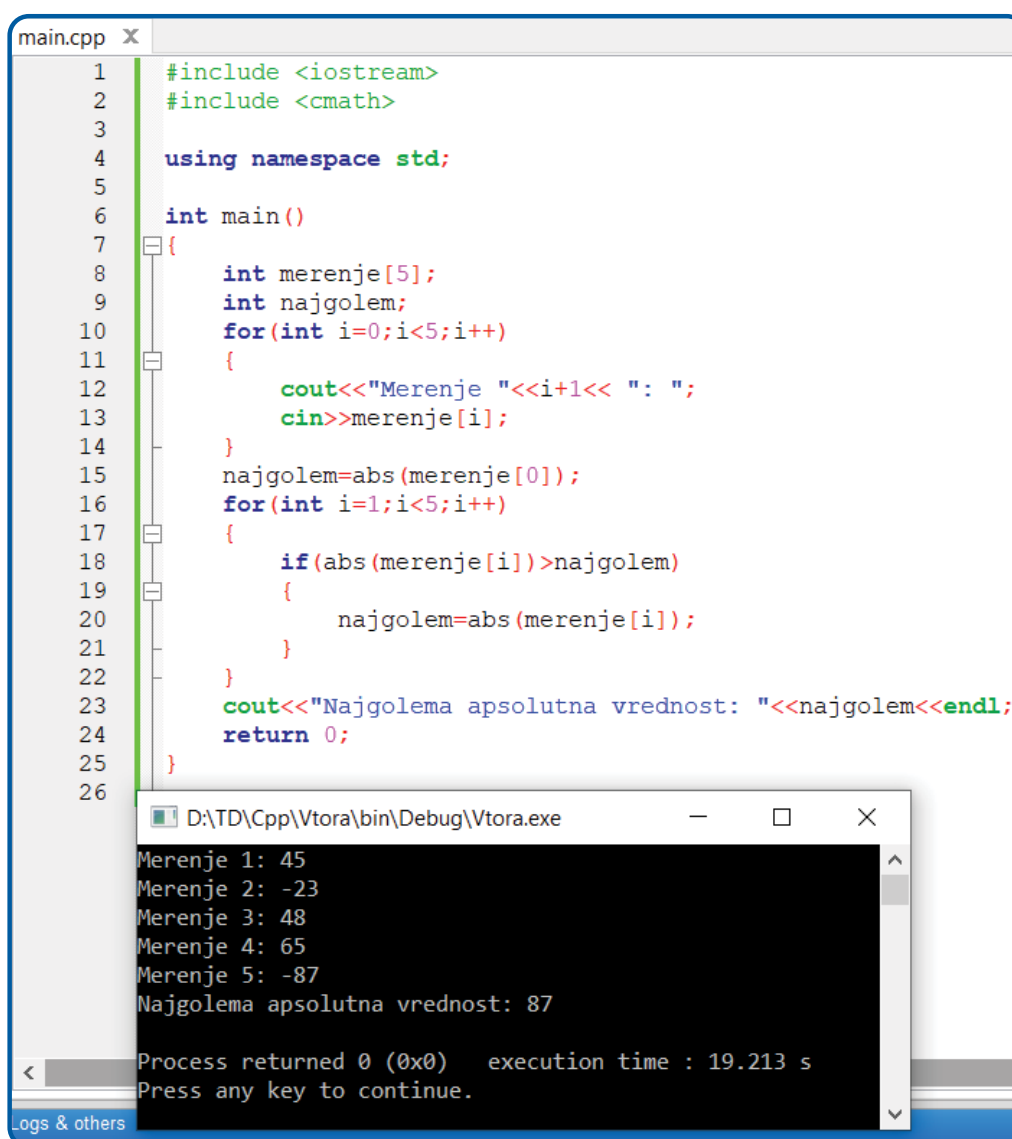
Kvadrat: 16
Kvadrat: 4
Kvadrat: 9
Kvadrat: 16
Kvadrat: 36
```

Во примерот секој елемент од низата поединечно се обработува. Со помош на функцијата `pow()` се пресметува квадратот на секоја внесена вредност.

При практичната обработка на податоци често се појавуваат и негативни вредности. Во одредени ситуации не е важен нивниот знак, туку само нивната големина. За таа цел се користи функцијата `abs()`, која ја определува апсолутната вредност на бројот. По обработката на сите елементи може да биде извршена и дополнителна анализа на добиените резултати.

Пример 3: Најголема апсолутна вредност

Во научна лабораторија се внесуваат пет мерења што можат да бидат позитивни или негативни. Да се определи најголемата апсолутна вредност.



```

main.cpp X
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main()
7  {
8      int merenje[5];
9      int najgolem;
10     for(int i=0;i<5;i++)
11     {
12         cout<<"Merenje "<<i+1<<" ": ";
13         cin>>merenje[i];
14     }
15     najgolem=abs(merenje[0]);
16     for(int i=1;i<5;i++)
17     {
18         if(abs(merenje[i])>najgolem)
19         {
20             najgolem=abs(merenje[i]);
21         }
22     }
23     cout<<"Najgolema apsolutna vrednost: "<<najgolem<<endl;
24     return 0;
25 }
26
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Merenje 1: 45
Merenje 2: -23
Merenje 3: 48
Merenje 4: 65
Merenje 5: -87
Najgolema apsolutna vrednost: 87

Process returned 0 (0x0)   execution time : 19.213 s
Press any key to continue.

```

Во примерот врз секој елемент најпрво се применува функцијата `abs()`. Потоа добиените вредности се споредуваат и се определува најголемата апсолутна вредност.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Зошто во програмата треба да биде вклучена библиотеката `cmath`?
2. Како може математичка функција да биде применета врз сите елементи на една низа?
3. Зошто е корисно математичките функции да се комбинираат со низи и циклуси



ЗАДАЧИ

1. Во геодетска служба се внесуваат пет растојанија. Напиши програма што ќе го пресметува квадратниот корен на секое растојание.
2. Во физичка лабораторија се внесуваат шест мерења. Напиши програма што ќе го пресметува квадратот на секоја вредност.
3. Во истражувачка станица се внесуваат пет позитивни и негативни мерења. Напиши програма што ќе ја определува најголемата апсолутна вредност.
4. Во астрономска опсерваторија се внесуваат седум растојанија. Напиши програма што ќе го пресметува кубот на секое растојание.
5. Во инженерска лабораторија се внесуваат шест мерења. Напиши програма што ќе ја пресметува апсолутната вредност на секое мерење и ќе ги прикажува добиените резултати.



ЗАПОМНИ ШТО НАУЧИ

- Функциите од библиотеката `cmath` може да се применат врз елементите на една низа.
- Со комбинирање на низи, циклуси и математички функции може да се решат различни практични задачи.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите информатички системи голем број податоци се обработуваат со примена на математички функции. Во науката може да се анализираат резултати од мерења, во инженерството различни физички величини, а во финансиските системи – различни статистички показатели. Иако современите програми обработуваат огромни количества информации, основната идеја останува иста – секој податок се обработува со соодветна математичка функција, а добиените резултати понатаму може да бидат анализирани и споредувани.

2.24



КОМБИНИРАНА ПРИМЕНА НА ФУНКЦИИ И ЕДНОДИМЕНЗИОНАЛНИ НИЗИ

ВЕЌЕ ЗНАЕШ ДА...

- користиш еднодимензионални низи;
- внесуваш и да обработуваш елементи;
- дефинираш и да повикуваш функции;
- користиш циклус for;
- комбинираш различни програмски конструкции.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува аргумент на функција?
2. Што претставува повик на функција?
3. Дали една функција може да биде повикана повеќепати во една програма?
4. Зошто при обработка на низи најчесто се користи циклус?
5. Каква предност има комбинирањето на функции и низи?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- применуваш функции врз елементите на низа;
- комбинираш циклуси и функции;
- обработуваш податоци со кориснички функции;
- користиш повратни вредности при работа со низи;
- решаваш практични програмски задачи.

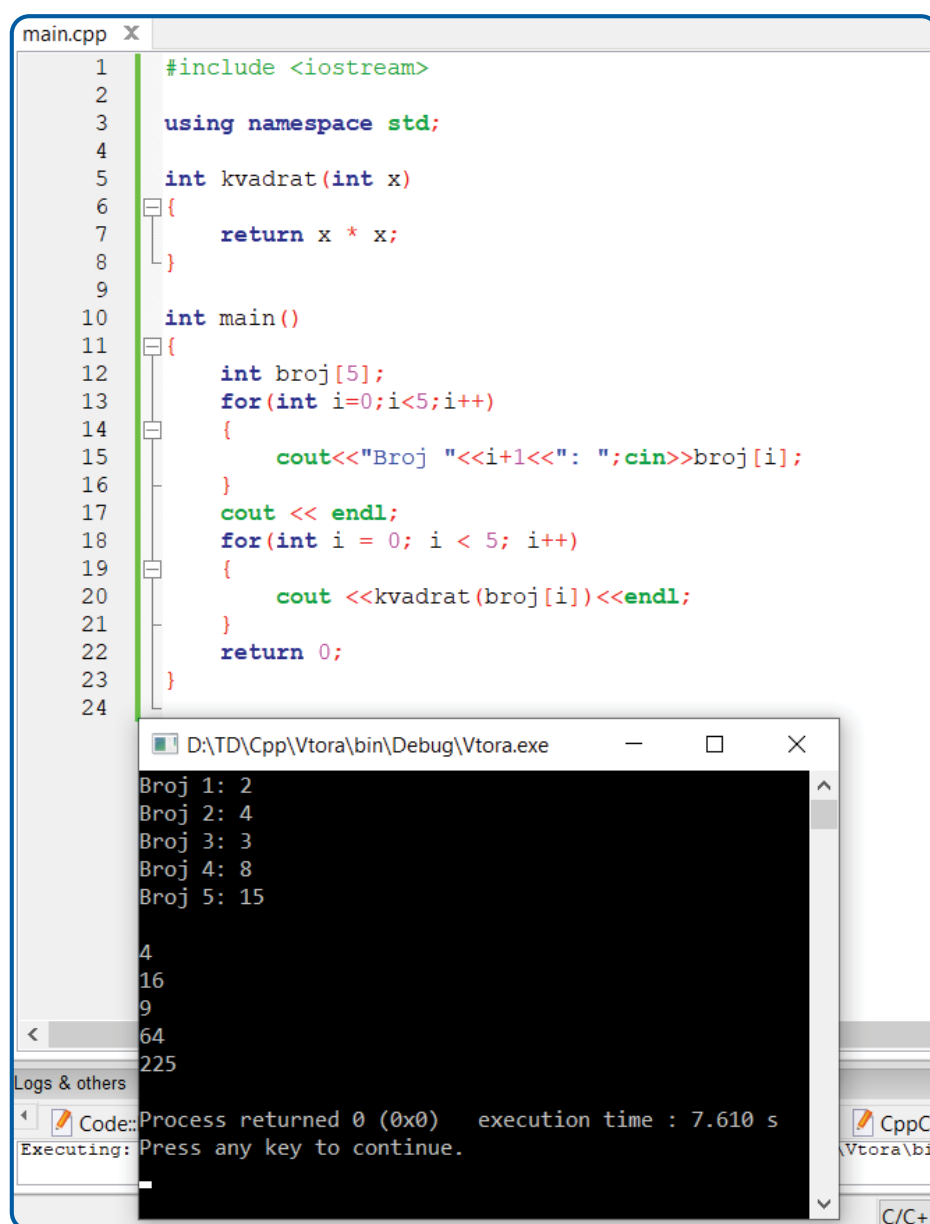
При обработка на податоците честопати постои потреба иста постапка да биде извршена врз повеќе елементи од една низа. Наместо истите програмски инструкции да бидат повторувани повеќепати, тие можат да бидат сместени во функција. Потоа функцијата последователно се повикува за секој елемент од низата.

Со комбинирање на функции и еднодимензионални низи програмите стануваат попрегледни и полесни за разбирање. Истата функција може да биде употребена за обработка на сите елементи, без потреба од повторно пишување на исти програмски инструкции.

При комбинирање на функции и низи најчесто функцијата обработува еден елемент, а циклусот овозможува таа да биде повикана за сите елементи од низата. На овој начин секоја вредност се обработува на ист начин, а програмата останува едноставна и прегледна. Ваквиот пристап има широка примена при решавање различни програмски задачи и претставува природно продолжение на изучувањето на функциите и еднодимензионалните низи.

Пример 1: Квадрат на елементите

Во истражувачка лабораторија се внесуваат пет вредности. Да се прикаже квадратот на секоја вредност со помош на функција.



```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int kvadrat(int x)
6  {
7      return x * x;
8  }
9
10 int main()
11 {
12     int broj[5];
13     for(int i=0;i<5;i++)
14     {
15         cout<<"Broj "<<i+1<<" : ";cin>>broj[i];
16     }
17     cout << endl;
18     for(int i = 0; i < 5; i++)
19     {
20         cout <<kvadrat(broj[i])<<endl;
21     }
22     return 0;
23 }
24
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Broj 1: 2
Broj 2: 4
Broj 3: 3
Broj 4: 8
Broj 5: 15

4
16
9
64
225

Code: Process returned 0 (0x0)   execution time : 7.610 s
Executing: Press any key to continue.
```


Во примерот функцијата обработува еден елемент, а циклусот овозможува истата функција да биде повикана за сите елементи од низата.

При комбинирање на функции и низи не е неопходно функцијата да пресметува само математичка вредност. Во многу ситуации потребно е да биде проверено дали одреден елемент исполнува зададен услов. Во такви случаи функцијата може да враќа логичка вредност. Ако е исполнет условот, функцијата враќа true, а во спротивно false. Со помош на циклус истата функција може да биде повикана за сите елементи од низата, а добиените резултати може дополнително да се обработуваат.

Пример 2: Проверка на парни броеви

Во истражувачки центар се внесуваат шест броеви. Да се определи колку од нив се парни со помош на функција.

```

main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  bool paren(int x)
6  {
7      if(x % 2 == 0)
8      {
9          return true;
10     }
11     return false;
12 }
13
14 int main()
15 {
16     int broj[6];
17     int vkupno = 0;
18     for(int i=0;i<6;i++)
19     {
20         cout<<"Broj "<<i+1<<": ";
21         cin >> broj[i];
22     }
23     for(int i = 0; i < 6; i++)
24     {
25         if(paren(broj[i]))
26         {
27             vkupno++;
28         }
29     }
30     cout<<"Parni broevi: "<<vkupno<<endl;
31     return 0;
32 }
33
D:\TD\Cpp\Vtora\bin\Debug
Broj 1: 2
Broj 2: 5
Broj 3: 45
Broj 4: 62
Broj 5: 188
Broj 6: 145
Parni broevi: 3
Process returned 0 (0x0)
Press any key to continue

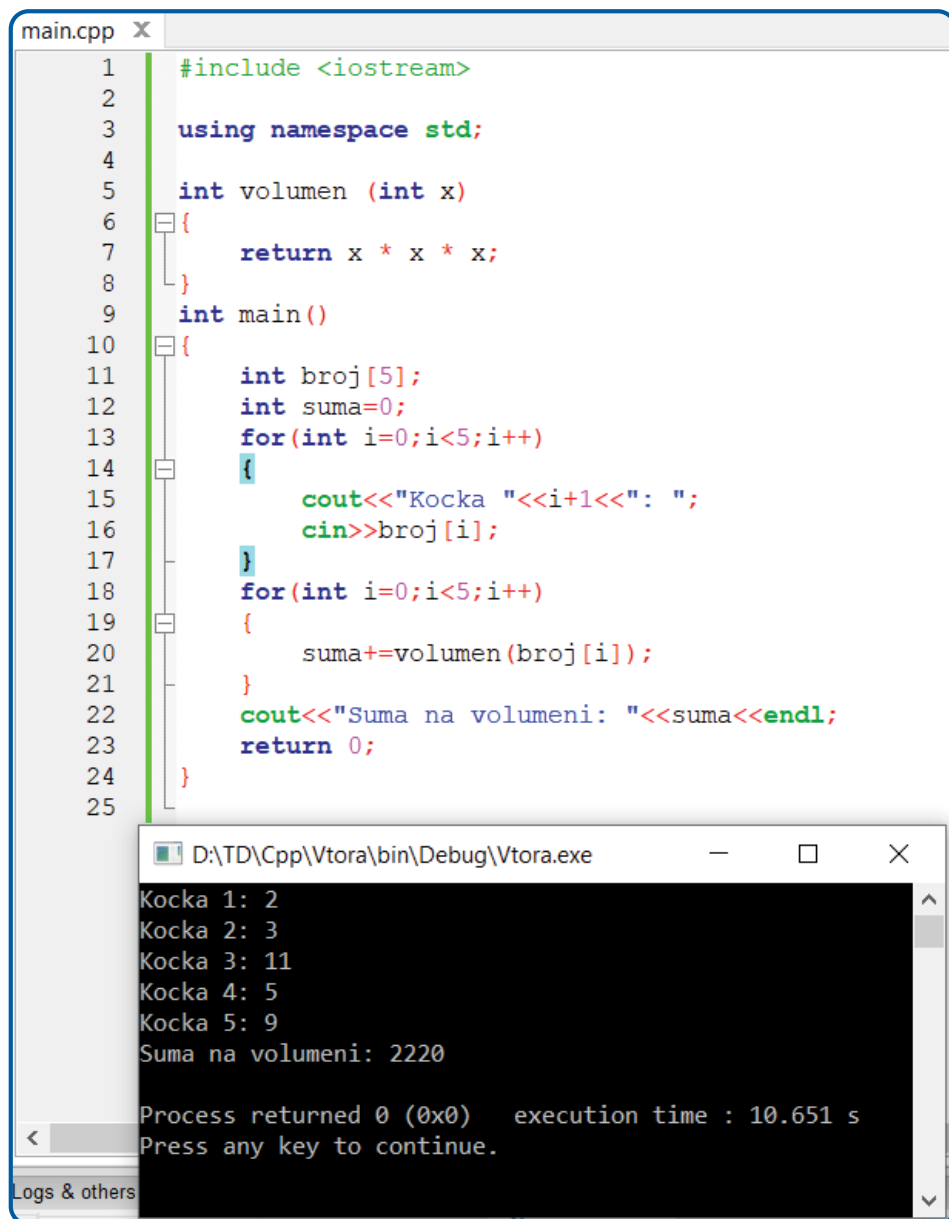
```

Во примерот функцијата проверува дали еден број е парен. Потоа истата функција се повикува за секој елемент од низата, а се пребројуваат сите броеви што го исполнуваат условот.

При работа со функции често има потреба одредена вредност да биде пресметана за секој елемент од низата, а потоа добиените резултати дополнително да бидат обработени. На овој начин може да се комбинираат различни програмски техники што претходно биле изучувани одделно. Благодарение на ваквиот пристап програмите стануваат поорганизирани и полесни за одржување.

Пример 3: Волумен на коцки и нивна сума

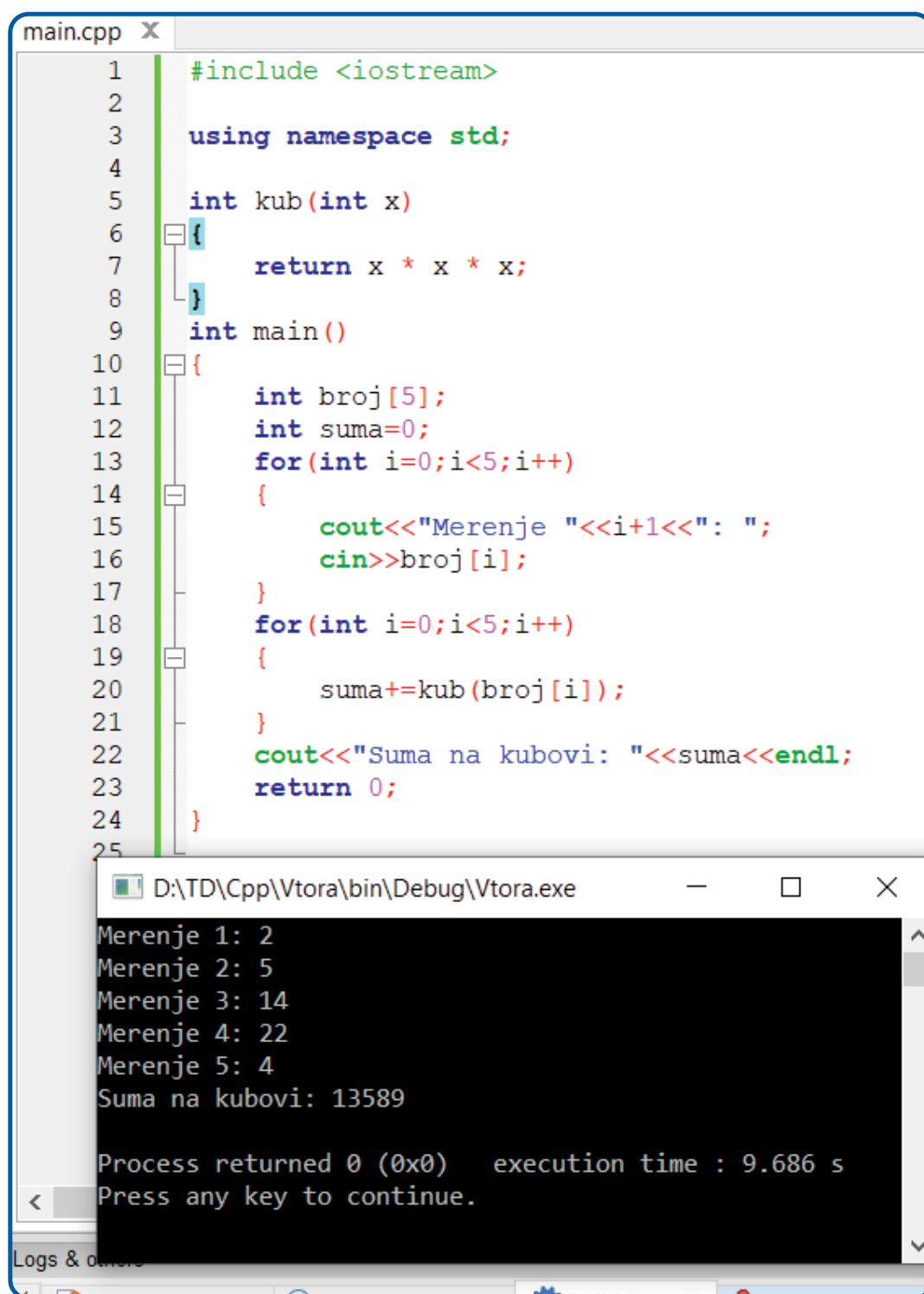
Во техничка лабораторија треба да се оптимизира просторот околу 5 објекти што ќе се испраќаат во вселената. Сите 5 објекти имаат форма на коцка. Потребни се пет мерења на волумен на тела во форма на коцка. Да се пресмета збирот на волумените на сите мерења со помош на функција.



```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int volumen (int x)
6  {
7      return x * x * x;
8  }
9
10 int main()
11 {
12     int broj[5];
13     int suma=0;
14     for(int i=0;i<5;i++)
15     {
16         cout<<"Kocka " <<i+1<<" : ";
17         cin>>broj[i];
18     }
19     for(int i=0;i<5;i++)
20     {
21         suma+=volumen(broj[i]);
22     }
23     cout<<"Suma na volumeni: " <<suma<<endl;
24     return 0;
25 }
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Kocka 1: 2
Kocka 2: 3
Kocka 3: 11
Kocka 4: 5
Kocka 5: 9
Suma na volumeni: 2220

Process returned 0 (0x0)   execution time : 10.651 s
Press any key to continue.
```



```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int kub(int x)
6  {
7      return x * x * x;
8  }
9  int main()
10 {
11     int broj[5];
12     int suma=0;
13     for(int i=0;i<5;i++)
14     {
15         cout<<"Merenje "<<i+1<<" : ";
16         cin>>broj[i];
17     }
18     for(int i=0;i<5;i++)
19     {
20         suma+=kub(broj[i]);
21     }
22     cout<<"Suma na kubovi: "<<suma<<endl;
23     return 0;
24 }
25
```

D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe

Merenje 1: 2
Merenje 2: 5
Merenje 3: 14
Merenje 4: 22
Merenje 5: 4
Suma na kubovi: 13589

Process returned 0 (0x0) execution time : 9.686 s
Press any key to continue.

Во примерот функцијата го пресметува волуменот кој се добива од вредноста на еден елемент, а циклусот овозможува таа да биде применета врз сите елементи од низата. Добиените резултати постепено се собираат и на крајот се прикажува нивниот збир.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ



ПРАШАЊА

1. Каква задача може да извршува функција што обработува еден елемент од низа?
2. Како овозможува циклусот иста функција да биде применета врз повеќе елементи?
3. Во кои ситуации функцијата може да враќа логичка вредност?
4. Зошто е корисно обработката на податоците да биде сместена во функција?
5. Кои се предностите од комбинирањето на функции и еднодимензионални низи?



ЗАДАЧИ

1. Во метеоролошка станица се внесуваат шест температури. Напиши програма во која ќе употребиш функција што ја пресметува тројната вредност (помножена со 3) на температурата и ќе ги прикажува резултатите.
2. Во центар за роботика се внесуваат пет резултати. Напиши програма во која ќе употребиш функција што проверува дали е бројот позитивен и ќе определи колку позитивни вредности се внесени.
3. Во научна лабораторија се внесуваат шест мерења. Напиши програма во која ќе употребиш функција што го пресметува остатокот при делење со бројот 5 и ќе ги прикажува добиените вредности.
4. Во спортски центар се внесуваат пет резултати. Напиши програма во која ќе употребиш функција што ја пресметува половината од секој резултат и ќе ги прикажува добиените вредности.
5. Во енергетски центар се внесуваат шест мерења. Напиши програма во која ќе употребиш функција што определува дали вредноста е поголема од 100 и ќе го пресмета бројот на такви мерења.



ЗАПОМНИ ШТО НАУЧИ

- Функција може да обработува поединечен елемент од низа.
- Истата функција може да биде повикана за сите елементи од низата.
- Функцијата може да враќа бројчена или логичка вредност.
- Циклусот овозможува функцијата да биде применета врз сите елементи.
- Комбинирањето на функции и низи овозможува програмите да бидат попрегледни и полесни за одржување.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современото програмирање функциите претставуваат еден од најважните начини за организирање на програмскиот код. Наместо истите инструкции да бидат повторувани на повеќе места, тие се сместуваат во функции што може повеќепати да бидат повикувани. Кога ваквиот пристап се комбинира со еднодимензионални низи, може едноставно да бидат обработени големи множества податоци. Овој начин на работа се применува при обработка на научни мерења, финансиски анализи, статистички податоци и многу други современи информатички системи.



ЗАДАЧИ ЗА ВЕЖБАЊЕ СО НИЗИ

Лесни задачи

1. Во планинарски дом во текот на една седмица се бележи бројот на посетители. Напиши програма што ќе ја пресмета вкупната посетеност.
2. Во расадник се евидентира висината на десет млади садници. Напиши програма што ќе ја определи просечната висина.
3. Во музеј се евидентира бројот на посетители во текот на осум последователни дена. Напиши програма што ќе го определи денот со најголема посетеност.
4. Во ветерна електрана се евидентира дневното производство на електрична енергија на девет генератори. Напиши програма што ќе определи кој генератор има најмало производство.
5. Во парк за диви животни се бележи бројот на родени младенчиња во седум различни живеалишта. Напиши програма што ќе определи во колку живеалишта се родени повеќе од три младенчиња.
6. Во библиотека пристигнуваат книги во различен број примероци. Во десет испораки се евидентира бројот на книги. Напиши програма што ќе провери дали некоја испорака содржи точно 100 книги.
7. Во фабрика за сокови се бележи бројот на произведени шишиња во текот на осум часа. Напиши програма што ќе ја определи разликата помеѓу најуспешниот и најслабиот час.
8. Во национален парк се бележи бројот на забележани елени во текот на девет набљудувања. Напиши програма што ќе определи во колку набљудувања биле забележани најмалку пет елени.
9. Во камп за астрономија се бележи бројот на забележани метеори во текот на десет ноќи. Напиши програма што ќе го пресмета вкупниот број на метеори забележани во парните ноќи.
10. Во аквариум се следи бројот на новородени риби во осум базени. Напиши програма што ќе определи колку базени имаат број на новородени риби поголем од просечниот број.

Средно тешки задачи

11. Во рудник се евидентираат површините на осум квадратни ископи. Напиши програма што ќе ја определи должината на страната на секој ископ.
12. Во расадник се бележат должините на страните на седум квадратни цветни леи. Напиши програма што ќе ја пресмета површината на секоја леа.

13. Во геолошка лабораторија се евидентираат позитивни и негативни поместувања на земјиштето во текот на девет мерења. Напиши програма што ќе определи кое поместување е најголемо, без разлика на насоката.
14. Во центар за спортски подготовки се следи растојанието што го истрчале шест спортисти. Да се пресмета квадратот на растојанието на секој спортист и да се прикажат добиените вредности.
15. Во архитектонско студио се проектираат кружни фонтани. Познати се радиусите на осум фонтани. Напиши програма што ќе ја пресмета површината на секоја фонтана.
16. Во градежна компанија се внесуваат должините на страните на седум коцки што ќе се користат како декоративни елементи. Напиши програма што ќе го пресмета волуменот на секоја коцка.
17. Во астрономска опсерваторија се евидентираат позитивни и негативни отклонувања на телескопот. Напиши програма што ќе ја пресмета вкупната големина на сите отклонувања без да ја земе предвид нивната насока.
18. Во научна лабораторија се внесуваат површини на квадратни примероци. Напиши програма што ќе определи кој примерок има најдолга страна.
19. Во центар за обновлива енергија се познати радиусите на шест кружни соларни платформи. Напиши програма што ќе ја пресмета вкупната површина на сите платформи.
20. Во геодетска служба се евидентираат позитивни и негативни грешки при мерење. Напиши програма што ќе ја пресмета просечната големина на грешките, без разлика дали се позитивни или негативни.

Тешки задачи

21. Во пчеларски центар секоја кошница добива оценка за продуктивност. Кошница се смета за успешна ако произведе најмалку 25 kg мед. Во текот на сезоната се евидентира производството на осум кошници. Напиши програма што ќе определи колку успешни кошници има.
22. Во планинарски клуб се бележат височините на десет врвови. Врв се смета за висок ако неговата надморска височина е најмалку 2000 метри. Напиши програма што ќе определи колку високи врвови се евидентирани.
23. Во аквариум се следи бројот на новородени риби во девет базени. Базен се смета за успешен ако во него се родени повеќе од 50 риби. Напиши програма што ќе го определи бројот на успешни базени.
24. Во астрономска опсерваторија се бележат растојанијата на седум небесни тела. Небесно тело се смета за блиско ако растојанието е помало од однапред зададена вредност. Напиши програма што ќе го определи бројот на блиски тела.
25. Во национален парк се бележи староста на десет дрвја. Дрво се смета за вековно ако неговата старост е најмалку 100 години. Напиши програма што ќе определи колку вековни дрвја има.
26. Во центар за спасување животни се евидентира телесната маса на осум животни. За секое животно треба да биде определено дали припаѓа во групата на лесни или тешки животни според однапред зададена граница. Напиши програма што ќе го определи бројот на тешки животни.
27. Во спортски камп се бележат должините на скоковите на девет натпреварувачи. Скок се смета за успешен ако е подолг од зададената граница. Напиши програма што ќе определи колку успешни скокови се реализирани.
28. Во ветерен парк се бележи производството на седум турбини. Турбина се смета за ефикасна ако произведе повеќе од зададената количина енергија. Напиши програма што ќе го определи бројот на ефикасни турбини.

29. Во археолошка експедиција се евидентира староста на осум пронајдени предмети. Предмет се смета за историски значаен ако неговата старост е поголема од однапред определена вредност. Напиши програма што ќе го определи бројот на значајни предмети.
30. Во центар за обновлива енергија се следи дневното производство на девет соларни системи. Систем се смета за успешен ако произведе повеќе од зададената количина енергија. Напиши програма што ќе определи колку успешни системи има.
31. Во мото клуб се бележат поминатите километри на осум возачи. Возач добива авантуристички статус ако поминал најмалку 300 km. Напиши програма во која обработката за еден возач ќе биде издвоена во функција, а програмата ќе определи колку возачи добиле авантуристички статус.
32. Во пчеларник се следи количината мед од седум кошници. Кошница се смета за продуктивна ако произведе најмалку 25 kg мед. Напиши програма во која проверката за една кошница ќе биде реализирана со функција, а податоците ќе бидат обработени преку низа.
33. Во вселенска мисија се бележат големините на девет астероиди. Астероид се смета за ризичен ако неговата големина е поголема од зададена граница. Напиши програма што ќе ги обработи сите астероиди во низа и со функција ќе утврдува дали некој астероид е ризичен.
34. Во археолошка експедиција се внесува староста на осум предмети. Предмет се смета за значаен ако е постар од 1000 години. Напиши програма во која функција ќе ја одредува значајноста на еден предмет, а програмата ќе го прикаже бројот на значајни предмети.
35. Во фабрика се бележи бројот на изработени делови од шест работи. Робот се смета за ефикасен ако изработил повеќе од 120 делови. Напиши програма во која функција ќе ја проверува ефикасноста на еден робот, а низата ќе ги чува резултатите од сите работи.
36. Во центар за спасување се внесува телесната маса на седум животни. Животно се смета за тешко ако масата е поголема од однапред зададена граница. Напиши програма во која функција ќе ја врши проверката за едно животно, а програмата ќе пресмета колку животни се тешки.
37. Во паметен град се следи дневната потрошувачка на електрична енергија на осум објекти. Објект се смета за голем потрошувач ако е потрошувачката над 500 единици. Напиши програма во која функција ќе определува дали еден објект е голем потрошувач.
38. Во спортски камп се внесуваат резултатите од девет скокови во далечина. Скок се смета за успешен ако е подолг од зададена граница. Напиши програма во која функција ќе проверува дали еден скок е успешен, а програмата ќе го прикаже бројот на успешни скокови.
39. Еколошка патрола бележи количина собран отпад на осум локации. Локација се смета за критична ако е количината на отпадот поголема од зададена граница. Напиши програма во која функција ќе ја проверува состојбата на една локација, а програмата ќе определи колку критични локации има.
40. Во подводна мисија се внесуваат длабочините на седум нуркања. Нуркање се смета за длабоко ако е длабочината поголема од 40 метри. Напиши програма во која функција ќе проверува дали е некое нуркање длабоко, а низата ќе ги чува сите мерења.

За темата

Во современото општество секојдневно се создаваат и се обработуваат големи количества податоци. Податоци се користат во училиштата, банките, болниците, библиотеците, продавниците, државните институции и во многу други области. За нивно ефикасно чување, организирање и обработка се применуваат бази на податоци, кои овозможуваат брзо пронаоѓање, ажурирање и управување со информациите. Благодарение на нивната примена, работата на различни организации станува поефикасна, а пристапот до потребните информации е побрз и посигурен.

Во оваа тематска целина се обработуваат основните поими поврзани со базите на податоци и системите за управување со базите на податоци. Посебно внимание се посветува на организирањето на податоците во табели, записите и полињата, воспоставувањето врски меѓу податоците, како и на пребарувањето, внесувањето и ажурирањето на информациите. Преку практични активности се стекнуваат основни знаења и вештини за користење на бази на податоци во различни области од секојдневниот живот и во современото работење.

НОВИ ПОИМИ

- база на податоци;
- систем за управување со база на податоци;
- примарен клуч;
- врска;
- пребарување;
- ажурирање;
- сортирање;
- филтрирање;
- прашање (query);
- извештај.

ВЕЌЕ ЗНАЕШ ДА...

- внесуваш и да уредуваш податоци;
- организираш информации во табели;
- користиш различни компјутерски програми;
- пребаруваш информации;
- анализираш и да споредуваш податоци;
- применуваш дигитални технологии при решавање практични задачи;
- користиш информатички алатки за обработка на информации.



TEMA 3





3.1

ОСНОВИ НА БАЗИТЕ НА ПОДАТОЦИ

ВЕЌЕ ЗНАЕШ ДА...

- користиш различни компјутерски програми;
- внесуваш и да обработуваш податоци;
- организираш информации во табеларен облик;
- пребаруваш информации на интернет;
- користиш различни дигитални услуги;
- анализираш и да споредуваш податоци.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Како банките ги чуваат податоците за своите корисници?
2. Како училиштето води евиденција за учениците и наставниците?
3. Како знаат интернет-продавниците кои производи се достапни?
4. Како може илјадници корисници истовремено да користат исти информации?
5. Зошто е важно податоците лесно да се пронаоѓаат и да се ажурираат?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што претставува база на податоци;
- наведуваш примери за примена на базите на податоци;
- објаснуваш зошто се користат базите на податоци;
- ги анализираш предностите од нивната примена;
- го препознаваш нивното значење во современите информатички системи.

Современото општество секојдневно создава огромни количества информации. Секое купување во продавница, секое подигнување пари од банкомат, секое пребарување на интернет и секоја порака испратена преку социјалните мрежи е податок што треба да се зачува и да се обработи. Покрај поединците, големи количества информации создаваат училиштата, универзитетите, банките, болниците, државните институции и многу други организации. За нивното правилно функционирање потребно е податоците да бидат достапни во секој момент и да може брзо да се пронајдат и да се обработат.

Со развојот на информатичките технологии значително се зголемени и барањата за управување со информациите. Не е доволно податоците само да бидат зачувани, ами тие треба да бидат сигурни, точни и лесно достапни за корисниците што имаат право да ги користат. Дополнително, потребно е новите информации лесно да се додаваат, постојните да се изменуваат, а непотребните податоци да се отстрануваат. Поради овие причини, современите информатички системи користат специјални решенија за организирање и управување со големи количества податоци.

3.1.1 ОД ДАТОТЕКИ ДО СОВРЕМЕНИ СИСТЕМИ

Во почетниот развој на информатичките системи, податоците најчесто се чувале во посебни датотеки. Секоја програма користела сопствен начин за организирање на информациите, а податоците што биле потребни за работа се запишувале во различни документи. Ваквиот начин на работа бил сосема доволен додека била релативно мала количината на информации и додека информатичките системи ги користеле ограничен број корисници.

Со текот на времето, бројот на податоците значително се зголемил. Истите информации почнале да се користат во повеќе различни програми, а честопати било потребно некој податок да се внесе на повеќе места. На пример, ако одреден корисник ја промени адресата или телефонскиот број, промената требало да се изврши во сите датотеки каде што се наоѓале тие информации. Доколку некоја датотека не била ажурирана, можело да се појават различни вредности за истиот податок, што доведувало до грешки и несигурност во работата.

Проблем претставувало и пребарувањето на информациите. Со зголемувањето на бројот на податоците, нивното пронаоѓање барало сè повеќе време и ресурси. Дополнително, честото повторување на истите информации непотребно го зголемувало просторот за складирање и ја отежнувало нивната обработка. Одржувањето на ваквите системи станувало сè посложено и поскапо.

Развојот на информатичките технологии наметнал потреба од поинаков начин на организирање на податоците. Наместо да бидат распределени во голем број независни датотеки, започнало нивно организирање во единствени системи што овозможуваат централизирано чување, полесно ажурирање и побрзо пребарување на информациите. Ваквиот пристап претставува основа за развојот на современите бази на податоци.



3.1.2 ШТО ПРЕТСТАВУВА БАЗА НА ПОДАТОЦИ?

Базата на податоци претставува организирана збирка на податоци што овозможува нивно сигурно чување, лесно пребарување и ефикасна обработка. Нејзината основна задача е информациите да бидат достапни тогаш кога се потребни и да овозможи нивно едноставно додавање, менување и користење. За разлика од класичното чување на податоците во независни датотеки, базите на податоци овозможуваат информациите да бидат организирани на начин што ја подобрува нивната точност и достапност.

Една од најважните предности на базите на податоци е можноста истите информации да ги користат повеќе корисници и различни информатички системи. Доколку се додаде нов податок или ако се измени постоечка информација, промената веднаш станува достапна за сите корисници што имаат право на пристап. На овој начин се избегнува непотребното повторување на податоците и се намалува можноста за појава на грешки.

Базите на податоци не се користат само за чување информации. Тие овозможуваат и нивно пребарување, споредување, сортирање и анализа. Благодарение на овие можности, големи количества податоци може брзо да се обработуваат и да се користат за донесување различни одлуки. Поради тоа, базите на податоци претставуваат еден од најважните делови на современите информатички системи.

3.1.3 ПРИМЕНА НА БАЗИТЕ НА ПОДАТОЦИ

Базите на податоци денес се применуваат во речиси сите области на човековата дејност. Во образованието се користат за водење евиденција за учениците, за наставниците, предметите и за наставните активности. Во здравствените установи се применуваат за организирање на медицинските информации и податоците за пациентите, додека во банките овозможуваат сигурно управување со финансиските услуги и со трансакциите.

Базите на податоци имаат голема примена и во современите информатички системи. Тие се користат во интернет-продавниците, на социјалните мрежи, системите за електронска пошта, платформите за учење на далечина и за многу други дигитални услуги. Секој пат кога корисникот се најавува на некоја веб-страница, пребарува информација или извршува електронско плаќање, во заднината најчесто работи некоја база на податоци.

Развојот на облак-технологиите, мобилните апликации и вештачката интелигенција дополнително ја зголемува важноста на базите на податоци. Современите информатички системи обработуваат милијарди податоци во секунда, при што нивната сигурност, точноста и брзината претставуваат клучни фактори за успешно функционирање на дигиталните услуги што секојдневно ги користиме.



3.1.4 ПРЕДНОСТИ НА БАЗИТЕ НА ПОДАТОЦИ

Примената на базите на податоци обезбедува многубројни предности во однос на класичниот начин на чување информации. Податоците се организирани на систематски начин, што овозможува нивно побрзо пронаоѓање и полесно ажурирање. Истовремено, се намалува можноста за појава на грешки и непотребно повторување на информациите.

Друга важна предност е можноста повеќе корисници истовремено да работат со истите податоци. Додека едни корисници внесуваат нови информации, други можат да ги пребаруваат или да ги анализираат постојните податоци. Ова е особено важно кај современите информатички системи што опслужуваат голем број корисници.

Базите на податоци овозможуваат и подобра сигурност на информациите. Пристапот до податоците може да биде ограничен според потребите на корисниците, а современите системи обезбедуваат механизми за заштита и зачувување на информациите. Благодарение на овие карактеристики, базите на податоци претставуваат незаменлив дел од современите информатички технологии и затоа нивната примена постојано се проширува.

Да се разгледа едно средно училиште во кое во секоја учебна година се запишуваат нови ученици. Покрај личните податоци на учениците, потребно е да се евидентираат паралелките, наставниците, наставните предмети, распоредот на часовите, успехот и многу други информации. Во текот на годината некои податоци се менуваат, се запишуваат нови ученици, а други го завршуваат своето образование. Овие внесени податоци треба да бидат достапни во иднина. Секој што го завршил образованието по извесно време треба да го приложи овој податок во идни систе-



ми или компании. Ако се изгуби некој документ, повторно треба да има можност да биде издаден, без оглед на тоа кога е завршено образованието.

Да се претпостави дека сите овие информации се чуваат во голем број различни документи. Ако е потребно да се пронајде телефонскиот број на некој ученик, списокот на ученици во одредена паралелка или податок за определен наставник, ќе биде потребно пребарување низ повеќе различни документи. Освен тоа, секоја промена ќе треба да се направи на повеќе места, што ја зголемува можноста да се појават грешки.

Со користење на база на податоци, сите информации може да бидат организирани во единствен систем. Новите податоци лесно се внесуваат, постојните информации се ажурираат, а потребните податоци може брзо да се пронајдат. На овој начин значително се подобрува организацијата на работата и се намалува времето потребно за обработка на информациите.

Истата логика се применува и во многу други области. Болниците водат евиденција за пациентите, банките за своите корисници и за трансакциите, библиотеците за книгите, а современите интернет-услуги за милиони кориснички сметки и различни дигитални содржини. Без базите на податоци, функционирањето на овие системи би било многу потешко.

Базите на податоци овозможуваат информациите да бидат централизирани и достапни на корисниците што имаат потреба од нив. Промените може лесно да се извршат, а податоците брзо да се пронајдат и да се обработат. Благодарение на овие можности, базите на податоци претставуваат основен дел од современите информатички системи. Организираното чување на информациите не придонесува само за побрза работа туку и за подобра сигурност и точност на податоците. Современите информатички технологии се потпираат на вакви системи за да обезбедат непречено функционирање на различни дигитални услуги што секојдневно се користат.

Развојот на информатичките технологии доведе до создавање и обработка на огромни количества податоци. Класичниот начин на нивно чување во независни датотеки со текот на времето покажал многубројни недостатоци, како што се повторување на информациите, сложено ажурирање и тешкотии при пребарувањето на податоците. Поради тоа се појавила потреба за поефикасен начин за нивно организирање и управување.

Базите на податоци претставуваат современо решение за сигурно чување и обработка на информациите. Тие овозможуваат податоците лесно да се додаваат, да се изменуваат и да ги користат повеќе корисници, при што се подобруваат точноста и ефикасноста на работата. Благодарение на своите карактеристики, базите на податоци денес имаат широка примена во различни области на современото општество. Стекнувањето основни знаења за базите на податоци претставува важен дел од информатичката писменост. Разбирањето на нивното значење и начините на примена претставува основа за изучување на современите информатички системи и на нивната практична примена.





ДАЛИ ЗНАЕШ?

Дали знаеш дека секој ден во светот се создаваат огромни количества нови податоци? Голем дел од нив потекнуваат од користење на мобилни телефони, интернет-услуги, социјални мрежи и различни дигитални уреди. За нивно организирање и обработка се користат современи бази на податоци што можат да складираат и да управуваат со огромен број информации.



ЗАПОМНИ ШТО НАУЧИ

Базата на податоци претставува организирана збирка на податоци.

Современите информатички системи обработуваат големи количества информации.

Класичното чување на податоците во независни датотеки има одредени недостатоци.

Базите на податоци овозможуваат полесно чување, пребарување и ажурирање на информациите.

Тие имаат широка примена во различни области од современото општество.

Базите на податоци претставуваат важен дел од современите информатички технологии.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Современите информатички системи можат да обработуваат огромни количества податоци. Големите интернет-компани, научните центри и различни дигитални платформи секојдневно работат со милијарди информации што постојано се создаваат и се ажурираат. Управувањето со ваквите податоци претставува посебна област во информатиката, а развојот на нови технологии постојано овозможува побрза и поефикасна обработка на информациите.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

1. Одговори на прашањата.

- Што претставува база на податоци?
- Зошто се користат базите на податоци?
- Наведи три области во кои се применуваат бази на податоци.

2. Точно или неточно.

- Базите на податоци служат само за чување информации.
- Податоците во базата на податоци може да се ажурираат.
- Современите информатички системи не користат бази на податоци.

3. Размисли и образложи.

Кои би биле најголемите проблеми ако една банка, болница или училиште не користи база на податоци за организирање на своите информации?

3.2



КРЕИРАЊЕ ТАБЕЛИ ВО БАЗА НА ПОДАТОЦИ

ВЕЌЕ ЗНАЕШ ДА...

- објаснуваш што претставува база на податоци;
- наведуваш примери за примена на базите на податоци;
- објаснуваш зошто се користат базите на податоци;
- ги анализираш предностите од нивната примена;
- користиш табели за организирање информации.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

- Како најлесно може да се организираат голем број информации?
- Зошто податоците треба да бидат систематски подредени?
- Како се организираат информациите во една база на податоци?
- Што се случува ако не се организирани податоците на прегледен начин?
- Дали сите информации треба да бидат сместени во една табела?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што претставува релациона база на податоци;
- разликуваш табела, поле и запис;
- ја анализираш организацијата на податоците во табели;
- планираш структура на едноставна табела;
- користиш програми за работа со бази на податоци;
- креираш едноставна табела.

Во секојдневниот живот информациите често се организираат во табели. Училиштата водат табели за учениците, библиотеките за книгите, спортските клубови за натпреварите, а различни институции за своите корисници и за услугите. Табеларниот приказ овозможува информациите да бидат прегледни, лесно да се споредуваат и брзо да се пронаоѓаат потребните податоци.

Истата идеја се применува и кај базите на податоци. Иако современите информатички системи обработуваат огромни количества информации, нивната организација најчесто започнува со табели. Благодарение на ваквиот начин на работа, податоците може систематски да се внесуваат, да се ажурираат и да се користат за различни намени.

Интересно е што табеларниот приказ претставува врска меѓу секојдневната работа и базите на податоци. Голем број програми овозможуваат податоците да се извезат во табеларен облик, а истовремено табели создадени во различни програми може да се внесат во база на податоци. На пример, податоци организирани во табела може да се импортираат во база на податоци, а податоците од

базата повторно да се експортираат за понатамошна обработка и анализа. Оваа можност значително ја олеснува размената на информации меѓу различни информатички системи.

3.2.1 ТАБЕЛАТА КАКО НАЧИН ЗА ОРГАНИЗИРАЊЕ НА ПОДАТОЦИТЕ

Кога е потребно да се чуваат поголем број информации, потребно е тие да бидат организирани на логичен начин. Еден од наједноставните и најпрегледните начини за нивна организација е табеларниот приказ. Во табелата податоците се подредени така што лесно може да се внесуваат, да се прегледуваат и да се споредуваат.

Да се претпостави дека е потребно да се води евиденција за компјутерите во едно училиште. За секој компјутер треба да се знаат повеќе информации, како што се инвентарниот број, производителот, моделот, оперативниот систем и просторијата во која се користи. Организирањето на овие информации во табела овозможува секој податок да има свое место и значително го олеснува нивното користење.

Во современите бази на податоци табелите претставуваат основен начин за организирање на информациите. Тие овозможуваат податоците да бидат систематизирани и лесно достапни, а истовремено создаваат добра основа за понатамошна обработка и анализа.

Практичен пример

Да се разгледа следната табела.

Инвентарен број	Производител	Модел	Оперативен систем	Локација
1001	Lenovo	ThinkCentre	Windows 11	Кабинет 1
1002	Dell	OptiPlex	Windows 11	Кабинет 2
1003	HP	ProDesk	Linux	Кабинет 3
1004	Lenovo	ThinkCentre	Windows 11	Кабинет 4

Од табелата може лесно да се забележи дека секој компјутер е опишан со повеќе различни информации. Ако се набави нов компјутер, податоците за него може едноставно да се додадат. Ако се премести некој компјутер во друга просторија, потребно е да се промени само односната информација.

Ваквиот начин на организација овозможува податоците лесно да се прегледуваат и да се споредуваат. На пример, брзо може да се утврди колку компјутери користат одреден оперативен систем или кои уреди се наоѓаат во определена просторија.

Како да ја анализираме табелата?

При внимателно разгледување на претходниот пример може да се забележи дека е табелата составена од колони и редови. Секоја колона содржи определен вид информации, додека секој ред претставува комплетен опис на еден компјутер. Благодарение на ваквата организација, податоците се прегледни и лесно разбирливи.

Во базите на податоци колоните се нарекуваат полиња, бидејќи секоја од нив е наменета за определен вид информации. Редовите се нарекуваат записи и содржат податоци за еден конкретен објект. Ваквата организација претставува основа за уредување на податоците во базата на податоци.

Добрата организација на информациите овозможува нивно побрзо внесување, полесно одржување и поефикасно користење. Поради тоа, правилното креирање на табелите претставува еден од најважните чекори при создавање на база на податоци.

3.2.2 РЕЛАЦИОНА БАЗА НА ПОДАТОЦИ

Во практиката многу ретко сите податоци се сместуваат во една единствена табела. Доколку во една табела се запишуваат сите информации за учениците, наставниците, предметите, оценките, редоследот и отсуствата, таа би станала преголема, сложена за одржување и подложна на грешки. Поради тоа современите бази на податоци ги организираат информациите во повеќе меѓусебно поврзани табели. Овој начин на организација се нарекува релационен модел, а базите што го користат се нарекуваат релациони бази на податоци.

Основната идеја на релациониот модел е секоја табела да содржи податоци за една определена целина. На пример, во еден училиштен информациски систем може да постои посебна табела за учениците, посебна табела за наставниците, посебна табела за предметите и посебна табела за оценките. Наместо истите податоци да се повторуваат на повеќе места, табелите се поврзуваат меѓусебно преку заеднички полиња. На тој начин се намалува можноста за грешки, а податоците се чуваат поорганизирано и поефикасно.

Релационите бази на податоци денес се најраспространет вид бази на податоци. Тие се користат во образовни институции, во банки, здравствени установи, државни институции, интернет-продавници и во многубројни други информатички системи. Благодарение на нивната структура, големи количини податоци може брзо да се внесуваат, пребаруваат, ажурираат и да се анализираат.

Пример за организација на податоци во повеќе табели

Да претпоставиме дека е потребно да се води евиденција за учениците и за нивните оценки. Наместо сите информации да се сместат во една табела, може да се креираат две табели.

Табела: Ученици

ID	Име	Презиме
1	Ана	Петрова
2	Марко	Стојанов
3	Елена	Николовска

Табела: Оценки

ID	УченикID	Предмет	Оценка
1	1	Информатика	5
2	1	Математика	4
3	2	Информатика	5
4	3	Математика	5

Во првата табела се чуваат основните податоци за учениците, а во втората нивните оценки. Наместо секојпат да се повторуваат името и презимето на ученикот, се користи идентификациски број кој овозможува двете табели да се поврзат. На тој начин се намалува повторувањето на податоците и се подобрува организацијата на информациите.

Зошто се користат повеќе табели?

Една од најважните предности на релационите бази на податоци е намалувањето на дуплираните податоци. Доколку податоците за ученикот се чуваат само еднаш, секоја промена ќе треба да се направи на едно место. Ако ученикот го промени презимето или ако е потребно да се коригира некој податок, измената ќе биде едноставна и нема да постои опасност дел од записите да останат непроменети.

Организирањето на податоците во повеќе табели овозможува и побрзо пребарување на информациите. Наместо програмата да пребарува огромна табела со илјадници колони и редови, таа работи со помали и подобро организирани табели. Ова е особено важно кај современите информациски системи кои обработуваат милиони записи.

Дополнително, ваквиот начин на работа овозможува полесно проширување на базата на податоци. Ако се појави потреба во иднина за нови информации, може да се додаде нова табела без значително менување на постојната структура. Поради тоа релационите бази на податоци претставуваат стандард во современите информатички системи.

3.2.3 ПРОГРАМИ ЗА РАБОТА СО БАЗИ НА ПОДАТОЦИ

За креирање, организирање и обработка на податоци се користат специјализирани програми познати како системи за управување со бази на податоци. Овие програми обезбедуваат алатки за внесување, уредување, пребарување и прикажување на податоците, без да мора корисникот директно да се грижи за начинот на кој се чуваат информациите физички во компјутерот.

Во образованието и при изучување на основите на базите на податоци често се користат програмите Microsoft Access и LibreOffice Base. Тие обезбедуваат графичка работна околина во која корисникот може да креира табели, да внесува податоци, да воспоставува врски меѓу табелите и да изработува форми, прашалници и извештаи. Благодарение на визуелниот пристап, овие програми се погодни за изучување на основните принципи на работа со бази на податоци.

Иако постојат и многу други системи за управување со бази на податоци кои се користат во големи компании и институции, принципите на работа се слични. Независно од тоа дали се користи едноставна образовна алатка или комплексен корпоративен систем, податоците се организираат во табели, се воспоставуваат врски меѓу нив и се обезбедува можност за нивно пребарување и обработка.

Важно е да се разбере дека програмата претставува алатка за работа со базата на податоци, а не самата база на податоци. Базата на податоци е организирана збирка на информации, додека програмата обезбедува средства преку кои корисникот работи со тие информации.

Креирање табела

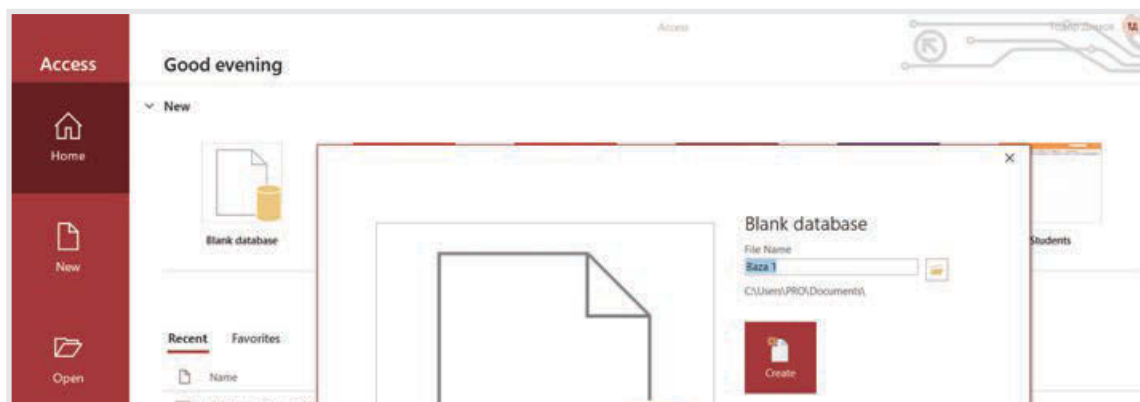
Пред да започне внесувањето на податоците, потребно е внимателно да се испланира кои информации ќе се чуваат во базата на податоци. Овој чекор е важен затоа што квалитетот на организацијата директно влијае врз ефикасноста на работата. Ако е табелата лошо осмислена, подоцна може да се појават проблеми при внесување, пребарување и ажурирање на податоците.

При креирање на табела најпрво се определуваат полињата што ќе ги содржи. Секое поле е наменето за одреден вид информации. На пример, доколку се креира табела за ученици, потребно е да се одредат полиња за името, презимето, датумот на раѓање, адресата, телефонскиот број и за други потребни податоци.

Добриот избор на полиња овозможува податоците да бидат организирани, прегледни и лесни за обработка. Поради тоа, креирањето на табелата не претставува само техничка постапка туку и процес на внимателно планирање на информациите што ќе се чуваат во базата на податоци.

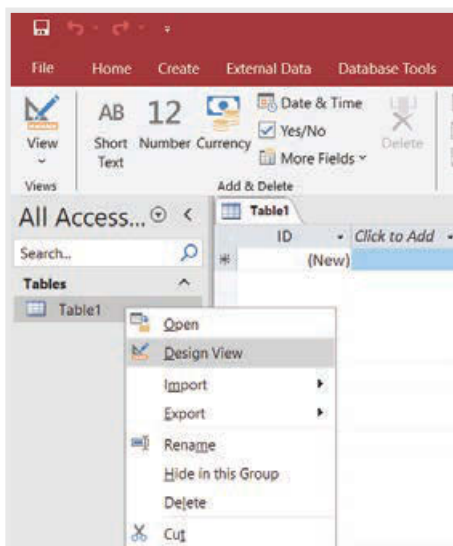
Пример: Планирање на табела за ученици

Во MS Access, откако ќе се стартува програмата, потребно е да се одбере празна база на податоци. Се доделува име на новата база и се одбира локацијата на која ќе биде сочувана комплетната база на податоци.



Да се претпостави дека училиштето сака да води електронска евиденција за учениците.

Пред да се креира табелата, потребно е да се размисли кои информации ќе бидат потребни за секој ученик. Може да се утврди дека е потребно да се чуваат името, презимето, датумот на раѓање, адресата, телефонскиот број и електронската адреса.



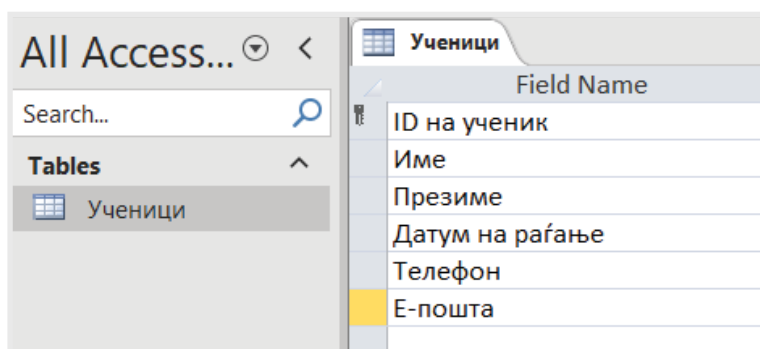
За да се постават полињата со потребните елементи се пристапува преку дизајнерскиот преглед на табелата.

Веднаш при изборот се зачувува табелата под име кое ќе ѝ биде доделено од страна на корисникот. Пожелно е да се доделуваат имиња кои ќе означуваат какви податоци би имало во самата табела.



Врз основа на оваа анализа може да се испланира следната структура:

ID на ученик, Име,	Презиме,	Датум на раѓање,	Адреса,	Телефон,	Е-пошта



Од примерот може да се забележи дека секоја колона претставува одреден вид податок кој ќе се внесува за сите ученици. Ваквото планирање претставува прв чекор во процесот на создавање табела.



Анализа на примерот

При креирањето на табелата не е доволно само да се внесат колони со различни имиња. Потребно е внимателно да се одреди кои информации се навистина потребни. Ако се додадат непотребни полиња, табелата ќе стане сложена и потешка за одржување. Од друга страна, доколку недостасуваат важни полиња, подоцна ќе биде потребно дополнително менување на структурата.

Планирањето на табелите претставува еден од најважните чекори при проектирање на база на податоци. Во практиката, пред да започне изработката на системот, често се врши детална анализа на информациите што треба да се чуваат и начинот на кој ќе се користат. На тој начин се обезбедува поефикасна организација и полесна работа со податоците.



ЗАКЛУЧОК:

Табелите претставуваат основен начин за организирање на податоците во релационите бази на податоци. Преку нив информациите стануваат прегледни, лесни за внесување и едноставни за обработка. Правилната организација на табелите претставува прв чекор кон создавање квалитетна база на податоци.

Секоја табела содржи полиња и записи што овозможуваат систематско организирање на информациите. Благодарение на ваквата структура, податоците може полесно да се пребаруваат, да се ажурираат и да се користат во различни информатички системи.



ДАЛИ ЗНАЕШ?

Дали знаеш дека најголем дел од податоците што ги користиме секојдневно се организирани во табели? Без разлика дали станува збор за училишна евиденција, за банкарски систем или интернет-продавница, основата на организацијата најчесто е табеларен приказ на информациите.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува релациона база на податоци?
2. Што претставува табела?
3. Која е улогата на полето во табелата?
4. Што претставува запис?
5. Зошто се организираат податоците во повеќе табели?
6. Кои програми може да се користат за работа со бази на податоци?
7. Зошто е важно планирањето на табелата пред нејзиното креирање?



ПРАКТИЧНИ ЗАДАЧИ

Задача 1

Направи план за табела во која ќе се чуваат податоци за книги во библиотека.

Задача 2

Одреди кои од следните информации претставуваат полиња:

- Име на ученик
- Марко Петров
- Телефон
- Кабинет 12
- Датум на раѓање

Задача 3

Во следната табела означи колку полиња и колку записи има.

Име	Презиме	Клас
Ана	Петрова	I-1
Марко	Стојанов	I-2
Елена	Николовска	I-1

Задача 4

Објасни зошто не е добра идеја сите податоци на едно училиште да се чуваат во една единствена табела.



ИСТРАЖУВАЧКА ЗАДАЧА

Истражи кои програми за работа со бази на податоци се најчесто користени денес. За секоја пронајди:

- име;
- производител;
- дали е бесплатна или е комерцијална;
- област на примена.



ЗАПОМНИ ШТО НАУЧИ

Табелата е основен елемент на релационата база на податоци.

Полето претставува колона во табелата.

Записот претставува ред во табелата.

Релационите бази на податоци содржат повеќе меѓусебно поврзани табели.

Податоците може да се увезуваат и да се извезуваат во табеларен облик.

Добро планирана табела овозможува полесна работа со податоците.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во големите компании и институции базите на податоци содржат илјадници табели и милиони записи. И покрај тоа, благодарение на добрата организација на табелите, потребните информации може да се пронајдат за дел од секундата.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

Заокружи го точниот одговор

Поле претставува:

- ред
- колона
- табела
- база на податоци

Запис претставува:

- колона
- поле
- ред
- база на податоци

Точно или неточно

- Една база на податоци може да содржи повеќе табели.
- Полето и записот се исти поими.
- Табелите се користат за организирање на податоците.



196

несоодветна вредност, може да дојде до грешки при обработката на податоците. Поради тоа, уште при креирањето на табелата се дефинира какви вредности ќе може да содржи секое поле.

Во современите системи за работа со бази на податоци постојат повеќе типови податоци. Иако називите може малку да се разликуваат од една програма до друга, нивната намена е слична. Најчесто се користат текстуални податоци, бројчени податоци, датуми и времиња, логички вредности и полиња за подолг текст.

3.3.1 НАЈЧЕСТО КОРИСТЕНИ ТИПОВИ ПОДАТОЦИ

Текстуални податоци

Текстуалниот тип на податок се користи за внесување зборови, имиња, адреси, телефонски броеви и други информации што не се користат во математички пресметки. Иако телефонскиот број содржи цифри, тој најчесто се чува како текст затоа што не се собира, одзема или множи.

Примери:

- Тодор Димов
- Свети Николе
- II-3
- 071123456

Бројчени податоци

Бројчениот тип се користи кога податоците треба математички да се обработуваат. Со нив може да се извршуваат пресметки, да се определуваат просеци, максимални и минимални вредности и други статистички анализи.

Примери:

- 5
- 18
- 125
- 98.5

Датум и време

Овој тип се користи за внесување датуми и временски информации. Благодарение на него може да се пресметува возраста на учениците, времетраењето на одредени активности или временската разлика помеѓу два настани.

Примери:

- 15.09.2025
- 21.05.2026
- 08:30

Логички податоци

Логичкиот тип овозможува внесување само на две вредности. Во различни програми тие може да бидат претставени како Да/Не, Вистина/Невистина или Вклучено/Исклучено.

Примери:

- Да
 - Не
- или
- True
 - False

Подолг текст

Понекогаш е потребно да се внесат подолги описи, забелешки или коментари. За вакви ситуации се користи тип на податок што овозможува складирање на поголемо количество текст.

Пример:

„Ученикот активно учествува во натпревари по информатика и има освоено повеќе награди на регионални и на државни натпревари.“

Практичен пример

Да се разгледа табела за ученици.

Поле	Тип на податок
ID на ученик	Број
Име	Текст
Презиме	Текст
Датум на раѓање	Датум/Време
Број на изостаноци	Број
Е-пошта	Текст
Активен ученик	Логички

Ученици	
Field Name	Data Type
ID на ученик	AutoNumber
Име	Short Text
Презиме	Short Text
Датум на раѓање	Date/Time
Телефон	Short Text
Број на изостаноци	Number
Е-пошта	Short Text
Активен ученик	Yes/No

Слика 1 Тип на податок

Од примерот може да се забележи дека секое поле има определен тип на податок. Благодарение на оваа организација, програмата знае какви вредности може да се внесуваат и како треба да се обработуваат. На пример, бројот на изостаноци може да се собира и да се анализира, додека кај името тоа нема смисла.

Зошто е важно правилно да се избере типот на податок?

Изборот на соодветен тип на податок влијае врз точноста и ефикасноста на работата со базата на податоци. Кога би се чувале сите информации како обичен текст, програмата не би можела правилно да извршува пресметки, да ги подредува податоците или да проверува дали се исправни внесените вредности.

Дополнително, правилниот избор придонесува за подобро искористување на меморијата и побрза обработка на информациите. Кај големите бази на податоци што содржат милиони записи, ваквите оптимизации имаат значително влијание врз работата на системот.

Со внимателно определување на типот на податок за секое поле се создава добра основа за сигурна и ефикасна база на податоци. Овој чекор претставува еден од најважните делови при креирањето на табелите.

Својства на полињата

При креирање табела, за секое поле не се определува само неговото име и типот на податок. Дополнително може да се постават различни својства што го определуваат начинот на внесување, прикажување и обработка на информациите. Благодарение на овие својства, податоците се внесуваат контролирано и се намалува можноста за појава на грешки.

Да претпоставиме дека во една табела постои поле за име на ученик. Ако не се постават никакви ограничувања, корисникот може да внесе празна вредност, премногу долг текст или случајни знаци што немаат никакво значење. Со користење на својствата на полињата може однапред да се определат правила што ќе овозможат поквалитетно внесување на податоците.

Својствата на полињата претставуваат важен механизам за контрола на информациите во базата на податоци. Тие овозможуваат стандардизација на внесувањето, подобра организација на податоците и поголема сигурност при нивната обработка.

Најчесто користени својства

Големина на поле

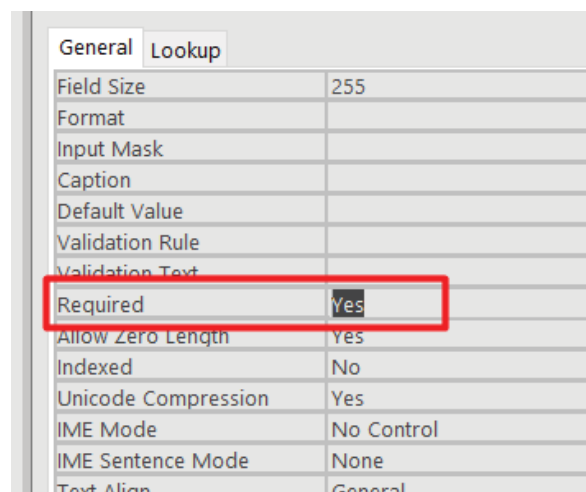
Кај текстуалните податоци често се определува максималниот број знаци што може да се внесат во полето. На пример, ако се знае дека шифрата на ученикот секогаш содржи најмногу десет знаци, нема потреба да се дозволува внесување на многу подолг текст.

Со ограничување на големината на полето се намалува можноста за внесување несоодветни податоци и се обезбедува поефикасно користење на меморискиот простор.

Задолжително поле

Понекогаш одредени информации мора задолжително да бидат внесени. На пример, во училишна евиденција не би било прифатливо да има ученик/чка без име или презиме. Во такви ситуации се користи својство што не дозволува зачувување на записот ако е полето празно.

Ова својство придонесува за подобрување на квалитетот на податоците и спречува појава на нецелосни записи. Во MS Access ова е прави во делот General од табелата, каде што една од многуте можности е да се одбере „Да“ во делот задолжително (Required)



Слика 2 Задолжително поле

General		Lookup	
Field Size	255		
Format			
Input Mask			
Caption			
Default Value	2026		
Validation Rule			
Validation Text			

Подразбрана вредност

Во одредени ситуации голем број записи имаат иста почетна вредност. Наместо корисникот постојано да ја внесува истата информација, може однапред да се постави подразбрана вредност.

На пример, доколку повеќето ученици припаѓаат на тековната учебна година, таа вредност може автоматски да се појавува при внесување на нов запис. Корисникот ќе ја промени само ако е потребно.

Слика 3 Вредност која ќе се појавува автоматски

Формат на податоците

Одредени информации треба да бидат прикажани на стандарден начин. Датумите, броевите и паричните вредности најчесто имаат однапред определен формат.

Благодарение на ова својство, сите податоци од ист тип се прикажуваат на единствен и прегледен начин, што ги подобрува и читливоста и разбирливоста на информациите.

Валидација на податоците

Во секоја база на податоци има можност корисниците по грешка да внесат неточни информации. Наместо број може да се внесе текст, наместо датум може да се внесе случајна вредност или може да се внесе податок што не ги задоволува правилата на системот. За да се спречат вакви ситуации се користи валидација на податоците.

Валидацијата претставува постапка со која се проверува дали внесената вредност ги исполнува однапред определените услови. Ако вредноста не ги исполнува правилата, системот го предупредува корисникот и не дозволува зачувување на неточните информации.

Доколку се дизајнира систем со кој треба се пресмета наплата на паркинг простор преку SMS порака, потребно е да се валидира дали внесената регистарска табличка е во вистински формат. Често пати луѓето прават грешки при внесувањето на бројот на табличките. Во полето „Input mask“ од „General“ картичката може да се постави валидација. Доколку очекуваме во систем да се внесуваат букви и броеви од форматот на регистарски таблички, потребно е да се внесе дека во неа има две букви, по кои следуваат четири броеви и на крајот две букви.

Притоа „L“ означува дека на таа позиција ќе има исклучиво Буква, додека „0“ означува позиција за цифра. Вака поставената маска ќе валидира дали е точно внесена регистарската табличка.

General		Lookup	
Field Size	Decimal		
Format			
Precision	18		
Scale	2		
Decimal Places	2		
Input Mask	LL 0000 LL		
Caption			
Default Value	0		
Validation Rule			

Овој механизам има големо значење во информатичките системи што обработуваат големи количества податоци. Дури и мала грешка при внесување може да предизвика проблеми при пребарување, анализа или изработка на извештаи. Поради тоа, современите бази на податоци користат различни техники за проверка на внесените информации.

Слика 4 поставување на маска на запис

Примери за валидација

Да се разгледаат следните ситуации:

Пример 1

Поле: Возраст

Правило: Вредноста мора да биде помеѓу 14 и 19.

✓ 16

✓ 18

✗ -5

✗ 120

Во овој случај системот ќе прифати само реални вредности за возраста на учениците.

General	Lookup
Field Size	255
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	>=14 And <=19
Validation Text	
Required	Yes

Слика 5 Поставување на правила за валидација

Вака треба да изгледа валидацијата на податоците.

Пример 2

Поле: Оценка

Правило: Дозволен се само вредности од 1 до 5.

✓ 1

✓ 5

✗ 7

✗ 0

Со ова правило се спречува внесување оценки што ги нема во системот за оценување.

General	Lookup
Field Size	255
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	>=1 And <=5
Validation Text	Дозволен се само вредности од 1 до 5.
Required	Yes
Allow Zero Length	Yes
Indexed	No

Слика 6 Постапување на правила и текст за валидација

Текстот за валидација дополнително објаснува што треба да содржи полето. Добра практика е да се поставува и текст за валидација кога се користи валидација. При алармирање од страна на програмата наместо да се слушне само звук, во овој случај се појавува и порака со грешката.

Пример 3

Поле: Е-пошта

Правило: Вредноста мора да го содржи знакот @.

✓ učenik@uciliste.mk

✗ ucenik@gmail.com

Валидацијата помага да се откријат очигледни грешки уште при внесувањето на податоците.

General	Lookup
Field Size	255
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	Like "*"@"
Validation Text	Вредноста мора да содржи знак @.
Required	Yes
Allow Zero Length	Yes
Indexed	No
Unicode Compression	Yes

Слика 7 Валидација со задолжителен знак

Свездичките пред и по знакот што се бара означуваат дека пред и по знакот задолжително мора да има текст.

Анализа на примерите

Од претходните примери може да се забележи дека валидацијата не служи за забавување на податоците, туку за заштита на базата на податоци од неточни информации. Колку што е поголем бројот на корисници што внесуваат податоци, толку е поголема потребата од автоматска контрола на вредностите.

Комбинацијата на типови податоци, својства на полињата и правила за валидација создава сигурна основа за работа со информациите. Благодарение на овие механизми, податоците стануваат поконзистентни, попречни и полесни за обработка.



ЗАКЛУЧОК:

Типот на податок и својствата на полето имаат големо влијание врз квалитетот на информациите во базата на податоци. Правилниот избор овозможува податоците полесно да се обработуваат, да се пребаруваат и да се анализираат.

Валидацијата и својствата на полињата претставуваат важни механизми за контрола на внесените информации. Тие помагаат да се намали можноста за грешки и обезбедуваат поголема сигурност и точност на податоците.



ДАЛИ ЗНАЕШ?

Дали знаеш дека компјутерот различно ги обработува текстот, броевите и датумите? Благодарение на типовите податоци, системот знае дали може да пресметува со одредена вредност или треба само да ја прикажува.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува тип на податок?
2. Кои типови податоци најчесто се користат?
3. Кога се користи текстуален тип?
4. Кога се користи бројчен тип?
5. Што претставува валидација?
6. Зошто се користат својства на полињата?
7. Што е подразбрана вредност?



ПРАКТИЧНИ ЗАДАЧИ

Задача 1

За секое поле одреди соодветен тип на податок:

Поле
Име
Датум на раѓање
Број на изостаноци
Активен ученик
Забелешка

Задача 2

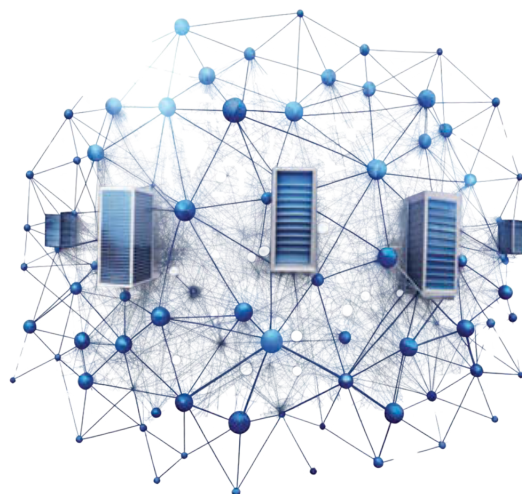
Наведи три полиња за кои би поставил задолжително внесување.

Задача 3

Наведи пример за поле што би имало подразбрана вредност.

Задача 4

Објасни зошто возраста треба да биде бројчен, а не текстуален податок.





ИСТРАЖУВАЧКА ЗАДАЧА

Истражи кои типови податоци ги користат Microsoft Access и LibreOffice Base. Направи споредба и пронајди кои типови се заеднички за двете програми.



ЗАПОМНИ ШТО НАУЧИ

Секое поле има определен тип на податок.

Типот на податок определува какви вредности може да се внесуваат.

Својствата на полињата овозможуваат контрола врз податоците.

Валидацијата спречува внесување на неточни вредности.

Правилниот избор на типови податоци ја подобрува ефикасноста на базата.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ



Во професионалните системи постојат десетици различни типови податоци. Некои од нив се користат за складирање фотографии, видеа, географски координати или сложени документи.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

Заокружи го точниот одговор

1. За датум на раѓање најсоодветен тип е:

- а) текст
- б) број
- в) датум/време
- г) логички

2. Валидацијата служи за:

- а) бришење записи
- б) проверка на внесените податоци
- в) печатење извештаи
- г) креирање табели

Точно или неточно

1. Во поле за возраст може да се внесува кој било текст.
2. Логичкиот тип најчесто има две можни вредности.
3. Валидацијата го подобрува квалитетот на податоците.





3.4

ПРИМАРЕН КЛУЧ И ПОВРЗУВАЊЕ НА ТАБЕЛИ

ВЕЌЕ ЗНАЕШ ДА...

- креираш табели во база на податоци;
- користиш различни типови податоци;
- применуваш својства на полиња;
- користиш правила за валидација;
- организираш информации во табеларен облик.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

- Што ќе се случи ако во два записи има исти податоци?
- Како може со сигурност да се разликуваат двајца ученици со исто име и презиме?
- Зошто секој запис треба да има единствен идентификатор?
- Како го пронаоѓа брзо компјутерот потребниот запис?
- Како може во различни табели да се користат исти информации без нивно повторување?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што претставува примарен клуч;
- препознаваш полиња погодни за примарен клуч;
- го анализираш значењето на единствената идентификација;
- ги објаснуваш предностите од користење на примарен клуч;
- применуваш примарен клуч при креирање табели;
- ја анализираш организацијата на податоците во релациона база.

При работа со поголем број записи често се појавува потреба секој запис да може недвосмислено да се препознае. Да претпоставиме дека во една табела се чуваат податоци за ученици. Можно е двајца ученици да имаат исто име и презиме, па само според тие податоци не може со сигурност да се утврди за кој ученик станува збор. Поради тоа, во базите на податоци се користи посебно поле што овозможува единствена идентификација на секој запис.

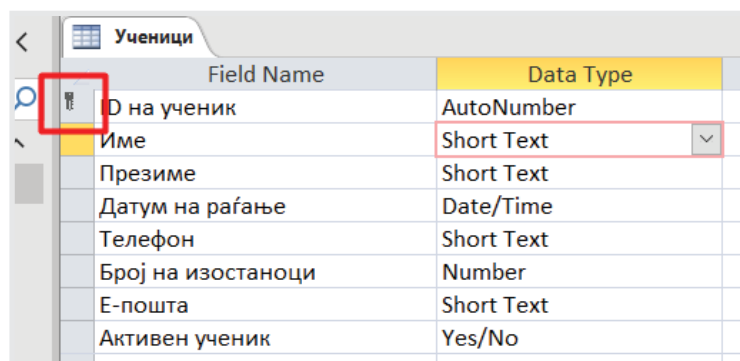
Поле то што содржи единствена вредност за секој запис се нарекува примарен клуч. Неговата основна задача е да обезбеди секој запис во табелата да биде единствен и лесно препознатлив. Вредноста на примарниот клуч не смее да се повторува кај два различни записи и не смее да остане празна.

Примарниот клуч има големо значење за правилното функционирање на базата на податоци. Освен што обезбедува единствена идентификација на записите, тој претставува основа за поврзување на табелите и за организирање на сложени информациски системи. Поради оваа причина, уште при креирањето на табелата потребно е внимателно да се избере кое поле ќе ја има оваа улога.

3.4.1 КАРАКТЕРИСТИКИ НА ПРИМАРНИОТ КЛУЧ

За да може да се користи едно поле како примарен клуч, потребно е да исполнува неколку услови.

Прво, вредноста мора да биде единствена за секој запис. Доколку две лица имаат ист идентификациски број, системот нема да може правилно да ги разликува. Второ, вредноста не смее да биде празна, затоа што секој запис мора да има свој единствен идентификатор. Трето, пожелно е вредноста да биде стабилна и ретко да се менува, бидејќи честите промени може да предизвикаат проблеми при организацијата на податоците.



Field Name	Data Type
ID на ученик	AutoNumber
Име	Short Text
Презиме	Short Text
Датум на раѓање	Date/Time
Телефон	Short Text
Број на изостаноци	Number
Е-пошта	Short Text
Активен ученик	Yes/No

Слика 1 Поставен примарен клуч

Во многу случаи како примарен клуч се користи специјално поле што автоматски добива нова вредност при внесување на секој нов запис. На овој начин се избегнува можноста корисникот случајно да внесе веќе постоечка вредност.

Пример

Да се разгледа следната табела.

Ученик ID	Име	Презиме
1	Ана	Петрова
2	Ajet	Rexhepi
3	Ана	Петрова
4	John	Wayne

Од табелата може да се забележи дека името и презимето „Ана Петрова“ се појавуваат двапати. Кога би се користеле само името и презимето за идентификација, би било тешко да се разликуваат двата записи.

Полеот УченикID содржи различна вредност за секој ученик и поради тоа може да се користи како примарен клуч. Без разлика колку ученици имаат исто име или презиме, нивниот идентификациски број ќе биде различен. Во овој случај како број за идентификација може да биде земен некој уникатен податок за ученикот, на пример, бројот во дневникот. Во еден клас може да има само еден ученик со број 3. Во образовниот систем се користи ЕИБУ (Единствен идентификациски број на ученикот), во Министерството за внатрешни работи се користи бројот на личната карта, во Министерството за надворешни работи бројот на пасошот, во Министерството за здравство се користи бројот на здравствената легитимација. Сите овие записи се поврзани со ЕМБГ (единствен матичен број на граѓанинот). Поради законските регулативи, не може секој да има допир со ЕМБГ на едно лице, па секоја институција креира некој уникатен начин како да ги разликува граѓаните со исти генералии. Доколку некоја организација или компанија им издава картички на своите клиенти и сака тие броеви

да се последователни, како тип на податок се користи Autonumber. Ова е многу корисна опција и кај фискалните апарати во продавниците, кои на дневно ниво издаваат и по стотици фискални сметки.

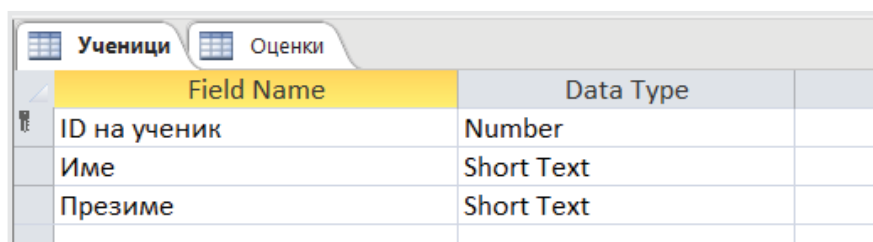
Анализа на примерот

Од претходниот пример може да се забележи дека примарниот клуч не мора да има значење за корисникот. Неговата улога е да обезбеди единственост на записите и правилна организација на податоците. Во практиката, корисниците најчесто ги гледаат имињата, адресите или другите информации, додека системот во заднината ги користи примарните клучеви за сигурно управување со податоците.

Употребата на примарен клуч значително ги олеснува пребарувањето, ажурирањето и одржувањето на информациите. Благодарение на него, секој запис може брзо да се пронајде и да се обработи без опасност од мешање со други записи што содржат слични податоци.

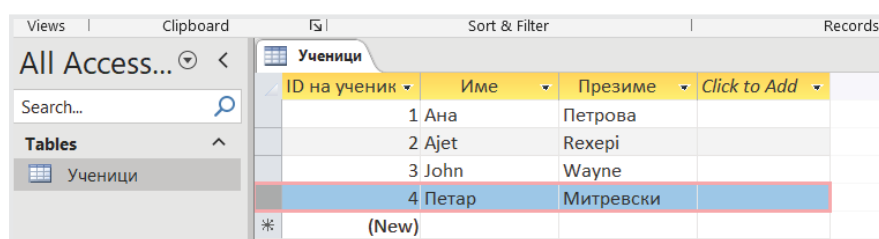
3.4.2 ПОВРЗУВАЊЕ НА ТАБЕЛИ

Во поголемите бази на податоци информациите најчесто се организираат во повеќе табели. За да може заеднички да се користат тие табели, потребно е меѓу нив да постои врска. Поврзувањето на табелите овозможува информациите да се организираат поефикасно и да се избегне непотребно повторување на податоците.



Field Name	Data Type
ID на ученик	Number
Име	Short Text
Презиме	Short Text

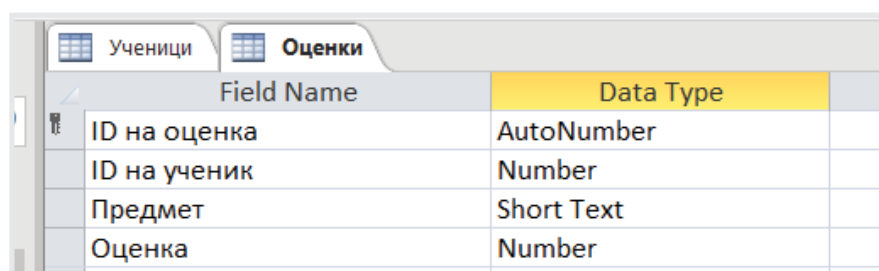
Слика 2 Уредување на табела со ученици



ID на ученик	Име	Презиме	Click to Add
1	Ана	Петрова	
2	Ајет	Рехери	
3	Џон	Вејн	
4	Петар	Митревски	
(New)			

Слика 3 Табела со ученици

Да претпоставиме дека има табела со ученици и посебна табела со оценки. Наместо во секој запис за оценка повторно да се запишуваат името и презимето на ученикот, може да се користи идентификацискиот број од табелата со ученици. На тој начин информациите се чуваат само еднаш, а сепак може лесно да се поврзат кога е потребно.



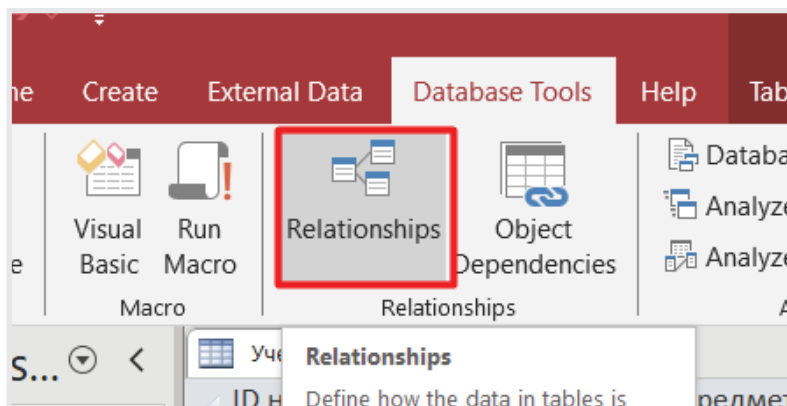
Field Name	Data Type
ID на оценка	AutoNumber
ID на ученик	Number
Предмет	Short Text
Оценка	Number

Слика 4 Уредување на табела со оценки

ID на оценка	ID на ученик	Предмет	Оценка
1	1	Информатика	4
2	1	Историја	5
3	2	Информатика	5
4	3	Информатика	3
5	1	Математика	4
(New)	0		0

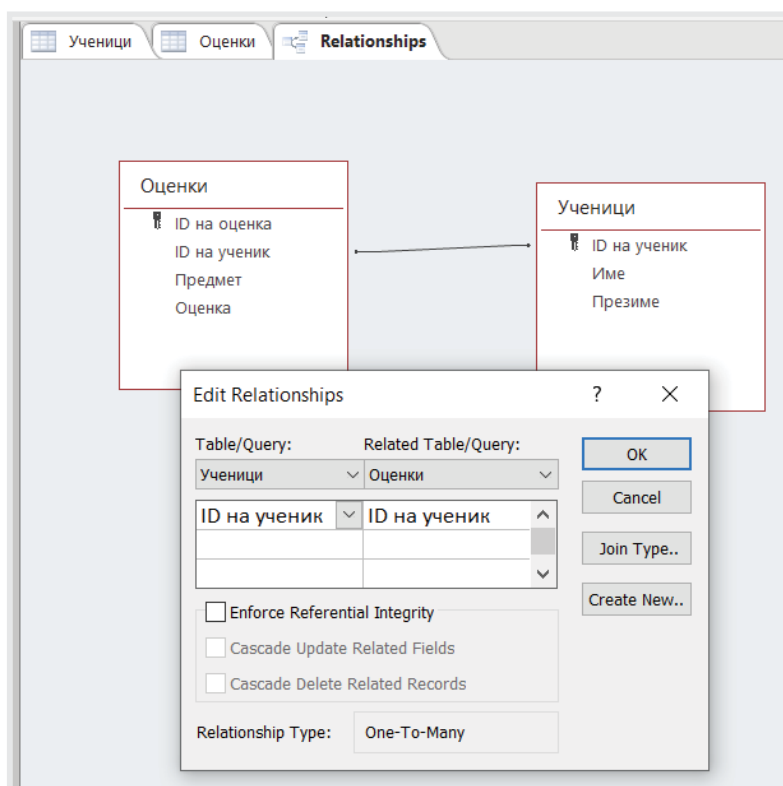
Слика 5 Пополнување на табела со оценки

Поврзувањето на табелите претставува една од најважните карактеристики на релационите бази на податоци. Благодарение на него, податоците остануваат организирани, а системот може ефикасно да работи и со голем број информации.



Слика 6 Поставување на врски во база на податоци

При поставување врски, потребно е елементите што се поврзуваат да бидат од ист тип. Во овој случај ќе се поврзат двете табели преку делот ID на ученик.



Слика 7 Поставување на врски меѓу табели

Уредување и ажурирање табели

Базите на податоци ретко остануваат непроменети. Со текот на времето се појавува потреба од додавање нови полиња, менување на постојните својства или внесување нови записи. Поради тоа, современите системи овозможуваат лесно уредување и ажурирање на табелите.

Уредувањето на табелата може да вклучува додавање ново поле, промена на името на постојно поле или промена на неговиот тип на податок. Ваквите промени најчесто се прават кога ќе се појават нови барања или кога е потребно подобро организирање на информациите.

Ажурирањето на табелите се однесува на промената на податоците. На пример, може да се промени телефонски број, адреса или друга информација што веќе ја има во базата на податоци. Редовното ажурирање е неопходно за податоците да останат точни и корисни.

Добро организираните табели, правилно избраните полиња и внимателното ажурирање на податоците претставуваат основа за сигурна и ефикасна база на податоци. Благодарение на овие активности, информациите остануваат точни, организирани и лесно достапни за корисниците.



ЗАКЛУЧОК:

Примарниот клуч претставува еден од најважните елементи на релационите бази на податоци. Неговата основна задача е да обезбеди секој запис да биде единствен и лесно препознатлив. Благодарение на примарниот клуч, податоците може сигурно да се организираат и да се одржуваат.

Користењето на примарен клуч ги подобрува ефикасноста на пребарувањето, ажурирањето и управувањето со информациите. Тој претставува основа за создавање поголеми и посложени бази на податоци.



ДАЛИ ЗНАЕШ?

Дали знаеш дека е можно две лица да имаат исто име и презиме, но не може да имаат ист матичен број? Сличен принцип се користи и во базите на податоци преку примарниот клуч.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува примарен клуч?
2. Кои услови треба да ги исполнува примарниот клуч?
3. Зошто вредностите на примарниот клуч не смеа да се повторуваат?
4. Дали примарниот клуч може да биде празен?
5. Кои се предностите од користење примарен клуч?
6. Како помага примарниот клуч при пребарување на податоците?



ПРАКТИЧНИ ЗАДАЧИ

Задача 1

Одреди кое поле би било најсоодветно за примарен клуч:

| Шифра | Производ | Количина |

Задача 2

Наведи три примери на полиња што може да се користат како примарен клуч.

Задача 3

Објасни зошто името и презимето не се секогаш добар избор за примарен клуч.

Задача 4

Размисли што би се случило ако два записи имаат иста вредност на примарниот клуч.



ИСТРАЖУВАЧКА ЗАДАЧА

Истражи како различни институции обезбедуваат единствена идентификација на лицата, производите или документите. Пронајди најмалку три примери и објасни како функционираат.



ЗАПОМНИ ШТО НАУЧИ

- Примарниот клуч овозможува единствена идентификација на записите.
- Неговите вредности не смее да се повторуваат.
- Не смее да содржи празни вредности.
- Ги олеснува пребарувањето и ажурирањето на податоците.
- Претставува важен елемент во организацијата на релационите бази на податоци.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ



Во големите бази на податоци примарните клучеви често се генерираат автоматски. На тој начин системот гарантира дека секој нов запис ќе добие единствен идентификациски број.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

Заокружи го точниот одговор

1. Примарниот клуч:

- а) може да се повторува
- б) мора да биде единствен
- в) не е важен
- г) служи за форматирање

2. Примарниот клуч најчесто се користи за:

- а) идентификација на записите
- б) печатење извештаи
- в) бришење табели
- г) внесување текст

Точно или неточно

1. Два различни записи може да имаат ист примарен клуч.
2. Примарниот клуч може да биде празен.
3. Примарниот клуч помага при организацијата на податоците.





3.5

ФОРМИ ЗА ВНЕСУВАЊЕ И ПРЕГЛЕД НА ПОДАТОЦИ

ВЕЌЕ ЗНАЕШ ДА...

- • креираш табели во база на податоци;
- • користиш различни типови податоци;
- • поставуваш својства на полиња;
- • користиш примарен клуч;
- • внесуваш и да ажурираш податоци во табели.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Дали е практично корисникот секогаш директно да внесува податоци во табелите?
2. Како може да се поедностави внесувањето на информации?
3. Дали сите корисници мора да ја гледаат структурата на табелата?
4. Како може да се направи попрегледен приказ на податоците?
5. Дали е можно внесувањето на податоци да изгледа како пополнување електронски образец?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што претставува форма;
- ги наведуваш предностите од користење форми;
- креираш едноставна форма;
- внесуваш податоци преку форма;
- уредуваш постојни записи;
- ја анализираш примената на формите во информатичките системи.

Во претходните лекции податоците беа внесувани директно во табелите. Иако е овој пристап едноставен, тој не е секогаш најпрактичен. Кај поголеми бази на податоци корисниците честопати немаат потреба да ја гледаат целата табела, ниту пак, да работат со сите полиња истовремено. Наместо тоа, многу попрактично е податоците да се внесуваат преку посебно подготвен екран што ги прикажува само потребните информации.

Токму поради оваа причина во базите на податоци се користат форми. Формите овозможуваат податоците да се внесуваат, да се прегледуваат и да се ажурираат на попрегледен и поконтролиран начин. Наместо корисникот да работи со редови и колони, тој работи со полиња, копчиња, листи и други елементи што потсетуваат на електронски образец.

Во секојдневниот живот често се среќаваат различни форми. Пополнувањето онлајн пријава, регистрација на корисничка сметка, внесување податоци за нарачка или електронско пријавување на испит претставуваат примери на форми што во заднина комуницираат со база на податоци.

Од табела до форма

Да се разгледа табела за ученици.

УченикID	Име	Презиме	Телефон	Е-пошта
1	Ана	Петрова	071123456	ana@email.com
2	Марко	Стојанов	075654321	marko@email.com

На прв поглед табелата ги содржи сите потребни информации. Меѓутоа, ако треба корисникот секојдневно да внесува нови ученици, ваквиот приказ не е секогаш најпрактичен. Особено ако табелата содржи десетина или дваесет полиња.

Поради тоа може да се креира форма што ќе ги прикажува податоците на поедноставен начин.

Предности на користењето форми

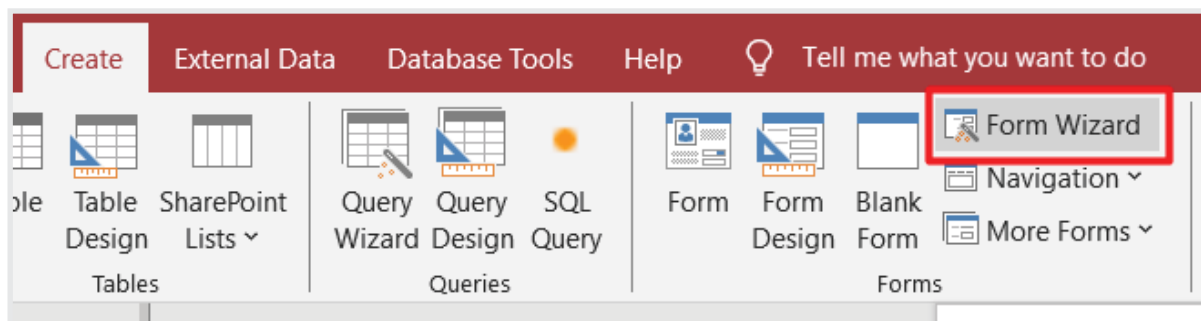
Формите значително ја подобруваат комуникацијата меѓу корисникот и базата на податоци. Тие овозможуваат поедноставен внес на информациите и ја намалуваат можноста за грешки. Наместо корисникот да се движи низ голема табела, вниманието е насочено само кон податоците што треба да се внесат.

Дополнителна предност е можноста за подобра организација на елементите. Поврзаните информации може да се групираат, да се користат копчиња за избор, паѓачки листи или полиња за означување. На овој начин формата станува попрегледна и полесна за користење.

Формите овозможуваат и ограничување на пристапот до одредени информации. Корисникот може да работи со формата без директно да ја отвора табелата. Ова придонесува за подобра безбедност и поголема контрола врз податоците.

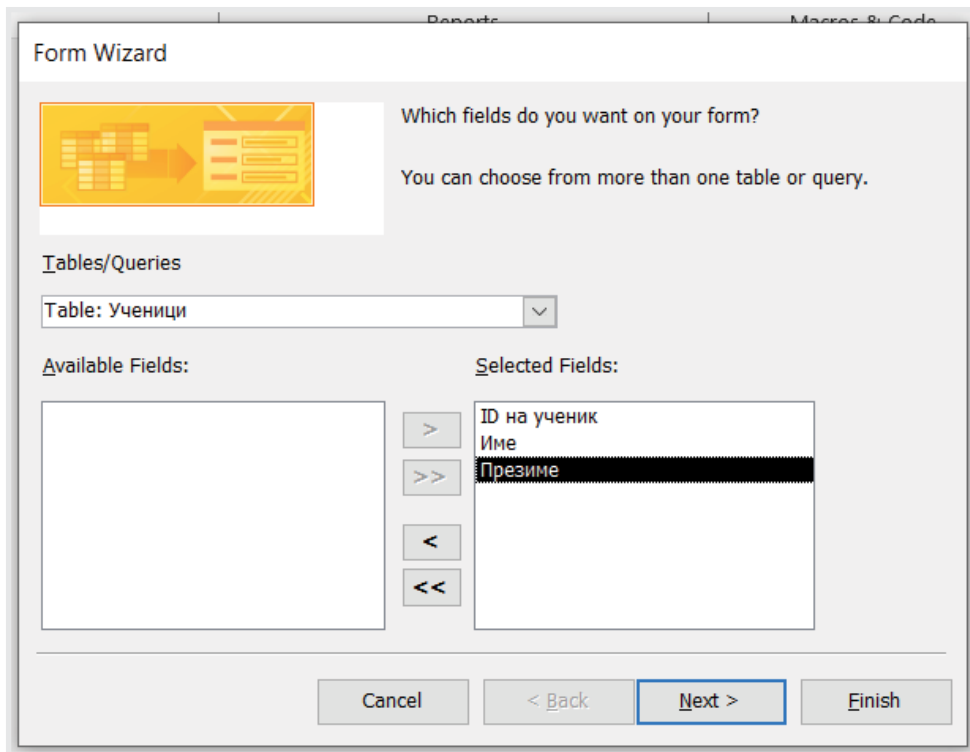
Креирање форма

Во програмите Microsoft Access и LibreOffice Base постојат алатки што овозможуваат автоматско создавање форми врз основа на постојна табела. Корисникот најпрво ја избира табелата што ќе биде извор на податоци, а потоа програмата автоматски ги поставува полињата во форма.



Слика 1 Вклучување на форма

Откако ќе биде креирана формата, можно е дополнително да се уредува нејзиниот изглед. Може да се променат насловите, големината на елементите, нивната положба и начинот на прикажување на податоците. На овој начин формата може да се приспособи на конкретните потреби на корисниците.



Слика 2 Избор на табела и полиња од табела

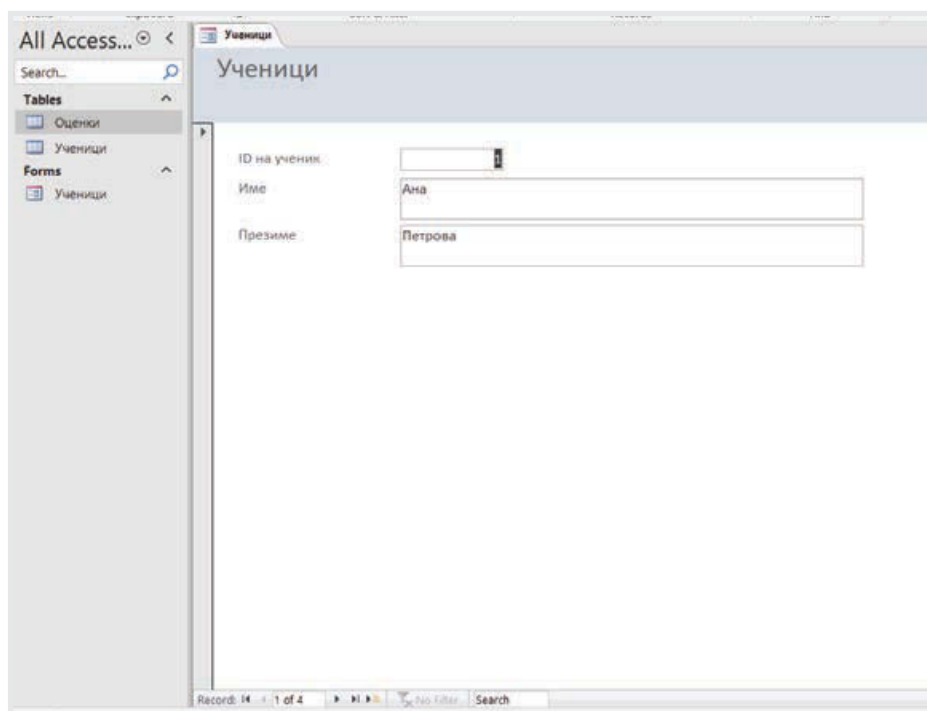
Сите полиња што се потребни се повикуваат во формата од конкретната табела. Ако има повеќе табели во базата, прво е потребно да се избере табелата, а потоа и полињата што треба да се појават во формата.

Внесување податоци преку форма

Откако ќе биде креирана формата, таа може да се користи за внесување нови податоци во базата на податоци. Корисникот веќе нема потреба директно да ја отвора табелата и да бара соодветен ред за внесување на информациите. Наместо тоа, потребно е само да ги пополни полињата што се прикажани на формата.

При внесување на нов запис, секоја информација се внесува во соодветното поле. Ако е за некое поле поставена валидација или ограничување, програмата автоматски ќе провери дали внесената вредност ги исполнува зададените услови. На тој начин се намалува можноста за внесување неточни или нецелосни податоци.

Формите овозможуваат процесот на внесување информации да биде поблизок до пополнување електронски образец отколку до работа со класична табела. Ова е особено корисно кога базата на податоци ја користат лица што немаат напредни информатички знаења.



Слика 3 Форма во MS Access

По креирањето на формата преку Вошебник за форми, се појавуваат сите записи, доколку ги има.

Од претходниот пример може да се забележи дека корисникот веќе не гледа редови и колони. Наместо тоа, секој податок се внесува во посебно поле. Ваквиот начин на работа е поблизок до секојдневните електронски обрасци што ги користиме на интернет.

Уредување на постојни записи

Податоците што се внесени во базата на податоци не остануваат секогаш непроменети. Телефонските броеви се менуваат, корисниците добиваат нови адреси на електронска пошта, а одредени информации треба редовно да се ажурираат. Формите овозможуваат ваквите промени да се извршуваат едноставно и брзо.

Кога корисникот ќе отвори постоечки запис преку формата, сите информации се прикажуваат во полињата што може директно да се изменат. По извршената промена, новите податоци автоматски се запишуваат во базата на податоци. На овој начин нема потреба корисникот да ја бара соодветната информација во табелата.

Слика 4 Промена на постоечки запис

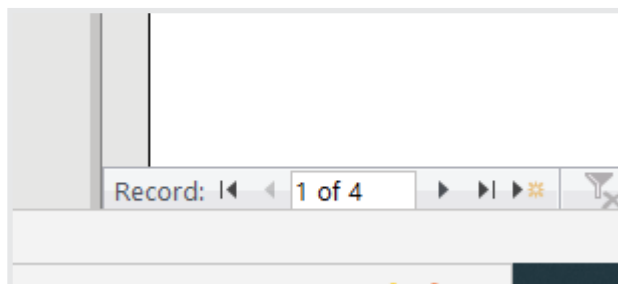
Уредувањето преку форма овозможува поголема прегледност и подобра контрола на информациите. Корисникот се концентрира само на записот што го обработува, без да биде оптоварен со останатите податоци што се наоѓаат во табелата.

Навигација низ записите

Една форма обично не прикажува само еден запис. Таа овозможува движење низ сите записи што се наоѓаат во табелата. За таа цел се користат навигациски копчиња преку кои корисникот може да премине на претходен, следен, прв или последен запис.

Оваа можност е особено корисна кога табелата содржи голем број записи. Наместо корисникот рачно да ги пребарува податоците, доволно е да ги користи навигациските елементи на формата. На тој начин прегледувањето на информациите станува побрзо и поедноставно.

Во многу современи информатички системи формите содржат и дополнителни можности за пребарување и филтрирање на податоците. Иако ќе бидат овие функции подетално обработени во следните лекции, важно е да се разбере дека формите не служат само за внесување информации туку и за нивно ефикасно користење.



Слика 5 Навигација низ записи

Практичен пример: Евиденција на ученици

Да претпоставиме дека е потребно да се изработи едноставен систем за евиденција на учениците во едно училиште. Во базата на податоци веќе постои табела што ги содржи потребните информации за нив. Наместо наставникот или административниот работник да работи директно со табелата, може да се креира форма што ќе овозможи полесен внес и преглед на податоците.

Формата може да содржи полиња за: име, презиме, телефонски број, адреса на електронска пошта и паралелка. При внесување на нов ученик корисникот ги пополнува полињата, а системот автоматски го зачувува записот во базата на податоци. Ако е потребно подоцна да се промени некој податок, корисникот едноставно го отвора соодветниот запис преку формата и ја прави потребната измена.

Ваквиот пристап значително ја подобрува ефикасноста на работата. Наместо сложена табела со многу редови и колони, корисникот работи со едноставен и прегледен екран што ги прикажува само информациите што му се потребни. Поради оваа причина, формите претставуваат еден од најчесто користените елементи во современите бази на податоци.

Анализа на примерот

Од претходниот пример може да се забележи дека формите претставуваат посредник меѓу корисникот и базата на податоци. Тие не ги заменуваат табелите, туку овозможуваат полесна работа со нив. Сите податоци и понатаму се чуваат во табелите, но корисникот најчесто комуницира преку формите.

Со користење на формите се подобрува прегледноста, се намалува можноста за грешки и се олеснуваат внесувањето и ажурирањето на податоците. Поради овие предности, формите имаат важна улога во развојот на информатички системи базирани на бази на податоци.



ЗАКЛУЧОК:

Формите претставуваат важен елемент во базите на податоци и овозможуваат поедноставна комуникација меѓу корисникот и податоците. Наместо директно да се работи со табелите, корисникот користи прегледен интерфејс преку кој може да внесува, да прегледува и да ажурира информации.

Со користење на формите се намалува можноста за грешки при внесувањето на податоците, а истовремено се подобруваат прегледноста и ефикасноста на работата. Поради овие причини, формите претставуваат еден од најчесто користените елементи во современите бази на податоци.



ДАЛИ ЗНАЕШ?

Дали знаеш дека секоја веб-страница за регистрација, најава или онлајн нарачка всушност користи форма? Иако корисникот гледа само полиња за внесување податоци, во заднината тие информации се запишуваат во база на податоци.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува форма во база на податоци?
2. Кои се предностите на формите во однос на директната работа со табели?
3. За што најчесто се користат формите?
4. Како се внесуваат податоци преку форма?
5. Како се врши ажурирање на постоечки записи?
6. Што претставува навигација низ записите?
7. Дали формите ги заменуваат табелите?



ПРАКТИЧНИ ЗАДАЧИ

Задача 1

Отвори постоечка табела и креирај форма што ќе ги прикажува сите полиња.

Задача 2

Преку формата внеси најмалку пет нови записи.

Задача 3

Промени два постоечки записи користејќи ја формата.

Задача 4

Прегледај ги сите записи користејќи ги навигациските копчиња.

Задача 5

Предложи три подобрувања на формата што би ја направиле попрегледна за корисниците.



ИСТРАЖУВАЧКА ЗАДАЧА

Истражи примери на електронски форми што секојдневно се користат на интернет.

Пронајди најмалку три примери и анализирај:

- која е нивната намена;
- кои податоци се внесуваат;
- како се организирани полињата;
- кои правила за внесување се применуваат.



ЗАПОМНИ ШТО НАУЧИ

- Формата претставува кориснички интерфејс за работа со податоците.
- Формите овозможуваат внесување, прегледување и ажурирање на записите.
- Податоците и натаму се чуваат во табелите.
- Формите ја подобруваат прегледноста и ја олеснуваат работата.
- Навигациските елементи овозможуваат движење низ записите.
- Формите се широко користени во современите информатички системи.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ



Покрај едноставните форми, современите системи овозможуваат креирање сложени форми што содржат копчиња, листи за избор, слики, автоматски пресметки и поврзани податоци од повеќе табели. Во професионалните апликации формите често се главниот начин на комуникација меѓу корисникот и базата на податоци.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

Заокружи го точниот одговор

1. Формата служи за:

- а) бришење база на податоци
- б) внесување и прегледување податоци
- в) форматирање на оперативен систем
- г) компресирање датотеки

2. Податоците внесени преку форма се чуваат во:

- а) оперативна меморија
- б) форма
- в) табела
- г) извештај

Точно или неточно

- 1. Формите овозможуваат полесна работа со податоците.
- 2. Податоците се чуваат во самата форма.
- 3. Формите може да се користат за ажурирање на записи.
- 4. Навигациските копчиња овозможуваат движење низ записите.



АНАЛИЗА

Во една училишна база на податоци постои табела со 500 ученици.

Објасни:

- зошто користењето форма е попрактично од директна работа со табелата;
- кои предности ги добива корисникот;
- како може да помогне формата при внесување нов ученик.

3.6



ПРАШАЛНИЦИ (QUERIES) – ПРЕБАРУВАЊЕ И СЕЛЕКТИРАЊЕ ПОДАТОЦИ

ВЕЌЕ ЗНАЕШ ДА...

- креираш табели во база на податоци;
- користиш различни типови податоци;
- применуваш примарен клуч;
- внесуваш податоци преку форми;
- прегледуваш и да ажурираш записи.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Што ако во базата има илјадници записи, а ти треба само еден?
2. Како може да се пронајдат сите ученици од една паралелка?
3. Како може да се прикажат само учениците со одличен успех?
4. Дали е потребно секогаш да се прегледуваат сите записи?
5. Како може компјутерот брзо да ги пронајде потребните информации?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што претставува прашалник;
- користиш прашалници за пребарување на податоци;
- применуваш услови за селектирање на записи;
- подредуваш информации според зададени критериуми;
- анализираш резултати добиени од прашалник;
- креираш едноставни прашалници.

Во реалните бази на податоци често се чуваат стотици, илјадници или дури и милиони записи. Иако се важни сите податоци, во практиката корисниците најчесто имаат потреба само од одреден дел од информациите. На пример, наставникот може да сака да ги види само учениците од определена паралелка, библиотекарот само книгите што моментално се издадени, а администрацијата само учениците што не ја подмириле екскурзијата.

Прегледувањето на целата табела секогаш кога е потребна одредена информација би било бавно и неефикасно. Поради тоа, системите за управување со бази на податоци овозможуваат користење на посебни механизми за пребарување и селектирање на информациите. Овие механизми се нарекуваат прашалници.

Прашалниците претставуваат алатка што овозможува од базата на податоци да се издвојат само информациите што ги задоволуваат зададените услови. Наместо корисникот рачно да ги пребарува сите записи, системот автоматски ги пронаоѓа и ги прикажува само потребните резултати.

Што претставува прашалник?

Прашалникот претставува барање што му се поставува на системот за управување со базата на податоци. Со помош на прашалникот корисникот определува кои податоци сака да ги добие и под кои услови треба да бидат избрани.

За разлика од табелата, која ги содржи сите податоци, прашалникот најчесто прикажува само дел од нив. На пример, може да се прикажат само учениците од одредена паралелка, само учениците со просек поголем од 4,50 или само учениците што учествуваат на натпревари.

Важно е да се нагласи дека прашалникот не ги менува податоците во табелата. Тој само ги прикажува информациите што ги задоволуваат зададените услови. Откако ќе се затвори прашалникот, податоците во табелата остануваат непроменети.

Практичен пример

Да се разгледа следната табела.

Име	Презиме	Паралелка	Просек
Ана	Петрова	I-1	5.00
Марко	Стојанов	I-2	4.20
Елена	Николовска	I-1	4.80
Merita	Rexhepi	I-3	3.90

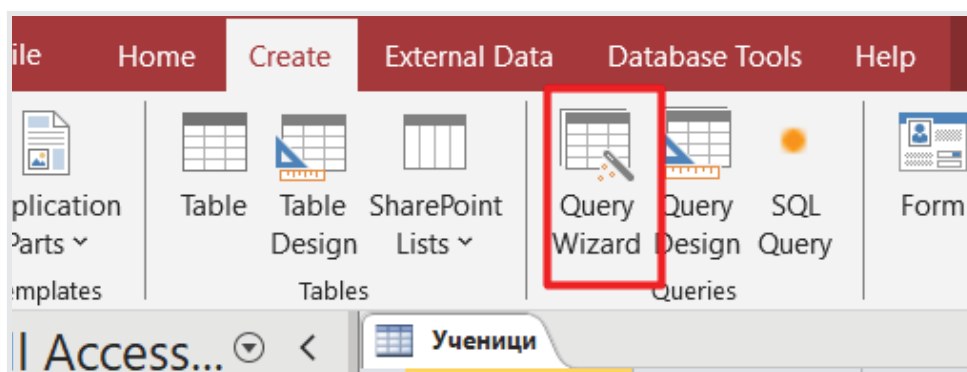
Да претпоставиме дека е потребно да се прикажат само учениците од паралелката I-1.

Наместо рачно да се пребарува табелата, може да се креира прашалник што ќе го постави условот:

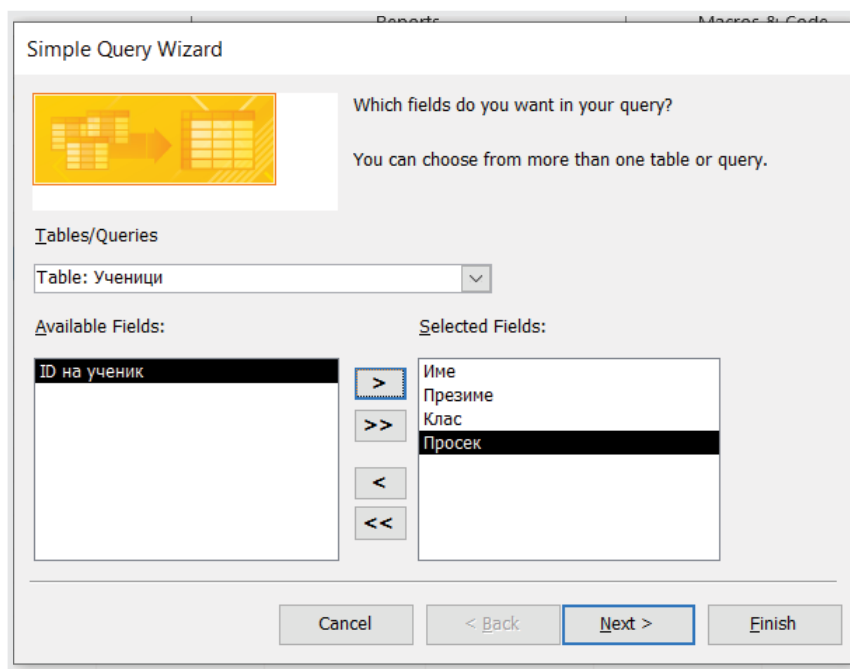
Паралелка = I-1

Како резултат ќе се добие:

Име	Презиме	Паралелка	Просек
Ана	Петрова	I-1	5.00
Елена	Николовска	I-1	4.80

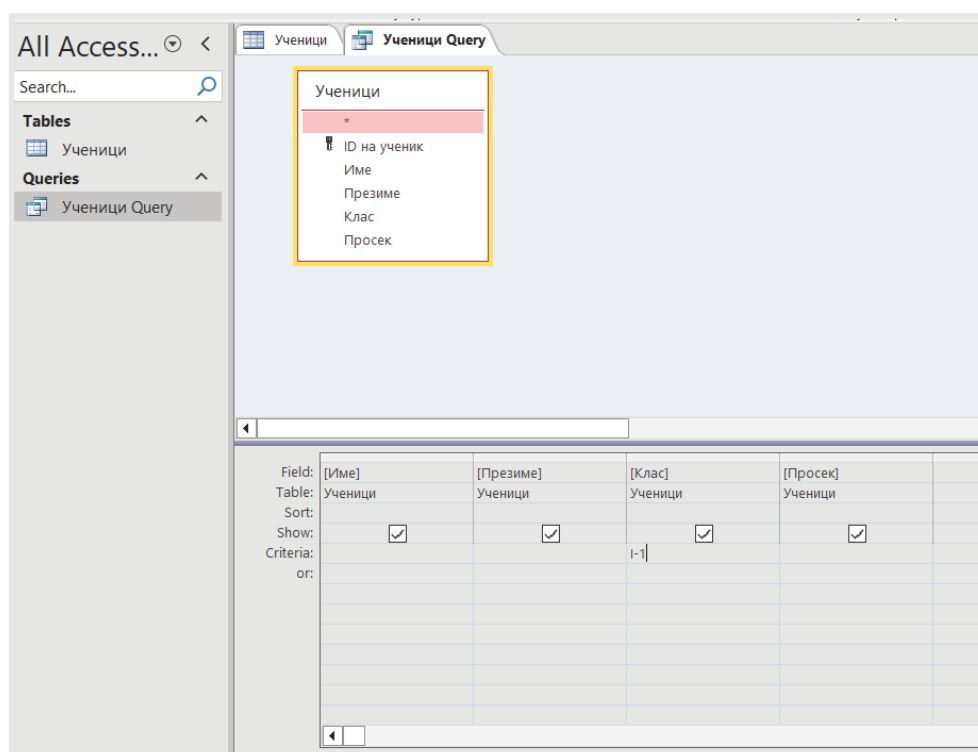


Слика 1 Прашалник во MS Access



Слика 2 Поставување параметри во прашалник

На веќе креираниот извештај кој ги содржи сите елементи, преку дизајнерски преглед може да се постават критериумите што се потребни.



Слика 3 Поставување критериум во прашалник

Во овој случај се учениците од I-1 и тоа се запишува во полето критериум кај „Клас“. Откако се извршува прашалникот со избор на командата „Run“, се појавува ситуација, каде што се прелистуваат само учениците од зададената паралелка.

The screenshot shows the Microsoft Access interface. On the left, the 'All Access Objects' pane shows 'Tables' and 'Queries'. The 'Queries' section is expanded, showing 'Ученици Query'. The main window displays a table view of the query results. The table has four columns: 'Име' (Name), 'Презиме' (Surname), 'Клас' (Class), and 'Просек' (Average). The data rows are: 'Ана Петрова' with average 5, 'Елена Николовска' with average 4.8, and a summary row with an asterisk and average 0. The 'Име' column has a filter applied, indicated by a small 'v' icon in the header.

Име	Презиме	Клас	Просек
Ана	Петрова	I-1	5
Елена	Николовска	I-1	4.8
*			0

Слика 4 Ефект од поставен критериум во прашалник

Вака подготвениот прашалник ги дава податоците што се бараат од базата. Во една база може да се изготват бесконечно прашалници за различни особености на базата.

Зошто се користат прашалници?

Прашалниците значително го олеснуваат користењето на базите на податоци. Наместо корисникот да работи со голем број записи, системот автоматски ги издвојува потребните информации. Ова не само што заштедува време туку и ја намалува можноста за грешки.

Дополнително, прашалниците овозможуваат податоците да се анализираат на различни начини. Истата табела може да се користи за добивање различни информации во зависност од поставените услови. Така, еднаш може да се пребаруваат ученици според паралелката, другпат според успехот, а третпат според бројот на изостаноци.

Поради овие можности, прашалниците се една од најкорисните алатки при работа со бази на податоци и имаат широка примена во различни информатички системи.

Услови во прашалниците

Една од најголемите предности на прашалниците е можноста за поставување услови според кои ќе се избираат податоците. Наместо да се прикажуваат сите записи од табелата, корисникот може да определи кои информации сака да ги добие. Ова овозможува брзо пронаоѓање на потребните податоци дури и кога базата содржи голем број записи.

Условот претставува правило што треба да го исполни записот за да биде прикажан во резултатите. На пример, може да се побараат сите ученици од определена паралелка, сите книги од одреден автор или сите производи чија количина е помала од зададена вредност. Системот автоматски ги проверува записите и ги прикажува само оние што го задоволуваат условот.

Во практиката може да се користат различни видови услови. Некои услови проверуваат дали е вредноста еднаква на одреден податок, додека други проверуваат дали е вредноста поголема, помала или се наоѓа во определен опсег. Благодарение на оваа флексибилност, прашалниците може да се приспособат на различни потреби на корисниците.

Пример за користење услов

Да се разгледа следната табела.

Име	Презиме	Просек
Ана	Петрова	5.00
Марко	Стојанов	4.20
Елена	Николовска	4.80
Merita	Rexhepi	3.90

Да претпоставиме дека е потребно да се прикажат само учениците со просек поголем од 4.50.
Условот ќе биде:

Просек > 4.50

Резултатот ќе ги содржи следните записи:

Име	Презиме	Просек
Ана	Петрова	5.00
Елена	Николовска	4.80

Од примерот може да се забележи дека системот автоматски ги издвоил само записите што го исполнуваат поставениот услов.

Field:	[Име]	[Презиме]	[Клас]	[Просек]	
Table:	Ученици	Ученици	Ученици	Ученици	
Sort:					
Show:	☒	☒	☒	☒	☐
Criteria:				>4.5	
or:					

Слика 5 Поставување услов за просек

Подредување на резултатите

Покрај пребарувањето на податоците, прашалниците овозможуваат и нивно подредување. Подредувањето претставува организирање на резултатите според определен редослед. Најчесто се користи за полесен преглед и анализа на информациите.

Податоците може да се подредуваат по азбучен редослед, по бројчени вредности или по датуми. На пример, списокот на ученици може да се подреди според презимето, списокот на производи според цената, а списокот на книги според годината на издавање.

Подредувањето не ги менува податоците во табелата. Тоа влијае само врз начинот на кој се прикажуваат резултатите во прашалникот. На тој начин корисникот може брзо да добие појасен преглед на информациите.

Пример за подредување

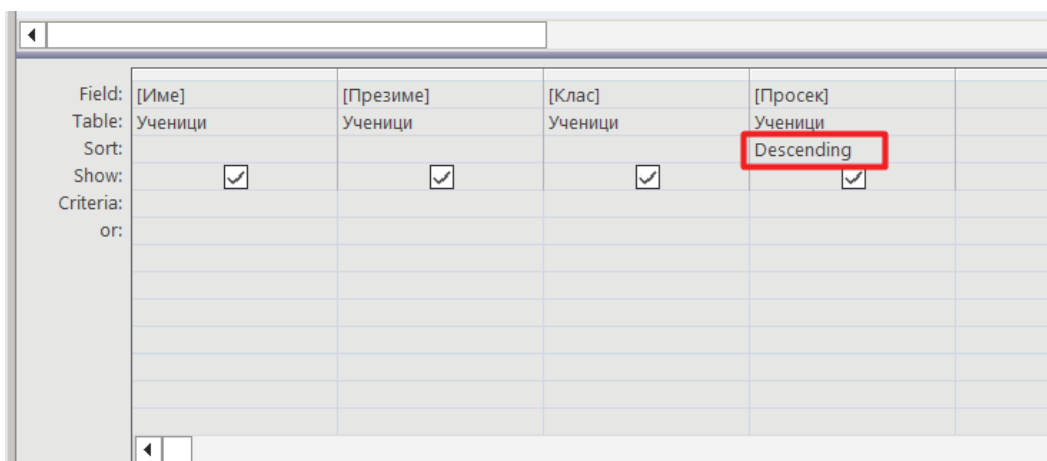
Да се разгледа следниот резултат.

Име	Просек
Марко	4.20
Ана	5.00
Merita	3.90
Елена	4.80

Ако се примени подредување по просек во опаѓачки редослед, резултатот ќе биде:

Име	Просек
Ана	5.00
Елена	4.80
Марко	4.20
Merita	3.90

На овој начин највисоките вредности се прикажуваат на почетокот на списокот.



Слика 6 Сортирање на резултати

По секоја промена што се прави во дизајнерскиот преглед на прашалникот потребно е да се активира командата „Run“ за да се добие новата состојба на прашалникот.

Комбинирање на повеќе услови

Во многу ситуации еден услов не е доволен за добивање на потребните информации. Поради тоа, прашалниците овозможуваат користење на повеќе услови истовремено. На пример, може да се побараат ученици што припаѓаат на определена паралелка и воедно имаат просек над определена вредност.

Да претпоставиме дека треба да се пронајдат учениците од паралелката I-1 со просек поголем од 4.50.

Потребни се два услови:

- Паралелка = I-1
- Просек > 4.50

Само записите што ги исполнуваат двата услови ќе бидат прикажани во резултатите.

Комбинирањето услови овозможува многу попрецизно пребарување на информациите. Колку што се попрецизно поставени условите, толку порелевантни ќе бидат добиените резултати.

Практичен пример: Училишна база на податоци

Да се претпостави дека училиштето води евиденција за сите ученици во прва година. Директорот сака да добие список на ученици што имаат просек поголем од 4.50 и истовремено немаат повеќе од три неоправдани изостаноци.

Наместо рачно да се прегледуваат стотици записи, може да се креира прашалник што автоматски ќе ги издвојува само учениците што ги исполнуваат двата услови. Резултатот ќе биде добиен за неколку секунди, без потреба од дополнителна обработка.

Ваквите ситуации се среќаваат секојдневно во образованието, во здравството, банкарството и во администрацијата. Благодарение на прашалниците, корисниците можат брзо да добијат точни информации од големи количини податоци. Поради оваа причина, прашалниците се еден од најмоќните механизми за работа со бази на податоци.

Анализа на примерот

Од претходните примери може да се заклучи дека прашалниците не служат само за пронаоѓање на податоци туку и за нивна анализа. Тие овозможуваат издвојување на потребните информации, нивно подредување и комбинирање според различни критериуми. На тој начин корисникот добива само релевантни податоци, наместо да работи со целата содржина на табелата.

Благодарение на можноста за поставување услови и подредување на резултатите, прашалниците значително ја подобруваат ефикасноста на работата со базите на податоци. Тие претставуваат незаменива алатка во современите информатички системи и се користат секојдневно за пребарување и анализа на информации.

 **Screenshot 5 – Прашалник со повеќе услови**

 **Screenshot 6 – Резултати од прашалникот**



ЗАКЛУЧОК:

Прашалниците претставуваат важна алатка за работа со бази на податоци бидејќи овозможуваат брзо пронаоѓање, селектирање и анализа на информациите. Наместо корисникот рачно да ги пребарува сите записи во табелата, системот автоматски ги издвојува податоците што ги задоволуваат зададените услови.

Со користење на прашалници може да се применуваат различни критериуми за пребарување, да се комбинираат повеќе услови и да се подредуваат резултатите според потребите на корисникот. На овој начин се подобрува ефикасноста на работата и се олеснува донесувањето одлуки врз основа на точни информации.

Поради својата практична примена, прашалниците претставуваат еден од најважните механизми во современите бази на податоци и се користат во речиси сите информациски системи.



ДАЛИ ЗНАЕШ?

Дали знаеш дека интернет-пребарувачите користат принцип сличен на прашалниците? Кога внесуваш поим за пребарување, системот автоматски пребарува низ огромен број податоци и ги прикажува само резултатите што соодветствуваат на поставеното барање.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува прашалник во база на податоци?
2. Која е разликата помеѓу табела и прашалник?
3. Што претставува услов во прашалник?
4. Зошто се користи сортирање на резултатите?
5. Дали прашалникот ги менува податоците во табелата?
6. Зошто е корисно комбинирањето на повеќе услови?
7. Во кои ситуации најчесто се користат прашалници?



ПРАКТИЧНИ ЗАДАЧИ

Задача 1

Креирај прашалник што ќе ги прикажува сите ученици од една избрана паралелка.

Задача 2

Креирај прашалник што ќе ги прикажува учениците со просек поголем од 4,50.

Задача 3

Подреди ги резултатите од претходниот прашалник според просек во опаѓачки редослед.

Задача 4

Креирај прашалник што ќе ги прикажува учениците од паралелката I-1 со просек поголем од 4,50.

Задача 5

Направи прашалник што ќе ги прикажува само учениците што имаат повеќе од пет изоста-ноци.



ИСТРАЖУВАЧКА ЗАДАЧА

Во училиштето постојат податоци за учениците, наставниците, предметите и за оценките.

Предложи најмалку пет прашања на кои директорот би можел да добие одговор со користење на прашалници.

За секое прашање наведи:

- кои податоци се потребни;
- кои услови треба да се постават;
- каков резултат се очекува.



ЗАПОМНИ ШТО НАУЧИ

- Прашалникот претставува алатка за пребарување и селектирање податоци.
- Прашалниците овозможуваат прикажување само на потребните информации.
- Условите определуваат кои записи ќе бидат прикажани.
- Можно е користење на повеќе услови истовремено.
- Резултатите може да се подредуваат според различни критериуми.
- Прашалниците не ги менуваат податоците во табелата.
- Прашалниците значително ја подобруваат ефикасноста на работата со базите на податоци.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите системи за управување со бази на податоци постојат напредни прашалници што можат да вршат статистички пресметки, групирање на податоците и анализа на информации од повеќе поврзани табели. Во големите информациски системи овие прашалници често се користат за изработка на извештаи, анализа на продажбата, следење на финансиските податоци и поддршка при донесување деловни одлуки.





ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

Заокружи го точниот одговор

1. Прашалникот служи за:

- а) форматирање табели
- б) пребарување и селектирање на податоците
- в) инсталација на програми
- г) креирање оперативен систем

2. Условот во прашалникот:

- а) ги брише записите
- б) определува кои записи ќе бидат прикажани
- в) ја менува структурата на табелата
- г) ја затвора базата на податоци

3. Сортирањето се користи за:

- а) подредување на резултатите
- б) бришење на резултатите
- в) креирање форми
- г) додавање нови полиња

Точно или неточно

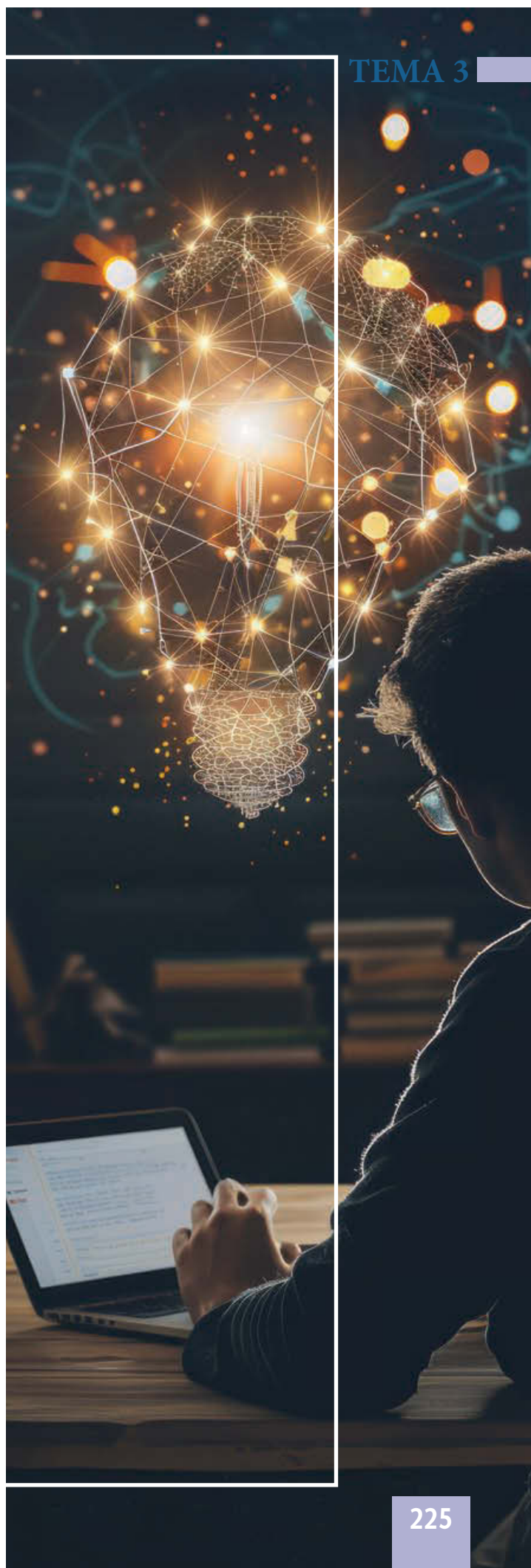
- 1. Прашалниците може да користат повеќе услови истовремено.
- 2. Резултатите од прашалникот може да се сортираат.
- 3. Прашалникот секогаш ги прикажува сите записи од табелата.
- 4. Прашалникот ги менува податоците во табелата.

АНАЛИЗА

Во база на податоци за библиотека се чуваат информации за книги, автори и корисници.

Објасни:

- каков прашалник би креирал за да ги прикажеш сите книги од определен автор;
- каков услов би поставил;
- како би ги подредил резултатите за да бидат попрегледни.





3.7

ИЗВЕШТАИ (REPORTS) – ПРИКАЖУВАЊЕ И ПЕЧАТЕЊЕ НА ПОДАТОЦИТЕ

ВЕЌЕ ЗНАЕШ ДА...

- креираш табели во база на податоци;
- користиш различни типови податоци;
- внесуваш податоци преку форми;
- креираш прашалници за пребарување информации;
- користиш услови и сортирање на резултатите;
- анализираш податоци добиени од база на податоци.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Како може да се подготват за печатење податоците од базата на податоци?
2. Дали е табелата секогаш најдобар начин за прикажување на информациите?
3. Како може да се добие список на ученици подготвен за печатење?
4. Како изработуваат институциите извештаи за својата работа?
5. На кој начин може попрегледно да се претстават резултатите од прашалниците?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што претставува извештај;
- ги наведуваш предностите од користење извештаи;
- креираш едноставен извештај;
- користиш податоци од табели и прашалници во извештаи;
- уредуваш изглед на извештај;
- ја анализираш примената на извештаите во информатичките системи.

При работа со податоци податоците се организираат во табели; може да се внесуваат преку форми и може да се пребаруваат со помош на прашалници. Меѓутоа, во многу ситуации е потребно добиените информации да бидат прикажани на попрегледен начин или да се подготват за печатење. На пример, можеби наставникот сака да испечати список на ученици, библиотекарот извештај за задолжени книги, а директорот преглед на успехот по паралелки.

Иако табелите и прашалниците ги содржат потребните податоци, нивниот изглед не е секогаш погоден за презентација или за печатење. Поради тоа, системите за управување со бази на податоци овозможуваат користење на извештаи. Извештаите служат за организирано и професионално прикажување на информациите.

Извештајот претставува посебен објект во базата на податоци што ги прикажува податоците во подготвен облик за прегледување, печатење или споделување. Тој може да содржи наслови, поднаслови, групирани информации, броеви на страници и други елементи што ја подобруваат прегледноста.

Што претставува извештај?

Извештајот е организиран приказ на податоците што се наоѓаат во базата на податоци. За разлика од формите, кои најчесто се користат за внесување и уредување на информации, извештаите се наменети за прикажување и презентирање на податоците.

Извештаите можат да ги користат информациите директно од табелите или од претходно креирани прашалници. На тој начин корисникот може да прикаже само одреден дел од податоците што се потребни за конкретна анализа или презентација.

Во практиката извештаите често се користат за изработка на списоци, статистички прегледи, резултати од анализи, финансиски извештаи и други документи што треба да бидат јасно организирани и лесни за читање.

Практичен пример

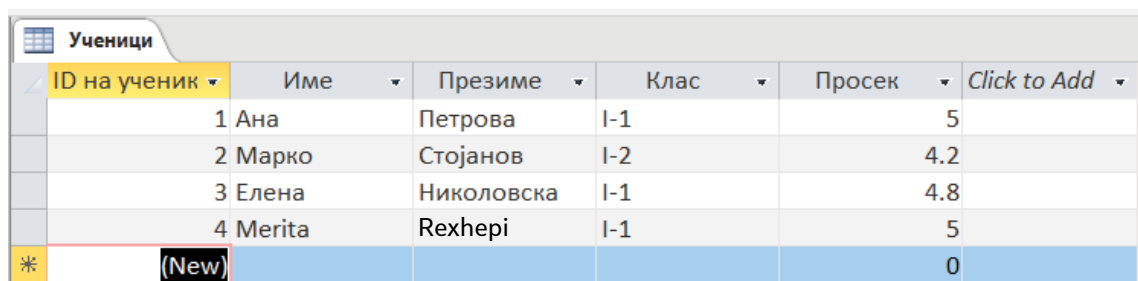
Да претпоставиме дека училиштето сака да подготви список на ученици од паралелката I-1.

Податоците најпрво се чуваат во табела, а потоа преку прашалник се издвојуваат само учениците од избраната паралелка.

Добиените резултати може да изгледаат вака:

Име	Презиме	Паралелка
Ана	Петрова	I-1
Елена	Николовска	I-1
Merita	Rexhepi	I-1

Иако се точни информациите, ваквиот приказ не е најсоодветен за печатење и архивирање. Поради тоа се креира извештај што ќе содржи наслов, датум на изработка и уредно подредени податоци.



ID на ученик	Име	Презиме	Клас	Просек	Click to Add
1	Ана	Петрова	I-1	5	
2	Марко	Стојанов	I-2	4.2	
3	Елена	Николовска	I-1	4.8	
4	Merita	Rexhepi	I-1	5	
*	(New)			0	

Слика 1 Изворни податоци за извештај

Предности на извештаите

Извештаите овозможуваат информациите да бидат прикажани на попрегледен и професионален начин. Тие може да содржат различни елементи што ја подобруваат читливоста, како што се наслови, логоа, броеви на страници и групирани податоци.

Дополнителна предност е можноста за печатење на информациите. Наместо корисникот да печати табели или резултати од прашалници, може да изработи извештај што е специјално подготвен за оваа намена.

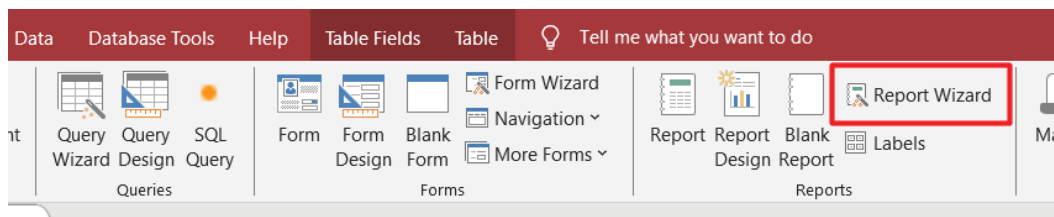
Извештаите овозможуваат и подобра презентација на информациите. Податоците може да бидат организирани според различни критериуми, а корисникот лесно да ги пронајде потребните информации.

Креирање извештај

Во Microsoft Access и LibreOffice Base постојат алатки што овозможуваат автоматско создавање извештаи. Корисникот најпрво избира табела или прашалник како извор на податоци, а потоа програмата генерира основен извештај.

По креирањето, извештајот може дополнително да се уредува. Можно е да се промени распоредот на податоците, да се додадат наслови и да се избере начинот на прикажување на информациите. На тој начин извештајот се приспособува на конкретните потреби на корисникот.

Во образовните институции извештаите често се користат за изработка на списоци на учениците, извештаи за успехот, статистики за изостаноците и други податоци што редовно се користат во работата.



Слика 2 Креирање извештај со Report Wizard

Уредување на изгледот на извештајот

Откако ќе биде креиран извештајот, неговиот изглед може дополнително да се подобрува. Корисникот може да го промени распоредот на полињата, да додаде наслови и поднаслови, да постави броеви на страниците и да избере соодветен распоред за печатење.

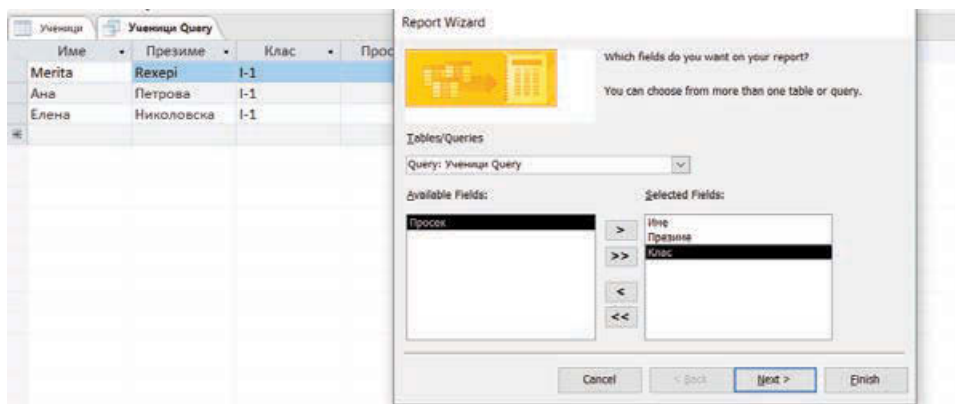
Добро организираниот извештај овозможува податоците полесно да се читаат и да се анализираат. Особено кај извештаите што содржат голем број записи, правилното уредување има големо значење за нивната употребливост.

Во многу случаи извештаите содржат и дополнителни информации како: датум на изработка, име на институцијата или краток опис на содржината. Овие елементи придонесуваат за подобра организација и професионален изглед на документот.

Практичен пример: Извештај за успехот на учениците

Да претпоставиме дека е потребно да се изработи извештај за учениците што имаат просек поголем од 4,50.

Најпрво се креира прашалник што ќе ги издвојува учениците што го исполнуваат условот. Потоа добиените резултати се користат како основа за извештајот. Прво се поставува прашалник во кој критериумот е просекот да е на 4.50. Таквиот прашалник понатаму се одбира при изработката на извештај.



Слика 2 Подготовка на извештај

Во извештајот може да бидат прикажани името, презимето, паралелката и просечниот успех на секој ученик. Дополнително може да се додаде наслов „Ученици со одличен успех“ и датумот на изработка на документот.

На овој начин добиените информации стануваат попрегледни и подготвени за печатење или презентација пред наставничкиот совет, класните раководители или раководството на училиштето.

Презиме	Клас	Просек
Рехери	I-1	5
Петрова	I-1	5
Николовска	I-1	4.8

Слика 3 Извештај поготвен за печатење

Анализа на примерот

Од претходниот пример може да се забележи дека извештајот не претставува нов извор на податоци, туку организиран приказ на веќе постоечки информации. Податоците остануваат во табелите и во прашалниците, додека извештајот служи за нивна презентација.

Со користење на извештаи се подобрува читливоста на информациите и се олеснува нивното користење во секојдневната работа. Наместо корисникот да работи со сложени табели и прашалници, тој добива документ што е подготвен за прегледување и печатење.

Поради овие причини, извештаите се важен дел од современите бази на податоци и имаат широка примена во образованието, во администрацијата, здравството, финансиските институции и во многу други области.



ЗАКЛУЧОК:

Извештаите претставуваат важен елемент на базите на податоци и овозможуваат информациите да бидат прикажани на организиран и прегледен начин. За разлика од табелите и прашалниците, кои се користат за чување и пребарување на податоците, извештаите се наменети за нивна презентација и печатење.

Со користење на извештаи може да се подготват различни списоци, анализи и прегледи што се користат во секојдневната работа на институциите и организациите. Благодарение на можноста за уредување на изгледот, извештаите овозможуваат информациите да бидат прикажани јасно и професионално.

Поради својата практична примена, извештаите се важен дел од секоја база на податоци и често се користат за подготовка на документи што треба да бидат разгледани, архивирани или отпечатени.



ДАЛИ ЗНАЕШ?

Дали знаеш дека најголем дел од документите што ги добиваат директорите, менаџерите и раководителите во различни институции најчесто се генерирани автоматски од бази на податоци? Наместо рачно да се подготвуваат, извештаите се создаваат за неколку секунди врз основа на податоците што веќе ги има во системот.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува извештај во база на податоци?
2. Која е разликата помеѓу форма и извештај?
3. Од кои извори може да се добиваат податоци за извештај?
4. Зошто се користат извештаи?
5. Кои се предностите на извештаите при печатење на податоците?
6. Како може да се подобри изгледот на извештајот?
7. Во кои области најчесто се користат извештаи?



ПРАКТИЧНИ ЗАДАЧИ

Задача 1

Креирај извештај врз основа на табелата со ученици.

Во извештајот прикажи:

- име;
- презиме;
- паралелка.

Задача 2

Креирај прашалник што ќе ги прикажува учениците со просек поголем од 4,50 и врз основа на него изработи извештај.

Задача 3

Додај наслов на извештајот:

„Список на ученици со одличен успех“

Задача 4

Додај го датумот на изработка и бројот на страниците во извештајот.

Задача 5

Испечати или извези го извештајот во PDF формат.



ИСТРАЖУВАЧКА ЗАДАЧА

Истражи во кои ситуации училиштата користат извештаи.

Наведи најмалку пет примери и објасни:

- која е нивната намена;
- кои податоци се користат;
- кој би ги користел тие извештаи.



ЗАПОМНИ ШТО НАУЧИ

- Извештајот служи за организирано прикажување на податоците.
- Извештаите можат да користат податоци од табели и прашалници.
- Тие се наменети за прегледување, печатење и презентација на информации.
- Изгледот на извештајот може дополнително да се уредува.
- Извештаите не претставуваат нови податоци, туку приказ на постојните информации.
- Извештаите се широко користени во современите информатички системи.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ



Во професионалните информациски системи извештаите често содржат графичи, статистички анализи и автоматски пресметки. На пример, еден финансиски извештај може автоматски да пресметува приходи и расходи, додека извештај за продажба може да прикажува трендови и споредби меѓу различни периоди.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

Заокружи го точниот одговор

1. Извештајот најчесто се користи за:

- а) внесување податоци
- б) прикажување и печатење на податоци
- в) креирање оперативен систем
- г) бришење табели

2. Извештајот може да користи податоци од:

- а) табела
- б) прашалник
- в) табела и прашалник
- г) ниту едно од наведените

3. Извештаите се користат кога:

- а) треба да се организираат податоци за презентација
- б) треба да се инсталира програма
- в) треба да се креира оперативен систем
- г) треба да се форматира диск

Точно или неточно

- 1. Извештаите може да се печатат.
- 2. Извештаите служат за внесување нови записи.
- 3. Извештаите можат да содржат наслови и броеви на страниците.
- 4. Извештаите можат да користат податоци од прашалници.



АНАЛИЗА

Во училишна база на податоци е потребно да се подготви список на ученици со одличен успех. Објасни:

- кои податоци ќе бидат потребни;
- дали најпрво треба да се креира прашалник;
- зошто е извештајот подобро решение од обична табела за печатење на списокот.



3.8

ИЗРАБОТКА НА ЦИРКУЛАРНИ ПИСМА СО БАЗА НА ПОДАТОЦИ

ВЕЌЕ ЗНАЕШ ДА...

- креираш табели во базата на податоци;
- внесуваш и да ажурираш податоци;
- користиш форми за работа со податоците;
- креираш прашалници за пребарување информации;
- подготвуваш извештаи;
- организираш податоци во базата на податоци.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

1. Како може ист документ да се испрати до голем број различни лица?
2. Дали е практично рачно да се внесува име и презиме во секоја покана?
3. Како може училиштето да испечати стотици потврди за ученици?
4. Како испраќаат компаниите персонализирани писма до своите корисници?
5. Дали е можно еден документ автоматски да се приспособува за секој примач?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

1. објаснуваш што претставува циркуларно писмо;
2. користиш податоци од базата на податоци во документ;
3. поврзуваш документ со извор на податоци;
4. вметнуваш полиња за спојување;
5. генерираш повеќе документи од еден шаблон;
6. ја анализираш примената на циркуларните писма во практиката.

Во многу ситуации е потребно ист документ да се испрати до голем број примачи. На пример, училиштето може да испраќа покани за родителска средба, потврди за ученици или благодарници за учесници на натпревари. Иако содржината на документот е речиси иста, одредени информации како името, презимето или адресата се различни за секој примач.

Кога би се изработувал посебно секој документ, тоа би барало многу време и би ја зголемило можноста за појава на грешки. Поради тоа, современите програми овозможуваат автоматско комбинирање на документ со податоци од базата на податоци или од табела. На овој начин еден шаблон може да се искористи за создавање голем број персонализирани документи.

Овој процес се нарекува циркуларно писмо. Тој овозможува автоматско вметнување на податоците од базата на податоци во однапред подготвен документ. Како резултат се добиваат повеќе документи што имаат иста содржина, но содржат различни лични податоци.

Што претставува циркуларно писмо?

Циркуларното писмо претставува документ што е поврзан со извор на податоци. Изворот на податоци најчесто е база на податоци, табела или листа со информации за примачите. Во документот се поставуваат специјални полиња што подоцна автоматски се заменуваат со вистинските податоци.

На пример, наместо во документот да биде напишано конкретно име, може да се постави поле „Име“. При генерирањето на документите, програмата автоматски ќе го замени ова поле со името на секој примач.

Благодарение на овој пристап, еден документ може да се искористи за создавање десетици, стотици или дури илјадници персонализирани писма. Ова значително ја подобрува ефикасноста на работата и овозможува брзо подготвување на голем број документи.

Практичен пример

Да претпоставиме дека училиштето организира натпревар по информатика и треба да испрати покани до учениците.

Во базата на податоци постои табела:

Име	Презиме	Паралелка
Ана	Петрова	II-1
Марко	Стојанов	II-2
Елена	Николовска	II-1

Во програмата за обработка на текст се подготвува следниот шаблон:

Почитуван/а <<Име>> <<Презиме>>,

Ве покануваме да учествувате на училишниот натпревар по информатика.

Со почит,

Организациски одбор

По спојувањето на документот со базата на податоци автоматски ќе се создадат посебни покани за секој ученик.

Ученици				
ID на ученик	Име	Презиме	Клас	Click to Add
1	Ана	Петрова	II-1	
2	Марко	Стојанов	II-2	
3	Елена	Николовска	II-1	
*	(New)			

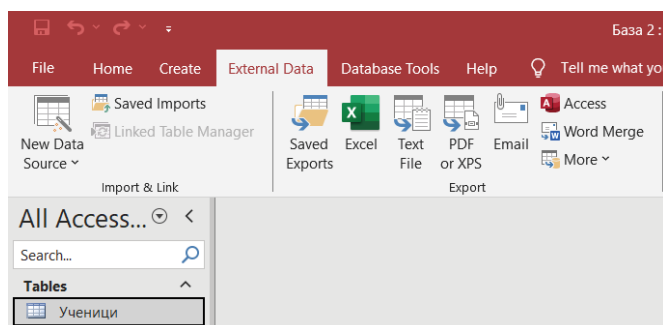
Слика 1 Табела со податоци за примачи

Поврзување на документ со базата на податоци

За да се создаде циркуларно писмо, најпрво е потребно документот да се поврзе со изворот на податоци. Изворот може да биде база на податоци, табела или друга структура што содржи информации за примачите.

Откако ќе се воспостави врската, програмата добива пристап до сите записи што се наоѓаат во базата. На тој начин податоците може автоматски да се користат при создавањето на документите.

Во современите програми оваа постапка најчесто се изведува со помош на посебен, т.н. волшебник (Wizard) што го води корисникот низ процесот на поврзување и избор на потребните полиња. Пред да се почне со постапката, табелите треба да бидат затворени. Всушност, овој ексклузивен мод на базата дозволува креирање на останати компоненти на „пасивни“ документи. Не смее да се креираат писма од бази во кои се работи во моментот. Доколку е табелата активна (отворена) волшебникот нема да може да креира писмо. Потребно е само да биде означена (кликната) табелата, но не и отворена. Дури на некои верзии на софтвери потребно е прво да се затвори целата база за да може да се креира спојување со текстуален уредувач.



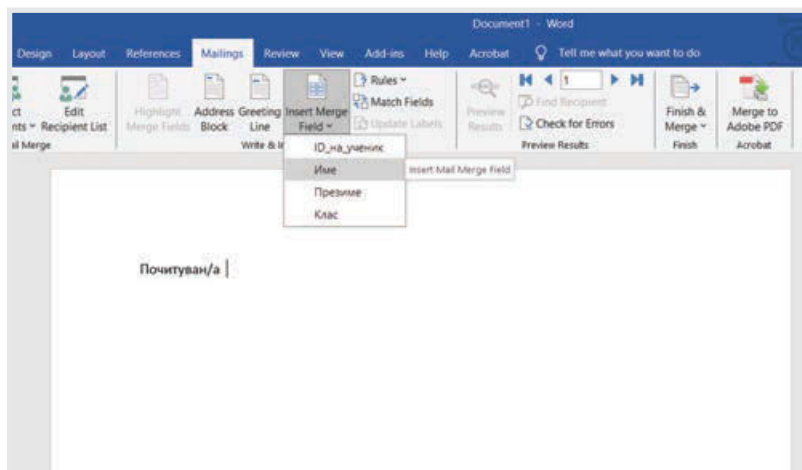
Слика 2 Поврзување со текстуален документ

Полиња за спојување

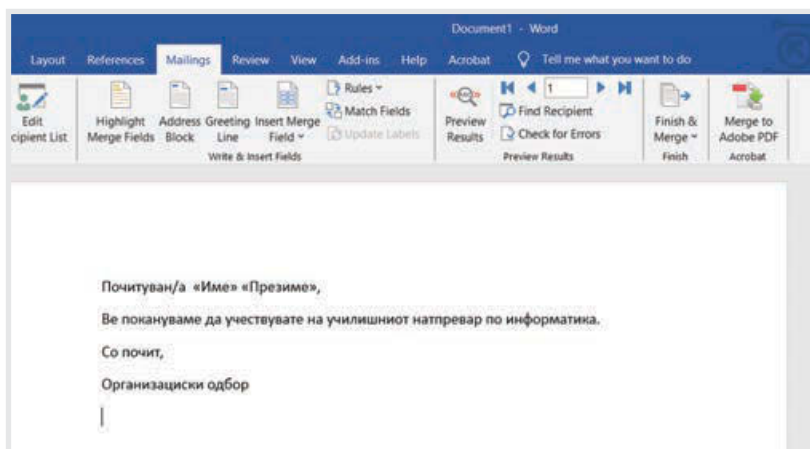
Откако ќе биде документот поврзан со базата на податоци, потребно е да се одредат местата каде што ќе се прикажуваат податоците од записите. За оваа намена се користат полиња за спојување.

Полињата за спојување претставуваат ознаки што го означуваат местото каде што програмата ќе вметне податоци од базата на податоци. Секое поле е поврзано со одредена колона од табелата.

На пример, полето <<Име>> е поврзано со колоната Име, а полето <<Презиме>> со колоната Презиме. При генерирањето на документите програмата автоматски ги заменува овие ознаки со соодветните вредности од секој запис.



Слика 3 Вметнување Merge Fields



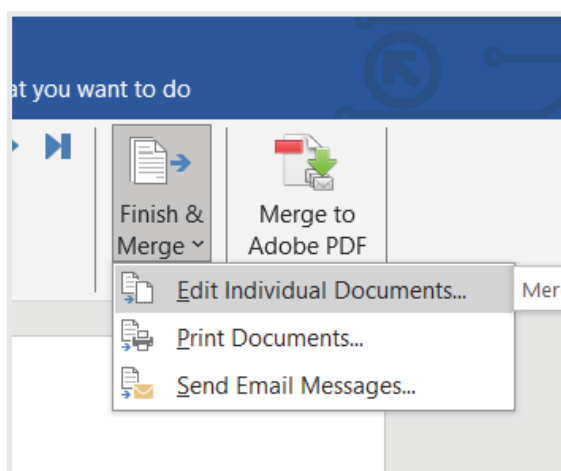
Слика 4 Комплетирано писмо

Генерирање циркуларни писма

Откако ќе бидат поставени сите полиња за спојување, програмата може автоматски да ги генерира документите. За секој запис од базата на податоци се создава посебна верзија на документот.

Доколку базата содржи десет записи, ќе се создадат десет документи. Доколку содржи сто записи, ќе се создадат сто документи. Корисникот не мора рачно да ги менува имињата и другите податоци.

Оваа можност значително ја забрзува работата и овозможува брза подготовка на персонализирани покани, потврди, сертификати, известувања и други документи што содржат податоци за повеќе лица.



Слика 5 Генерирање документи

За да се заврши целата постапка потребно е да се спојат со иконата од слика 5, и тоа да важи за сите записи што ни се потребни.

Практичен пример: Сертификати за натпревар

Да се претпостави дека училиштето организира натпревар по програмирање и дека е потребно да се изработат сертификати за сите учесници.

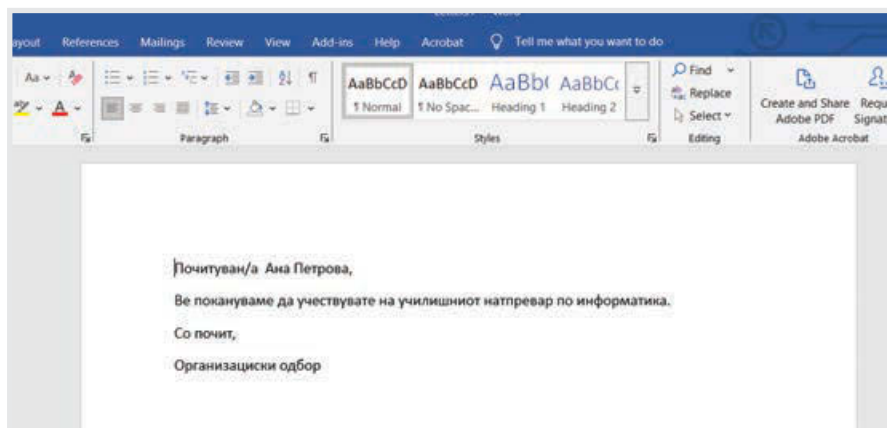
Во базата на податоци се чуваат имињата и презимињата на учениците. Наместо секој сертификат да се уредува посебно, може да се подготви еден шаблон во кој ќе бидат поставени полиња за спојување.

При генерирањето на циркуларното писмо програмата автоматски ќе создаде сертификат за секој ученик. На овој начин за неколку секунди може да се подготват голем број документи со професионален изглед.

Анализа на примерот

Од претходниот пример може да се забележи дека циркуларните писма претставуваат врска меѓу базите на податоци и програмите за обработка на текст. Податоците остануваат во базата на податоци, додека документот претставува шаблон што автоматски се пополнува со потребните информации.

Со користење на циркуларни писма се намалува времето потребно за изработка на документите, се намалува можноста за грешки и се обезбедува единствен изглед на сите документи. Поради овие причини, оваа техника има широка примена во образованието, во администрацијата и во деловното работење.



Слика 6 Финален персонализиран документ



ЗАКЛУЧОК:

Циркуларните писма претставуваат ефикасен начин за автоматско создавање голем број документи што имаат иста содржина, но содржат различни податоци за примачите. Наместо секој документ да се уредува посебно, корисникот подготвува еден шаблон што се поврзува со базата на податоци или со табела.

Со користење на полиња за спојување програмата автоматски ги вметнува потребните информации во документот. На тој начин се заштедува време, се намалува можноста за грешки и се обезбедува единствен изглед на сите документи.

Циркуларните писма имаат широка примена во образованието, во администрацијата и во деловното работење, каде што често е потребно создавање голем број персонализирани документи.



ДАЛИ ЗНАЕШ?

Дали знаеш дека голем дел од сертификатите, дипломите, поканите и известувањата што ги добиваат учениците и вработените денес се создаваат автоматски со помош на циркуларни писма? Наместо секој документ да се пополнува рачно, податоците се преземаат од базата на податоци и се вметнуваат во подготвен шаблон.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува циркуларно писмо?
2. Кои се предностите на циркуларните писма?
3. Што претставува извор на податоци?
4. Како се поврзува документот со базата на податоци?
5. Што се полиња за спојување?
6. Зошто се користат полињата за спојување?
7. Во кои ситуации најчесто се користат циркуларни писма?



ПРАКТИЧНИ ЗАДАЧИ

Задача 1

Креирај табела со следните полиња:

- Име
- Презиме
- Паралелка

Внеси податоци за најмалку пет ученици.

Задача 2

Подготви шаблон за покана за училиштен натпревар.

Во шаблонот предвиди места за:

- име;
- презиме;
- паралелка.

Задача 3

Поврзи го документот со табелата и вметни соодветни полиња за спојување.

Задача 4

Генерирај циркуларно писмо за сите ученици од табелата.

Задача 5

Подготви шаблон за сертификат што автоматски ќе ги користи податоците од базата на податоци.



ИСТРАЖУВАЧКА ЗАДАЧА

Истражи во кои области најчесто се користат циркуларни писма.

За секоја област наведи:

- вид на документ;
- извор на податоци;
- придобивки од користењето на циркуларни писма.



ЗАПОМНИ ШТО НАУЧИ

- Циркуларното писмо претставува документ поврзан со извор на податоци.
- Изворот на податоци може да биде база на податоци или табела.
- Полињата за спојување служат за автоматско вметнување на податоците.
- Еден шаблон може да создаде голем број различни документи.
- Циркуларните писма заштедуваат време и ја намалуваат можноста за грешки.
- Оваа техника се користи за покани, сертификати, потврди и известувања.





ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Современите системи овозможуваат циркуларните писма да се користат не само за печатење документи, туку и за автоматско испраќање електронска пошта. На овој начин еден шаблон може автоматски да се испрати до голем број корисници, при што секој примач добива персонализирана порака со личните податоци.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

Заокружи го точниот одговор

1. Циркуларното писмо служи за:

- а) инсталирање програми
- б) создавање повеќе персонализирани документи
- в) форматирање диск
- г) креирање база на податоци

2. Полињата за спојување:

- а) ги бришат податоците
- б) служат за автоматско вметнување податоци
- в) создаваат нова табела
- г) печатат извештаи

3. Извор на податоци може да биде:

- а) база на податоци
- б) табела
- в) база на податоци или табела
- г) оперативен систем

Точно или неточно

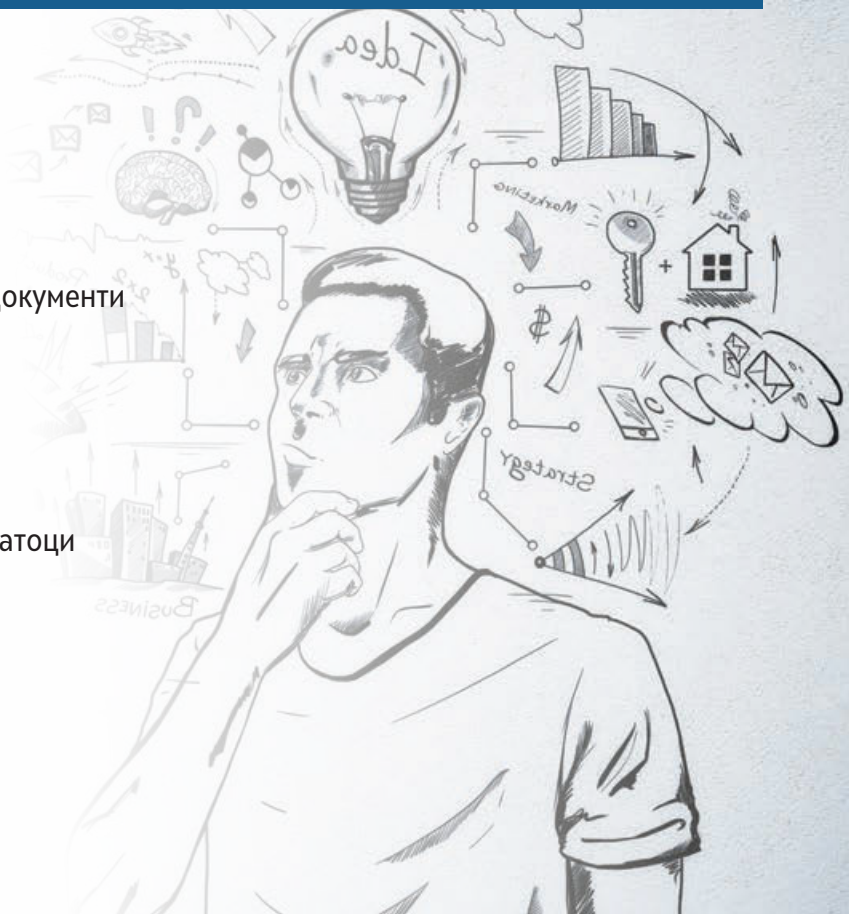
- 1. Циркуларното писмо користи еден шаблон за повеќе документи.
- 2. Секој документ мора рачно да се пополнува.
- 3. Податоците може автоматски да се преземат од базата на податоци.
- 4. Полињата за спојување се заменуваат со вистински податоци при генерирањето на документите.

АНАЛИЗА

Училиштето треба да изработи 300 сертификати за учесници на натпревар.

Објасни:

- зошто циркуларното писмо е подобро решение од рачното пополнување;
- кои податоци би биле потребни во базата на податоци;
- кои полиња за спојување би се користеле во сертификатот.



3.9



SQL – ОСНОВЕН ЈАЗИК ЗА РАБОТА СО БАЗИ НА ПОДАТОЦИ

ВЕЌЕ ЗНАЕШ ДА...

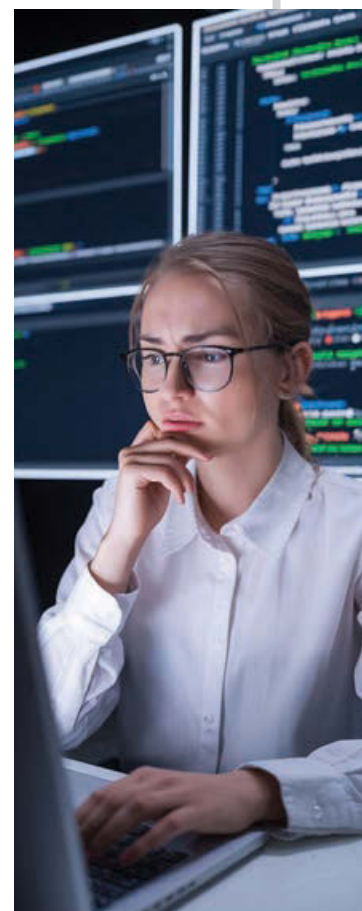
- креираш табели во базата на податоци;
- определуваш полиња и типови на податоци;
- користиш примарен клуч;
- внесуваш податоци преку форми;
- пребаруваш информации со прашалници;
- подготвуваш извештаи;
- користиш податоци од базата на податоци при изработка на циркуларни писма.

ПРАШАЊА ЗА РАЗМИСЛУВАЊЕ

- Како комуницира компјутерот со базата на податоци?
- Како може да се побараат само одредени информации од табелата?
- Како може да се подредат резултатите од пребарувањето?
- Дали постои стандарден јазик што се користи во различни бази на податоци?
- Како се изведуваат пребарувања во големи бази на податоци?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што претставува SQL;
- користиш основни SQL наредби;
- извршуваш едноставни пребарувања;
- поставуваш услови при пребарување;
- подредуваш резултати;
- анализираш резултати добиени со SQL.



Во претходните лекции беше покажано како може да се внесуваат податоците преку форми, да се пребаруваат со прашалници и да се прикажуваат преку извештаи. Во повеќето програми овие активности може да се извршуваат со графички алатки и менија. Меѓутоа, во заднината постои стандарден јазик што овозможува директна комуникација со базата на податоци.

Овој јазик се нарекува SQL (Structured Query Language). Тој претставува меѓународен стандард за работа со релациони бази на податоци и се користи во голем број системи за управување со податоци. Благодарение на SQL, корисниците и програмерите можат да креираат табели, да внесуваат податоци, да пребаруваат информации и да управуваат со содржината на базата на податоци.

Иако современите програми нудат графички алатки за работа со базите на податоци, многу од нивните функции всушност генерираат SQL наредби. Поради тоа, познавањето на основите на SQL овозможува подобро разбирање на начинот на кој функционираат базите на податоци.

Што претставува SQL?

SQL е специјализиран јазик што се користи за работа со релациони бази на податоци. Неговото име потекнува од англискиот израз Structured Query Language, што во превод значи 'структуриран јазик за поставување прашања' кон базата на податоци.

Со SQL може да се побараат информации од табелите, да се додаваат нови записи, да се менуваат постојни податоци и да се бришат информации што веќе не се потребни. Поради оваа разновидност, SQL претставува еден од најважните јазици во областа на базите на податоци.

Во оваа лекција ќе бидат обработени само основните SQL наредби што овозможуваат пребарување и селектирање на податоците. Операциите се извршуваат врз прашалник. За секоја од следните наредби прво се креира табела, па потоа прашалник за да може да се користи SQL.

Наредбата SELECT

Најчесто користената SQL наредба е SELECT. Таа служи за прикажување на податоците од една или од повеќе табели.

Да се разгледа табелата Ученици.

Име	Презиме	Паралелка
Ана	Петрова	II-1
Марко	Стојанов	II-2
Елена	Николовска	II-1

За прикажување на сите записи се користи:

```
SELECT * FROM Ucenici;
```

Свездичката (*) означува дека треба да се прикажат сите полиња од табелата.

Ако е потребно да се прикажат само имињата и презимињата, може да се напише:

```
SELECT Ime, Prezime  
FROM Ucenici;
```

Наредбата WHERE

Во многу ситуации е потребно да се прикажат само записите што исполнуваат определен услов. За оваа намена се користи наредбата WHERE.

Да претпоставиме дека е потребно да се прикажат само учениците од паралелката II-1.

```
SELECT *  
FROM Ucenici  
WHERE Paralelka = 'II-1';
```

Системот ќе ги прикаже само записите што го исполнуваат наведениот услов.

Условите овозможуваат значително побрзо пребарување на податоците и се користат во речиси сите реални бази на податоци.

Наредбата ORDER BY

Понекогаш податоците треба да бидат прикажани по определен редослед. За оваа намена се користи наредбата ORDER BY.

Да претпоставиме дека е потребно учениците да бидат подредени по презиме.

```
SELECT *
FROM Ucenici
ORDER BY Prezime;
```

Резултатите ќе бидат прикажани по азбучен редослед.

Можно е подредување и во опаѓачки редослед:

```
SELECT *
FROM Ucenici
ORDER BY Prezime DESC;
```

Клучниот збор DESC означува опаѓачки редослед.

Наредбата DELETE

Понекогаш е потребно одредени записи да бидат отстранети од табелата. За оваа намена се користи наредбата DELETE.

На пример:

```
DELETE FROM Ucenici
WHERE Paralelka = 'IV-4';
```

Оваа наредба ќе ги избрише сите записи што го исполнуваат условот.

Поради можноста за губење на податоците, DELETE треба да се користи внимателно и секогаш со јасно дефиниран услов.

Практичен пример

Да се разгледа база на податоци што содржи информации за ученици.

Потребно е да се прикажат сите ученици со просек поголем од 4.50 и резултатите да бидат подредени според презиме.

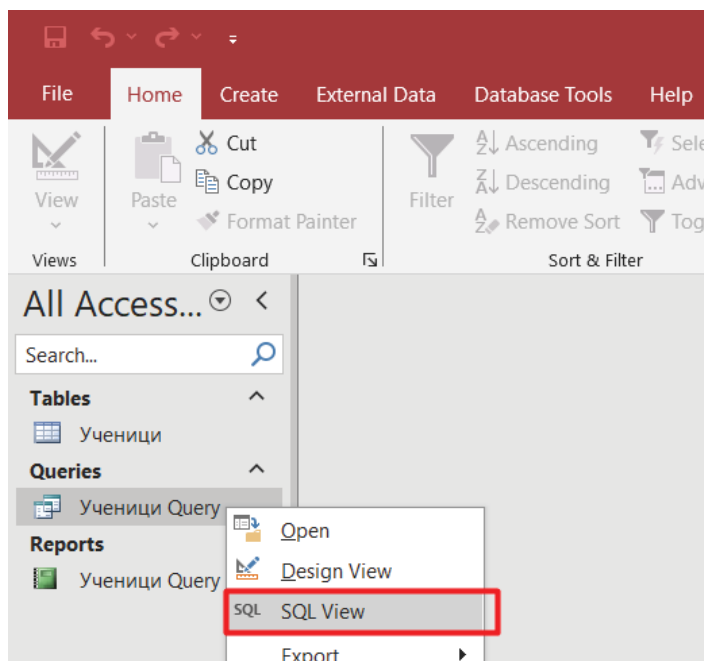
```
SELECT *
FROM Ucenici
WHERE Prosek > 4.50
ORDER BY Prezime;
```

Во една единствена наредба се комбинирани пребарување, услов и подредување на резултатите. Ова претставува една од најголемите предности на SQL.

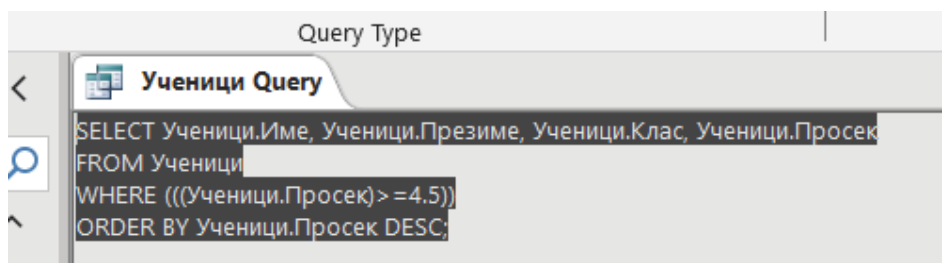
Анализа на примерот

Од претходниот пример може да се забележи дека SQL овозможува едноставно и прецизно управување со податоците. Наместо корисникот рачно да ги пребарува записите, доволно е да постави соодветна SQL наредба и системот автоматски ќе ги пронајде потребните информации.

Благодарение на својата едноставна структура и големите можности, SQL денес претставува стандард за работа со релациони бази на податоци и е застапен во речиси сите современи информатички системи.



Слика 1 SQL поглед во MS Access



Слика 2 Наредби во SQL

SQL претставува стандарден јазик за работа со релациони бази на податоци. Со негова помош може да се пребаруваат, да се подредуваат и да се обработуваат податоците што се чуваат во табелите. Иако многу програми нудат графички алатки за работа со базите на податоци, во нивната основа најчесто се користат SQL наредби.

Наредбите SELECT, WHERE и ORDER BY овозможуваат брзо пронаоѓање и организирање на информациите, додека DELETE се користи за отстранување на непотребни записи. Благодарение на овие можности, SQL претставува важна алатка за работа со податоци во современите информациски системи.

Со изучувањето на SQL се заокружува темата за бази на податоци, затоа што учениците добиваат увид не само во организацијата на податоците туку и во начинот на кој комуницираат системите со нив.



ДАЛИ ЗНАЕШ?

Дали знаеш дека SQL се користи повеќе од четири децении и денес претставува еден од најраспространетите програмски јазици во светот? Речиси сите големи информатички системи, веб-апликации и мобилни сервиси користат SQL за работа со податоците.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ

1. Што претставува SQL?
2. Од кои зборови е составена скратеницата SQL?
3. За што служи наредбата SELECT?
4. За што се користи наредбата WHERE?
5. Каква улога има ORDER BY?
6. Кога се користи наредбата DELETE?
7. Зошто се смета SQL за стандарден јазик за работа со бази на податоци?



ПРАКТИЧНИ ЗАДАЧИ

Задача 1

Напиши SQL наредба со која ќе се прикажат сите записи од табелата Ucenici.

Задача 2

Напиши SQL наредба со која ќе се прикажуваат само полињата:

- Ime
- Prezime

од табелата Ucenici.

Задача 3

Напиши SQL наредба што ќе ги прикажува само учениците од паралелката II-1.

Задача 4

Напиши SQL наредба што ќе ги прикажува сите ученици подредени по презиме.

Задача 5

Напиши SQL наредба што ќе ги прикажува учениците со просек поголем од 4.50.

Задача 6

Објасни што ќе направи следната SQL наредба:

```
SELECT *
FROM Ucenici
WHERE Prosek > 4.50
ORDER BY Prezime;
```



ИСТРАЖУВАЧКА ЗАДАЧА

Истражи кои системи за управување со бази на податоци користат SQL.

За најмалку три системи пронајди:

- име на системот;
- производител;
- област на примена;
- интересен податок за системот.

Подготви краток извештај или презентација.



ЗАПОМНИ ШТО НАУЧИ

- SQL е стандарден јазик за работа со релациони бази на податоци.
- SELECT служи за прикажување податоци.
- WHERE овозможува поставување услови при пребарување.
- ORDER BY се користи за подредување на резултатите.
- DELETE служи за бришење записи.
- SQL овозможува брза и ефикасна работа со големи количини податоци.
- Голем број современи системи користат SQL во својата работа.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Покрај основните наредби, SQL содржи голем број напредни можности. Со него може да се поврзуваат повеќе табели, да се извршуваат статистички пресметки, да се групираат податоци и да се создаваат сложени извештаи. Во професионалната примена SQL претставува една од најважните технологии во развојот на информациски системи и веб-апликации.



ПРОВЕРИ ГО СВОЕТО ЗНАЕЊЕ

Заокружи го точниот одговор

1. SQL претставува:

- а) оперативен систем
- б) програмски јазик за графика
- в) јазик за работа со бази на податоци
- г) антивирусна програма

2. Наредбата SELECT служи за:

- а) бришење записи
- б) прикажување податоци
- в) печатење документи
- г) креирање форми

3. Наредбата WHERE служи за:

- а) подредување податоци
- б) поставување услов
- в) бришење табели
- г) креирање извештаи

4. ORDER BY се користи за:

- а) сортирање на резултатите
- б) бришење податоци
- в) креирање полиња
- г) поврзување табели

Точно или неточно

- 1. SQL се користи за работа со бази на податоци.
- 2. SELECT ги брише записите од табелата.
- 3. WHERE овозможува селектирање на записи според услов.
- 4. ORDER BY овозможува подредување на резултатите.
- 5. DELETE се користи за отстранување записи.

ДОПОЛНИ

- 1. Наредбата _____ служи за прикажување на податоците.
- 2. Наредбата _____ служи за подредување на резултатите.
- 3. Наредбата _____ служи за бришење записи.

АНАЛИЗА

Да се разгледа табелата:

Име	Презиме	Просек
Ана	Петрова	5.00
Марко	Стојанов	4.20
Елена	Николовска	4.80

Напиши SQL наредба со која ќе се прикажат учениците со просек поголем од 4.50 подредени по презиме.

Објасни што ќе биде прикажано како резултат.

ЗАВРШНА МИСЛА

Базите на податоци претставуваат основа на современите информациски системи. Во текот на оваа тема беше покажано како се организираат податоците во табели, како се внесуваат преку форми, како се пребаруваат со прашалници, како се прикажуваат преку извештаи, како се користат во циркуларни писма и како се обработуваат со SQL. Овие знаења се важна основа за понатамошно изучување на информатиката и за развојот на современи софтверски решенија.



TEMA 4



ТЕМА 4: МУЛТИМЕДИЈА И КОМПЈУТЕРСКА ГРАФИКА

Мултимедијата и компјутерската графика претставуваат важен дел од современата технологија и од секојдневниот живот. Тие вклучуваат користење на текст, звук, слики, видео и анимација, но и создавање и обработка на визуелни елементи со помош на компјутер. Благодарение на компјутерската графика, можеме да создадеме реалистични слики, игри, филмови и дигитални дизајни. Мултимедијата ги прави информациите поинтересни и полесни за разбирање, особено во образованието и во сферата на забавата. Денес, овие технологии се користат во многу области, како што се: маркетинг, медицина, архитектура и игри. Поради тоа, познавањето на мултимедијата и компјутерската графика е важно за младите, кои растат во дигиталниот свет.

НОВИ ПОИМИ:

мултимедија, елементи – текст, слика, звук, видео, графика, растерска графика, векторска графика, пиксел, резолуција, селекција, слоеви, видливост.

ВЕЌЕ ЗНАЕШ ДА:

- уредуваш слики со помош на основни алатки;
- копираш и преместуваш делови од една слика, да бришеш делови од слика, да доделуваш ефекти на слика.
-

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ

- Наброј најмалку три алатки за уредување на слики.
- Што значи „копирање дел од слика“?
- Кои се најчестите ефекти што може да се применат на слика?
- Објасни ја разликата помеѓу „копирање“ и „преместување“ дел од слика.
- Објасни како ја менуваат ефектите визуелната појава на сликата.
- Копирај објект од една слика и постави го во друга.
- Додај ефект „црно-бело“ на дадена слика.
- Спореди оригинална и уредена слика – кои се разликите?
- Анализирај како преместувањето на објекти ја менува композицијата на сликата.
- Објасни зошто бришењето на одредени елементи може да ја подобри сликата.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш за мултимедија, нејзините основни елементи и примена, како и за карактеристиките на растерска и векторска графика;
- применуваш основни алатки и техники за обработка на слика, вклучително корекции, слоеви и комбинирање графички елементи;
- користиш алатки со вештачка интелигенција за генерирање и трансформација на слики, да формулира едноставни инструкции (prompt) и да ги анализира добиените резултати;
- креираш едноставни графички решенија (постер, банер или лого) со примена на изучените техники и ВИ алатки.

4.1



МУЛТИМЕДИЈА

Во современиот дигитален свет, информациите ретко се пренесуваат само на еден начин. Кога читаме електронска книга, гледаме образовно видео, следиме презентација, користиме апликација или пребаруваме содржини на интернет, најчесто истовремено се среќаваме со текст, слики, звук, видео и интерактивни елементи. Таквиот начин на претставување на содржините му е близок на човекот, бидејќи и во секојдневниот живот не ги процесираме појавите изолирано, туку преку повеќе сетилиа и искуства одеднаш. Ваквиот начин на пренесување информации е основа за разбирање на поимот мултимедија.

Поимот **мултимедија** може да се објасни како начин на претставување информации преку комбинирање на повеќе елементи, како што се: текст, слика, звук, видео и анимација. Овие елементи се користат заедно за да се објасни, да се прикаже или да се пренесе одредена содржина на поцелосен начин.

Зборот мултимедија е составен од два дела: мулти, што значи 'повеќе', и медија, што се однесува на медиуми, односно начини или средства преку кои се пренесуваат информации.

Во информатиката, мултимедијата се поврзува со дигитални содржини и апликации во кои се комбинираат различни елементи, како: текст, слика, звук, видео, анимација и интерактивни делови. Кога се овие елементи поврзани во една целина, се создава мултимедијална содржина или мултимедијална апликација. Примери за мултимедијални содржини се: презентација, веб-страница и електронски учебник, а примери за мултимедијални апликации се образовна апликација и компјутерска игра.

Деловите од кои е составена една мултимедијална содржина се нарекуваат **мултимедијални елементи**. Основни **мултимедијални елементи** се: текст, слика, звук, видео, анимација и интерактивни елементи. Секој од нив има своја улога: текстот дава информации, објаснувања или упатства; сликата визуелно прикажува поим, предмет или појава; звукот пренесува говор, музика или звучни сигнали; видеото прикажува движење, постапка или настан, а интерактивните елементи му овозможуваат на корисникот да избира, да одговара или да управува со делови од содржината.

Важно е да се знае дека мултимедијата не значи само поставување повеќе елементи на едно место. Тие треба да бидат поврзани со темата и да имаат заедничка цел. На пример, презентација со текст, слики и кратко видео за една наставна тема е мултимедијална содржина. Но, ако текстот, сликите и звукот се ставени без ред и не се поврзани со темата, тие нема да помогнат во разбирањето на информацијата.

Мултимедијата има широка примена во секојдневниот живот, затоа што овозможува информациите да се претстават на разбирлив, интересен и практичен начин. Таа се користи во различни области, како на пример:

- Во образованието, мултимедијата помага наставните содржини да се објаснат и да се прикажат на начин што е полесен за разбирање. На пример, наместо ученикот само да чита за Сончевиот Систем, може да гледа анимација за движењето на планетите, да слуша објаснување и да одговори на интерактивни прашања. Така учењето станува поедноставно и поактивно.

- Во медицината, мултимедијата може да помогне при објаснувањето на различни здравствени состојби, за функционирањето на органите или за определени медицински постапки. На пример, лекарот може да користи снимки, 3Д прикази, видеа и кратки објаснувања за да може пациентот појасно да разбере што се случува во неговото тело или како ќе се изведе одредена постапка.

- Во туризмот, мултимедијата им помага на луѓето да запознаат место пред да го посетат. На пример, туристичка веб-страница или апликација може да содржи фотографии, мапи, видеа, виртуелни прошетки и кратки информации за знаменитости, хотели или музеи.

- Во инженерството, мултимедијата се користи за поедноставно претставување на сложени проекти и технички системи. На пример, работата на една машина може да се прикаже преку дигитална симулација што содржи анимација на движењето, текстуални објаснувања, звучни насоки и можност корисникот да избере кој дел од машината сака да го разгледа.

- Во спортот, мултимедијата може да се користи за анализа на спортски настани. На пример, анализата на еден спортски настан може да содржи видеоснимки, забавени повторувања, графикони, статистички податоци и објаснувања за движењето на играчите.

Пример за примена на мултимедијата се и компјутерските игри. Една компјутерска игра може да содржи слика, звук, движење, текстуални инструкции и интерактивни елементи преку кои играчот управува со дејството.

Мултимедијата е дел од современиот начин на учење, работа, информирање и решавање практични задачи. Таа не е поврзана само со компјутери и мобилни уреди, туку и со начинот на кој денес се создаваат, се прикажуваат и се користат различни информации.



ЗАКЛУЧОК:

Мултимедијата претставува важен дел од современиот дигитален свет, овозможувајќи интеграција на различни типови содржини. Со неа информациите стануваат појасни, поинтересни и полесни за разбирање. Преку препознавање на нејзините елементи и примена во реални ситуации, се развиваат креативноста и дигиталните вештини. Овие знаења помагаат поефикасно да се комуницира и да се учи во современото општество.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што е мултимедија?
2. Наброј најмалку четири елементи на мултимедија.
3. Објасни зошто е важна мултимедијата во секојдневниот живот.
4. Објасни како сликите и звукот заедно ја подобруваат презентацијата.
5. Наведи два конкретни примери каде што се користи мултимедија во образованието.
6. Определи кои елементи на мултимедија се користат во едно YouTube видео.
7. Анализирај кои мултимедијални елементи се присутни во една онлајн игра.
8. Спореди два различни примери на мултимедија (на пр. филм и веб-страница) според нивните елементи.
9. Процени кој е најважниот елемент на мултимедија и објасни зошто.
10. Наведи пример од секојдневниот живот во кој мултимедијата ти помогнала подобро да разбереш некоја информација. Објасни кои елементи биле употребени. решавање конкретен проблем во таа област
11. Избери една област, на пример медицина, туризам, спорт или инженерство, и предложи како може да се употреби мултимедијата во соодветната област.





4.2

ВЕКТОРСКА И РАСТЕРСКА ГРАФИКА

Во секојдневниот живот често се среќаваме со фотографии, цртежи, икони, мапи, шеми, логоа и други визуелни прикази. Тие можат да пренесат порака, да претстават предмет, да прикажат податок или да помогнат појасно да се следи одредено упатство.

Претставувањето информации, идеи или предмети преку визуелни елементи, како слики, линии, форми, бои и симболи се нарекува **графика**.

Кога се создава, се обработува и се прикажува графиката со помош на компјутер, станува збор за **компјутерска графика**.

Во зависност од тоа од кои основни елементи е составена сликата, најчесто разликуваме два основни типа компјутерска графика: **растерска** и **векторска графика**.

Во векторската графика сликата е претставена со линии, криви, форми и други графички објекти. Овие елементи не се запишуваат како мрежа од пиксели, туку се опишуваат со математички правила. Компјутерот ги користи тие правила за да ја прикаже сликата на екранот.

На пример, една кружница во векторска графика не е запишана како голем број мали точки, туку како форма со одредена положба, големина, линија и боја. Затоа векторската слика може поедноставно да се зголемува или да се намалува без да стане заматена или составена од мали квадратчиња. Линиите и формите остануваат јасни и остри и кога сликата се прикажува зголемена.

Векторската графика најчесто се користи за логоа, икони, знаци, симболи, технички цртежи, илустрации и дизајни што треба да се користат во различни големини. На пример, едно лого може да се користи мало на визит-картичка, но и многу големо – на постер или билборд, без да изгледа заматено.

Предност на векторската графика е тоа што може да се зголемува и да се намалува без губење на јасноста на линиите и формите. Исто така, одделните елементи на сликата, како линии, форми и бои, може едноставно да се менуваат. Недостаток е тоа што векторската графика не е најпогодна за прикажување фотографии и реални слики со многу нијанси, сенки и ситни детали.

За работа со векторска графика се користат програми како што се Inkscape, Adobe Illustrator, CorelDRAW, Figma и LibreOffice Draw.

Во растерската графика сликата е претставена како мозаик од мали обоени точки што се нарекуваат пиксели. Пикселот претставува најмал елемент од кој е составена една растерска слика. Кога се пикселите многу мали и густо поставени, човечкото око не ги забележува одделно, туку ги поврзува во една целосна слика.

Кај растерските слики важен поим е резолуцијата. Во секојдневна употреба, таа често се поврзува со димензиите на сликата во пиксели, на пример 1920 × 1080 пиксели. Тоа значи дека сликата има 1920 пиксели по ширина и 1080 пиксели по висина. При печатење и скенирање, резолуцијата најчесто се изразува како број на точки по инч, на пример 300 dpi (dots per inch). Поголем број пиксели овозможува сликата да содржи повеќе детали, а поголема густина на точки при печатење е важна за појасен отпечаток.

Предност на растерската графика е тоа што може добро да прикаже сложени слики со многу бои, сенки и постепени премини меѓу боите. Затоа е таа многу корисна за фотографии и за реалистични слики.

Недостаток на растерската графика е тоа што при големо зголемување сликата може да изгуби од квалитетот. Бидејќи е составена од пиксели, тие при зголемување може да станат видливи, па сликата да изгледа заматена или составена од мали квадратчиња. Исто така, растерските слики со висока резолуција можат да зафаќаат повеќе меморија.

За работа со растерска графика се користат програми за цртање и обработка на слики, како што се Microsoft Paint, Adobe Photoshop, GIMP, Krita и Paint.NET.

Графичките датотеки може да се зачуваат во различни формати. Од форматот може да зависи големината на датотеката, квалитетот на сликата, можноста за просирна заднина и начинот на кој сликата може да се користи или да се уредува.

Основни формати на слики се:

- JPG или JPEG е растерски формат што најчесто се користи за дигитални фотографии и слики со многу бои и детали. Погоден е за фотографии што се поставуваат на интернет или се испраќаат преку електронска пошта, но не е погоден за слики со просирна заднина.
- PNG е растерски формат што ја зачувува сликата без загуба на квалитетот при компресирање. Овој формат поддржува просирна заднина, па често се користи за икони, слики за веб-страници, екранизирани слики, графички елементи и слики кај кои е важно рабовите и деталите да останат јасни.
- SVG е формат што најчесто се користи за векторска графика. Погоден е за логоа, икони, симболи и едноставни илустрации што треба да се прикажуваат во различни големина без да станат заматени или составени од мали квадратчиња.

При избор на формат треба да се внимава на намената на сликата. За фотографии најчесто се користи JPG, за слики со просирна заднина и јасни графички елементи се користи PNG, а за логоа, икони и векторски илустрации се користи SVG.

Други формати на графички датотеки се следните:

- За растерска графика: GIF, BMP, TIFF, RAW, PCX, TGA.
- За векторска графика: AI, EPS, CDR, DXF, WMF, EMF.

Разбирањето на разликата помеѓу растерската и векторската графика помага при правилно користење, уредување и зачувување на сликите. Кога се знае како е изградена една слика и во кој формат е зачувана, поедноставно може да се избере соодветен начин за нејзина обработка, печатење или споделување.



ЗАКЛУЧОК:

Разбирањето на разликата помеѓу растерската и векторската графика е клучно за правилна употреба на дигитални слики. Познавањето на поимите пиксел и резолуција помага да се оцени квалитетот на сликата. Согледувањето на предностите и недостатоците на двата типа графика овозможува избор на најсоодветен пристап во зависност од потребата. Познавањето на форматите JPG, PNG и SVG придонесува за поефикасно користење и зачувување на графичките содржини.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Спореди растерска и векторска графика според начинот на кој е изградена сликата.
2. Објасни зошто дигиталната фотографија најчесто се зачувува како растерска слика, а логото како векторска графика.
3. Анализирај што ќе се случи ако растерска слика со мала резолуција се зголеми за печатење на голем постер.
4. Спореди ги форматите JPG и PNG. Во која ситуација би избрал/а JPG, а во која PNG? Објасни зошто.
5. Спореди растерска и векторска слика според нивното однесување при зголемување (zoom).
6. Анализирај како влијаат пикселите врз квалитетот на една фотографија.
7. Објасни ја разликата помеѓу JPG и PNG во однос на квалитетот и големината на датотеката.
8. Прочени кој формат е најсоодветен за следниве ситуации: фотографија за веб-страница, лого за училиште и икона со просирна заднина. Образложи го изборот.
9. Замисли дека треба да подготвиш постер за училиштен настан. Кои елементи би ги изработил/а како растерска, а кои како векторска графика? Објасни го изборот.



4.3



ОСНОВИ НА ОБРАБОТКА НА СЛИКА

Обработката на слика претставува процес на изменување и приспособување на дигитална слика со цел да ѝ се подобри изгледот, квалитетот или употребната вредност. Таа опфаќа различни постапки, како што се отстранување на непотребни делови, промена на големината, приспособување на светлината, контрастот и боите, додавање текст или други елементи, како и примена на одредени ефекти. Со овие постапки сликата може да се подготви за печатење, презентација, веб-страница или за друг вид дигитална содржина. Затоа, познавањето на основните техники за обработка на слика е важно за правилно и ефикасно користење на визуелните информации.

Една од програмите што се користат за работа со растерска графика е ГИМП (GIMP – GNU Image Manipulation Program). GIMP е слободна софтверска апликација за обработка на слики, достапна под Општата јавна лиценца GNU GPL. Оваа лиценца им овозможува на корисниците слободно да ја користат програмата, да го проучуваат и да го менуваат нејзиниот изворен код, како и да ги споделуваат изменетите верзии под условите на истата лиценца. ГИМП нуди работна околина во која корисникот може да отвора, да уредува и да зачувува слики во различни формати. Содржи голем број алатки за работа со слики, организирани во менија, панели и палети, што овозможува постепено и прегледно извршување на задачите. Посебно значајна можност е работата со слоеви, преку која различни делови од сликата може да се уредуваат одделно.

4.3.1. РАБОТНА ОКОЛИНА НА ГИМП

Во режимот на еден прозорец (Single-Window Mode), работната околина на GIMP најчесто е организирана во три главни делови: лев панел со алатки, централна работна површина во која се прикажува и се уредува сликата и десен панел со дополнителни дијалози, како што се слоеви, канали, патеки и други поставки. Ширината на панелите може да се приспособи со повлекување на граничната линија меѓу нив.





Слика 4.1: Работна околина на ГИМП

Панелот со алатки содржи:

1. алатки за обработка на слика (може да се избераат и преку менито Tools);
2. икона за избирање на предна (главна, примарна) и заднинска (секундарна) боја во облик на бел и црн правоаголник;
3. својства на избраната алатка (помошна алатка Tool Options).

Помошниот панел на десната страна од работното опкружување е поделен на горна (означена со број 4) и долна група (означена со број 5) помошни прозорци. Тие нудат дополнителни можности за обработка на сликата како што се, на пример, работа со слоеви, канали, четки и друго. Помошните прозорци што не се користат може да се затворат со наредбата Close Tab од менито кое се отвора со клик на малата стрелка во горниот десен агол, а повторно може да се отвори со избор на Windows -> Dockable Dialogs.

Централниот дел од работното опкружување го зазема панелот за слика. На врвот на овој панел се поставени картички со минијатурни прикази на вчитани слики.

Над сликата е поставен хоризонтален линијар, а на левата страна е поставен вертикален линијар. Под сликата и од нејзината десна страна се наоѓаат лизгачи (хоризонтален и вертикален).

Статусната лента на дното од панелот со слики е поделена на четири дела: координати на покажувачот на глумчето (се прикажува само кога покажувачот на глумчето е во панелот со слики), паѓачка листа за избор на мерна единица, поле за избор на ниво на зумирање во процент и област што ги прикажува името и големината на датотеката од која е вчитана сликата.

Дотерувањето на работното опкружување се врши преку прозорецот Preferences, кој се наоѓа во менито Edit.

4.3.2. ОТВОРАЊЕ, КРЕИРАЊЕ НОВА СЛИКА, СНИМАЊЕ, ЕКСПОРТИРАЊЕ СЛИКИ

Наредбите за отворање (Open), креирање нова слика (New), снимање (Save, Save As) се наоѓаат во менито File. Сликите во ГИМП се зачувуваат во формат XCF. Во друг формат може да се сочуваат преку наредбата Export As.

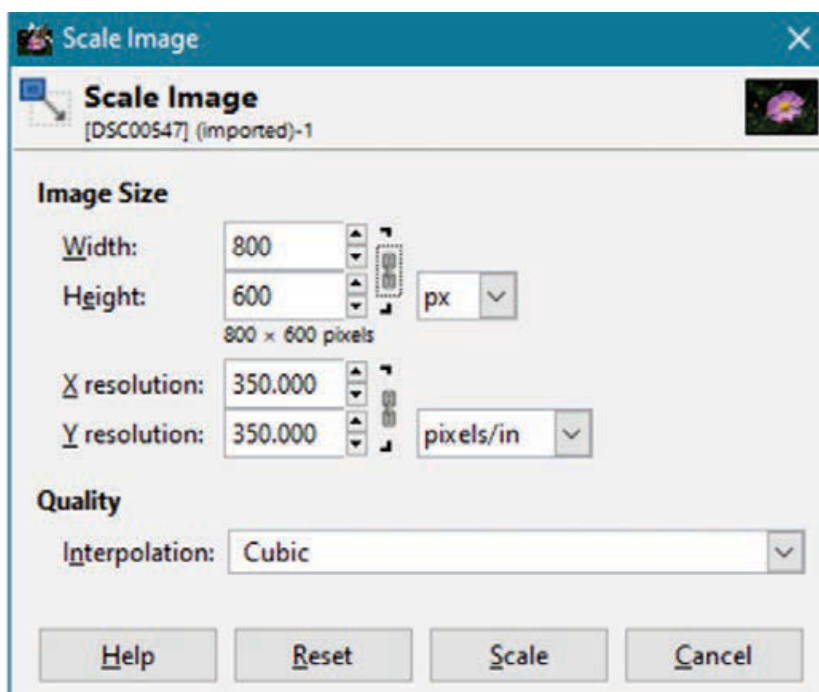
4.3.3. ПРОМЕНА НА МОДЕЛОТ НА БОИТЕ НА СЛИКАТА

Може да се направи со користење на ставката Mode во менито Image. GIMP нуди три модели:

- RGB – стандардниот модел на бои со 24-битна длабочина.
- Grayscale – секој пиксел на сликата е претставен со нијанса на сива боја.
- Indexed – бојата на секој пиксел на сликата е одредена од индексот (редоследен број) на боја од палетата на бои што може да содржи до 256 бои.

4.3.4. ПРОМЕНА НА ГОЛЕМИНАТА НА СЛИКАТА

Може да се направи преку Image -> Scale Image



Слика 4.2: Image -> Scale Image

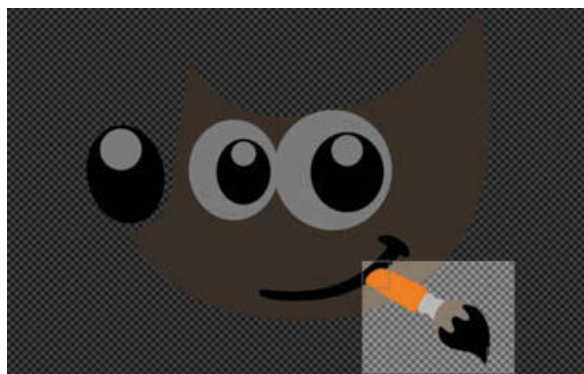
Новата ширина (Width) и висина (Height) може да се наведат во неколку мерни единици, како на пример, во пиксели (px, pixels) или во проценти (% , percent).

4.3.5. ПРОМЕНА НА ГОЛЕМИНАТА НА ПЛАТНОТО

Платното (Canvas) ја одредува видливата површина на сликата. Стандардно, големината на платното се совпаѓа со големината на сликата. Ако е платното поголемо од сликата, околу неа се создава дополнителен простор. Од друга страна, ако е платното помало од сликата, деловите од сликата што се надвор од платното нема да бидат видливи, но овие делови не се губат, туку ќе се појават ако се зголеми платното. Прозорецот за промена на големината на платното се отвора со избор на Image -> Canvas Size.

4.3.6. ОТСЕКУВАЊЕ НА ДЕЛ ОД СЛИКАТА

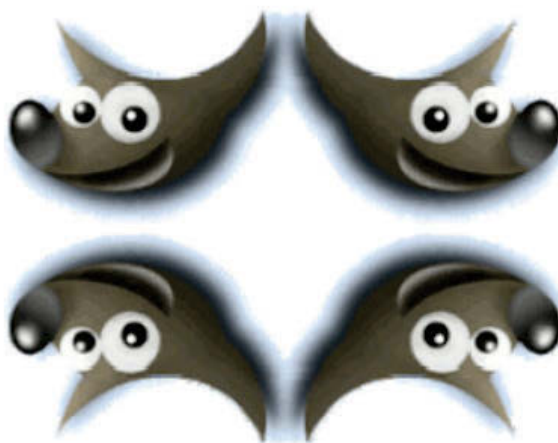
Се врши со избор на алатката Crop (). Се црта областа за отсекување и се притиска Enter за потврда. Резултатот ќе биде нова слика што ја содржи само внатрешноста на областа за отсекување. Сè што е надвор од таа област ќе биде отфрлено. Отсекувањето може да се откаже со притискање на копчето Esc.



Слика 4.3: Отсекување

4.3.7. ПРЕВРТУВАЊЕ НА СЛИКАТА

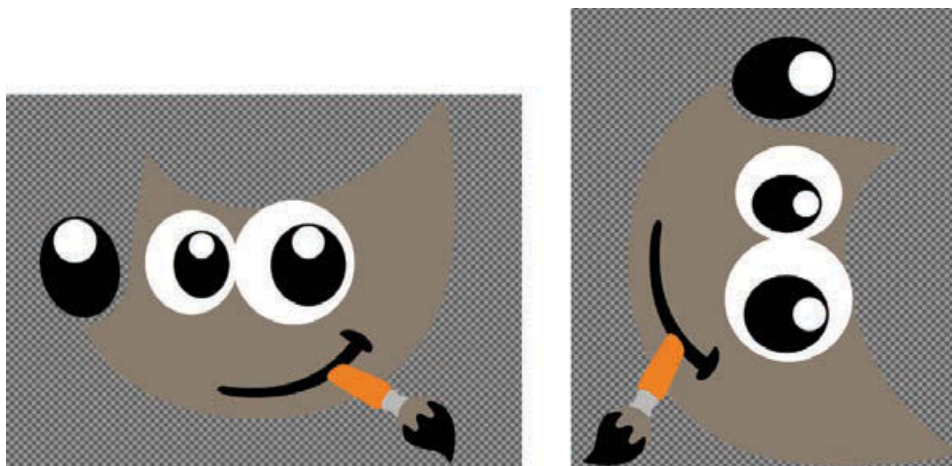
Превртувањето на сликата, хоризонтално или вертикално се врши со избор на алатката за превртување Flip () (од комплетот алатки) или со избор од менито Image -> Transform -> Flip Horizontally / Flip Vertically.



Слика 4.4: Превртување

4.3.8. РОТИРАЊЕ

Ротацијата на слика е овозможена преку избор на алатката за ротирање од комплетот со алатки () или со избор од менито Image -> Transform -> Rotate 90° clockwise (ротирање за 90 степени во насока на стрелките од часовникот) / Rotate 90° counter-clockwise (ротирање за 90 степени спротивно од насоката на движење на стрелките од часовникот) и Rotate 180° (ротирање за 180 степени).



Слика 4.5: Ротација

4.3.9. ЦРТАЊЕ НА КРИВИ И ПРАВИ ЛИНИИ

Следните три алатки за цртање закривени и прави линии се користат на ист начин:



- молив за цртање груби линии,



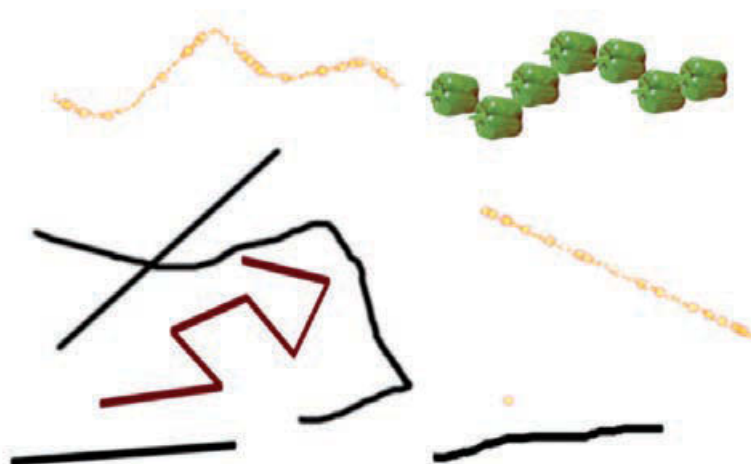
- четка за цртање меки линии,



- спреј.



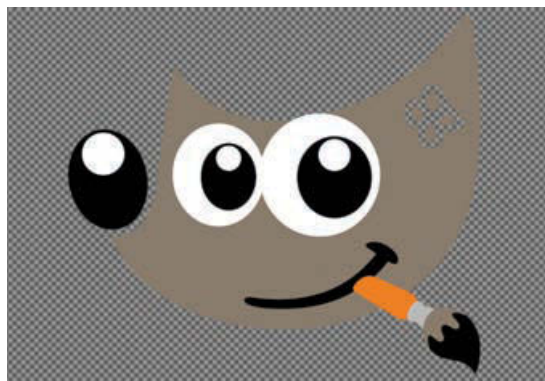
Најпрво, во прозорецот со својства на алатката се поставуваат опциите како што се бојата (Foreground Color), обликот (Brush) и големината (Size) на четката, а потоа се притиска левото копче на глумчето на платното и се движи без да се отпушти копчето, сè додека не се нацрта саканата форма. Правите линии се цртаат со поставување на крајните точки на линиите, со држење на копчето Shift при кликување на платното (првата точка се поставува без држење на копчето Shift). Секој пат кога ќе се кликне на овој начин, програмата ќе ја поврзе новата со претходната точка со права линија.



Слика 4.6: Цртање линии

4.3.10. БРИШЕЊЕ НА ДЕЛОВИ ОД СЛИКАТА

Алатката за бришење () се користи на ист начин како и алатките за цртање и има слични својства, како што се облик, големина на четката и непроѕирност. За разлика од горенаведените алатки за цртање што користат примарната боја (Foreground Color), гумата за бришење ја користи заднинската боја. Ако слојот на сликата има транспарентна заднина, областа на сликата избришана со гумата за бришење ќе биде проѕирна. Ако слојот на сликата нема транспарентна заднина, областа избришана со гумата за бришење ќе биде обоена со бојата на заднината.

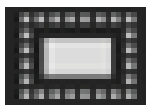


Слика 4.7: Бришење на слика со транспарентна заднина

4.3.11. АЛАТКИ ЗА СЕЛЕКТИРАЊЕ

Наредбите за селекција се групирани во рамките на менито Select. Основни наредби за работа со селекција се: All – селектирање на целата слика, None – поништување на селекцијата, Invert – инвертирање на селекцијата и Float – претворање на селекцијата во привремен слој.

Има неколку алатки за селектирање. Наједноставниот начин на селекција е со едноставно исцртување на делот што треба да се издвои. За таа цел може да се користат алатките од комплетот алатки и тоа:



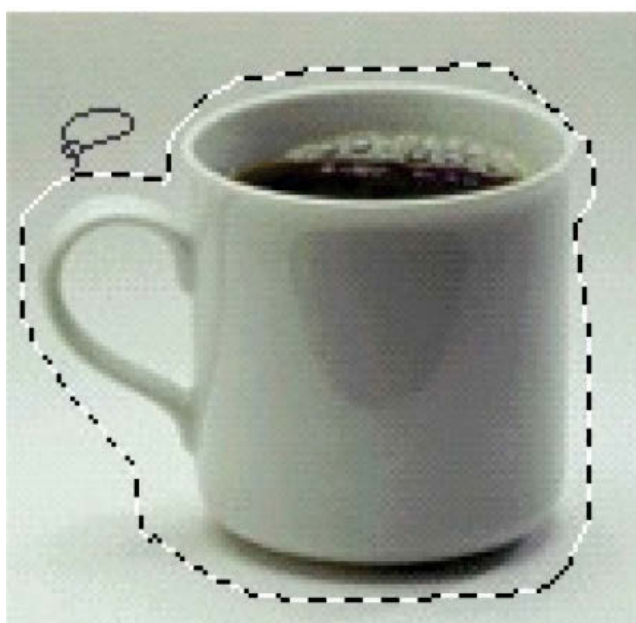
- алатка за избор на правоаголна површина



- алатка за елипса



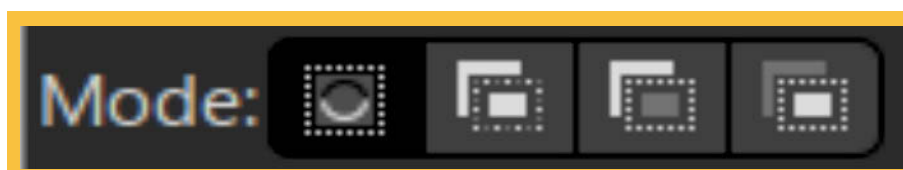
- алатка за слободен избор



Слика 4.8: Селекција со алатка за слободен избор

Селекцијата се потврдува со притискање на Enter или со кликување на друга алатка за уредување слики. Држењето на копчето Shift додека се селектира го претвора правоаголникот во квадрат, а елипсата во круг. Сè додека не се потврди селекцијата, нејзината големина може да се менува.

Со избор на алатките за селектирање се овозможува и опцијата Mode, која го одредува односот на новата и постоечката селекција.



1

2

3

4

Слика 4.9: Опција Mode

Има четири можности:

1. Replace the current selection – постоечката селекција се поништува и се прави нова селекција.
2. Add to the current selection – постоечката селекција се спојува со новата (унија на две селекции).
3. Subtract from the current selection – од постоечката селекција се бришат деловите што се заеднички со новата селекција (разлика – од постоечката се одзема областа на новата селекција).
4. Intersect with the current selection – селекцијата ги опфаќа заедничките делови од постоечката и од новата селекција (пресек на постоечката и новата селекција).

Методот на комбинирање на новата селекција со постоечката треба да се избере пред цртањето на новата селекција.



Слика 4.10: Пресек, разлика и унија на две селекции

Во ГИМП е овозможена и селекција по боја. Алатките Fuzzy Select () и Select by Color () овозможуваат да се избере област што многу наликува на бојата на точката на која е кликнато. Разликата помеѓу алатките Fuzzy Select и Select by Color е во тоа што првата ќе ја избере само областа околу почетната точка, а втората ќе ги избере сите области на сликата (не само околу почетната точка, туку каде било во сликата) што доволно личат на бојата на почетната точка.

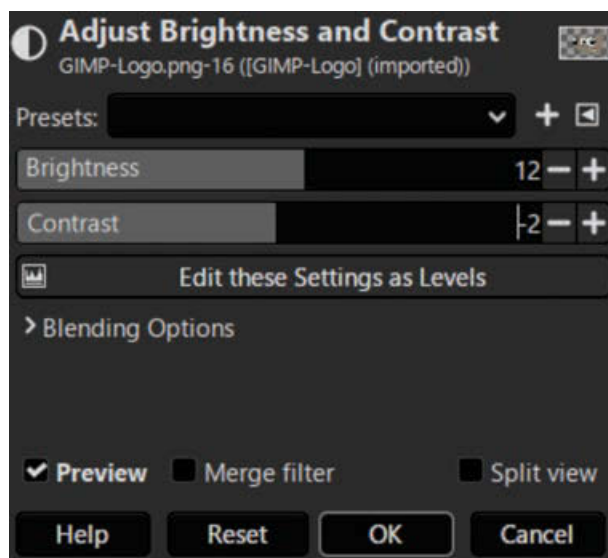
Селекција може да се изврши и со алатката ножички (). Алатката Ножички се користи за селектирање делови од слика што имаат неправилна форма. Корисникот поставува точки околу објектот, а програмата автоматски го следи неговиот раб, според разликите во бојата и контрастот. По затворање на селекцијата, избраниот дел може да се копира, да се премести, да се избрише или дополнително да се уредува.

4.3.12. ОСВЕТЛУВАЊЕ НА СЛИКАТА

Се користи кога е сликата премногу темна или кога одредени детали не се доволно видливи. Во GIMP оваа промена може да се направи преку менито Colors - Brightness-Contrast. По изборот на оваа опција се отвора прозорец во кој вредноста за Brightness се зголемува или се намалува со помош на лизгач. Со зголемување на вредноста сликата станува посветла, а со намалување станува потемна. При уредувањето треба да се внимава осветлувањето да не биде премногу силно, затоа што може да се изгубат детали и сликата да изгледа неприродно.

4.3.13. КОНТРАСТ НА СЛИКАТА

Се користи за нагласување на разликата помеѓу светлите и темните делови на сликата. Во истиот прозорец, преку лизгачот Contrast, вредноста на контрастот може да се зголеми или да се намали. Со зголемување на контрастот, светлите делови стануваат посветли, а темните потемни, па сликата изгледа поизразена. Со намалување на контрастот, разликите помеѓу светлите и темните делови се намалуваат, поради што сликата може да изгледа побледа или помека. По постигнување на соодветниот изглед, промената се потврдува со копчето OK.



Слика 4.11: Дотерување на осветлување и контраст

При операциите поврзани со селекцијата се користи и алатката за преместување.

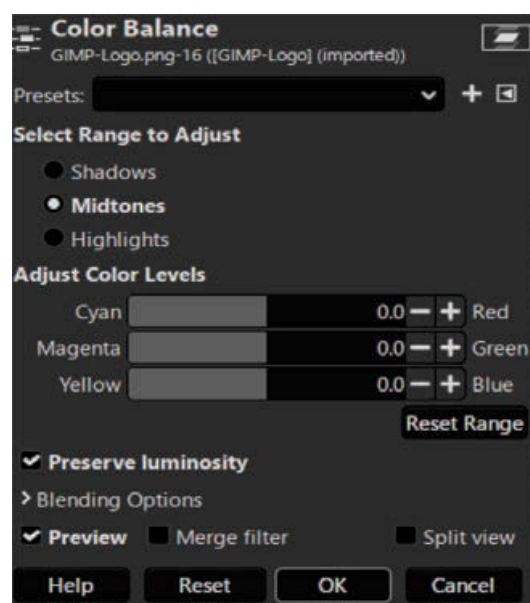
Може да се менува и дебелината на граничната линија на селектираната површина со избор на опциите Избери – Граница, по што се појавува прозорецот за дотерување.

4.3.14. БОЈАТА НА СЛИКАТА

Командата Colors > Color Balance овозможува корекција на сликите каде што боите не се избалансирани.

За да се приспособат темните бои, се избира Select Range to Adjust-> Shadows. Midtones ги означува средните тонови, а Highlights светлите бои. Со избор на копчето Reset (сите поставки се враќаат на нивните стандардни вредности).

Боите на сликата може да се менуваат и со наредбата Colors > Hue-Saturation.



Слика 4.12: Прозорец Color Balance



Слика 4.13: Практична примена на наредбата Hue-Saturation

4.3.15. КЛОНИРАЊЕ НА СЛИКАТА

Се користи за копирање делови од една област на сликата и нивно пренесување на друга област.

Во GIMP ова се прави со алатката Clone Tool (), која се избира од панелот со алатки. Најпрво се избира изворниот дел од сликата, односно областа од која ќе се копираат пикселите. Тоа се прави со притискање на копчето Ctrl и кликување врз избраната област. Се поставува покажувачот на глумчето на местото каде што се сака да се копира, се притиска левото копче на глумчето и со повлекување на глумчето, без да се отпушта копчето, избраната содржина се пресликува на новото место. Потоа, врз делот што треба да се уреди, избраната содржина се пресликува на новото место. При клонирањето треба да се внимава изворниот дел да одговара според бојата, текстурата и осветлувањето, за измената да изгледа природно. Оваа алатка најчесто се користи за отстранување мали неправилности, покривање несакани делови или повторување одредени елементи на сликата.



Слика 4.14: Пример за клонирање на слика

4.3.16. АЛАТКА ЗА ПОПРАВКА НА СЛИКИ

Алатката Healing () се користи за поправка на мали недостатоци на сликата, како дамки, точки или гребнатини.

- Се кликнува на иконата на алатката Healing;
- Се означува местото од каде што се зема примерок, се држи притиснато копчето Ctrl и се кликнува на тоа место. Откако ќе се означи изворот, треба да се отпушти Ctrl. GIMP ќе нацрта круг околу избраната точка;
- Се поставува покажувачот на местото што треба да се коригира. Ако е областа доволно мала за да се вклопи во четката, честопати е доволен еден клик. Ако е областа што треба да се коригира поголема од четката, се притиска левото копче на глумчето и се движи глумчето без да се отпушти копчето, сè додека не се покрие целата област.

Ако е потребно, за да се добијат најдобри можни резултати, истата област може да се поправи неколку пати, сè додека поправеното место не се вклопи доволно добро во околината.



Слика 4.15: Пример за поправка на слика

4.3.17. РАБОТА СО ТЕКСТ

Во GIMP, покрај обработка на слики, може да се додава и да се уредува текст. За таа цел се користи алатката Text Tool (), која се избира од панелот со алатки. По изборот на алатката, се кликнува на местото во сликата каде што треба да се внесе текстот, а потоа се пишува саканата содржина. Текстот може да се уредува преку опциите за избор на фонт, големина, боја, порамнување и растојание меѓу буквите и редовите. При додавање текст, GIMP автоматски создава нов текстуален слој, што овозможува текстот да се уредува одделно од останатите делови на сликата.



Слика 4.16: Лента за форматирање текст



ЗАКЛУЧОК:

Работната околина на ГИМП е составена од три панела: панел со алатки, панел со слика и помошен панел. Сликите изработени во ГИМП се зачувуваат во форматот XCF, со наставка .xcf. За зачувување на сликите во друг формат се користи наредбата File>Export as. Image → Scale Image се користи за промена на големината, Image → Canvas Size за приспособување на платното и Crop Tool за отсекување дел од сликата. Ротација и превртување преку Image → Transform → Rotate / Flip, селекција на објект со алатките Rectangle Select, Ellipse Select и Free Select, исто и бришење или преместување делови. Алатки како Eraser Tool, Clone Tool и Healing Tool се користат за отстранување или за коригирање на делови од сликата. Приспособување на изгледот на сликата се врши со наредбите Colors → Brightness-Contrast, Color Balance и Hue-Saturation. Додавање текст на слика се изведува со алатката Text Tool, со што се креираат уредени и креативни графички решенија.





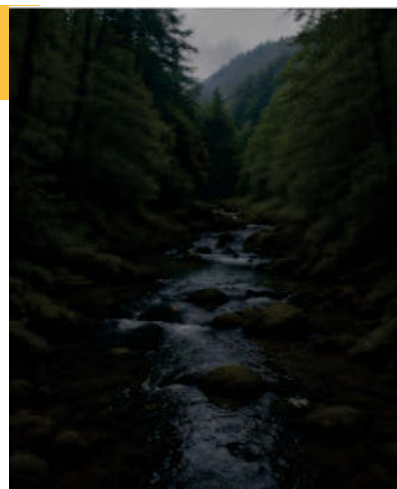
ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Спореди ја промената на големината на сликата со промената на големина-та на платното. Во кои ситуации би ја користел/а едната, а во кои другата постапка?
2. Анализирај ја разликата помеѓу наредбите Save As и Export As во GIMP. Зошто е важно проектот да се зачува во формат XCF, а конечната слика да се извезе во PNG или во JPG?
3. Објасни што ќе се случи ако се претвори сликата од RGB во Grayscale модел на бои. Кои информации од сликата се губат?
4. Спореди ги алатките Fuzzy Select и Select by Color. Во кој случај е подобро да се користи едната, а во кој другата алатка?
5. Анализирај ја разликата помеѓу алатката Clone Tool и алатката Healing. Зошто Healing може да даде поприроден резултат при отстранување мали недостатоци?
6. Објасни како режимите за селекција Add, Subtract и Intersect влијаат врз постоечката селекција. Наведи пример кога секој режим би бил корисен.
7. Спореди го отсекувањето на дел од слика со селектирање и бришење делови од слика. Каква е разликата во резултатот?
8. Анализирај како промената на осветлувањето и на контрастот може да го подобри или да го влоши квалитетот на сликата.
9. Процени која алатка би била најсоодветна за отстранување мала дамка од фотографија: Clone Tool, Healing или Eraser. Објасни го изборот.
10. Процени кога е оправдано да се намали контрастот на сликата. Дали секогаш зголемувањето на контрастот ја подобрува сликата?
11. Процени дали алатката Eraser секогаш брише дел од сликата на ист начин. Како влијае тоа дали слојот има транспарентна заднина или не?
12. Избери најсоодветна алатка за селектирање објект со неправилна форма и објасни зошто токму таа алатка е најпогодна.
13. Вреднувај ја важноста на работната околина во GIMP. Како панелите, менијата, линијарите и статусната лента му помагаат на корисникот при обработка на слика?
14. Осмисли постапка за уредување фотографија која е темна, малку искривена и има непотребен простор околу главниот објект. Кои алатки и наредби би ги користел/а и по кој редослед?
15. Изработи ги и следниве практични вежи.

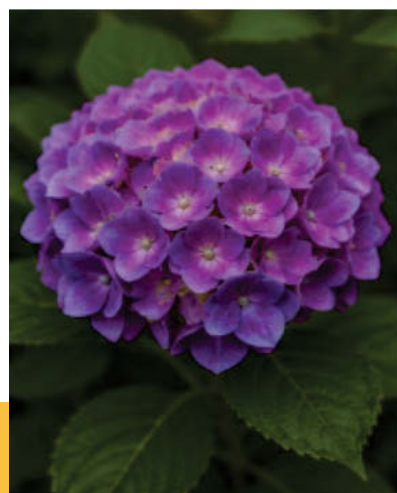


Отсечи ја сликата така што главниот објект ќе остане во фокусот и потоа промени ја нејзината големина.

Изврши корекција на темна и бледа слика.



Клонирај делови од околната трева и пополни ја празната површина, така што сликата ќе изгледа уредно и природно.



Користејќи алатка за селекција, промени ја бојата на цвеќето.



Креирај ја зададената слика .

4.4

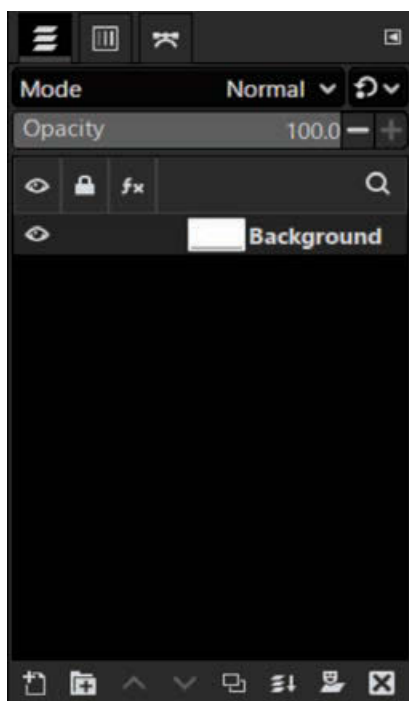


СЛОЕВИ (LAYERS) ВО ГРАФИКА

При обработка на слика во ГИМП често се користат повеќе елементи, како фотографија, текст, форма, цртеж или друга слика. За поедноставна работа, овие елементи може да се постават на посебни слоеви. Словите овозможуваат секој дел од сликата да се уредува одделно, без директно да се менуваат останатите делови.

Слојот може да се замисли како прозирна површина поставена врз друга површина. Ако на еден слој има фотографија, на друг текст, а на трет форма, сите заедно ја создаваат конечната слика. Сепак, секој слој може посебно да се премести, да се избрише, скрие, преуреди или да се измени. Токму затоа се важни слоевите за организирана и прецизна обработка на сликата.

Во GIMP слоевите **се прикажуваат** во панелот Layers. Ако не е видлив панелот, може да се отвори преку наредбата Windows → Dockable Dialogs → Layers. Во овој панел се прикажуваат сите слоеви што се користат во сликата, нивниот редослед, имињата и нивната видливост. Активниот слој е оној што е избран во панелот и врз него се применуваат алатките и наредбите.



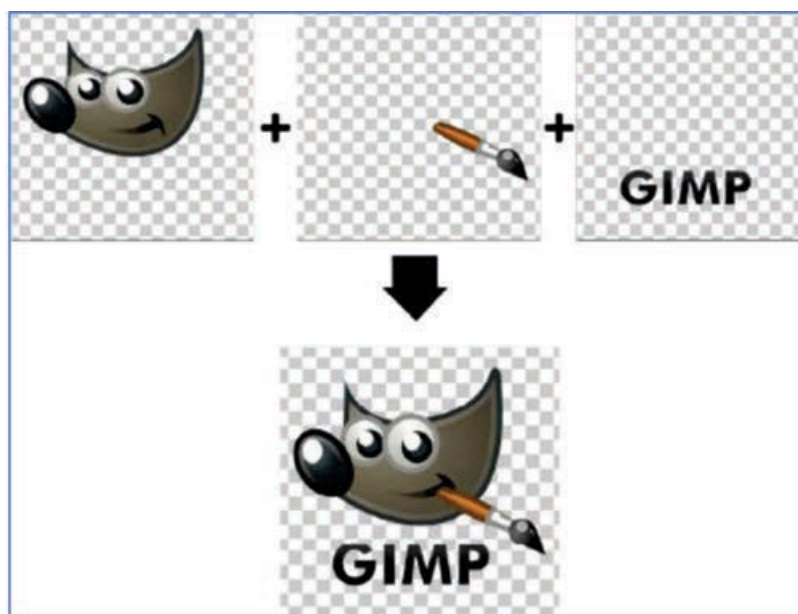
1. Креирање нов слој
2. Креирање групиран слој
3. Менување на редоследот на редување на слојот нагоре по една позиција
4. Менување на редоследот на редување на слојот надолу по една позиција
5. Дуплирање на активниот слој
6. Спојување на активен слој со оној под него
7. Креирање маска на слојот
8. Бришење на слојот

Слика 4.17: Панел за слоеви

Нов слој се додава со наредбата Layer → New Layer или со копчето за нов слој во панелот Layers. При создавање нов слој може да се зададе неговото име, големината и типот на пополнување, на пример, прозирна заднина, бела заднина или определена боја на заднината. Именувањето на слоевите е корисно, особено кога сликата содржи повеќе елементи.

Редоследот на слоевите влијае врз тоа кои елементи ќе се гледаат пред другите. Слојот што се наоѓа погоре во панелот Layers се прикажува пред слоевите што се под него. Редоследот може да се промени со повлекување на слојот нагоре или надолу во панелот. Исто така, може да се користат наредбите Layer → Stack → Raise Layer за преместување нагоре на слојот и Layer → Stack → Lower Layer за преместување надолу на слојот.

По завршување на обработката на сликите, најдобро е да се соединат слоевите. Со наредбата Layer → Merge Down се соединуваат соседни слоеви. На пример, слој со текст над слој со слика, со оваа наредба текстот ќе се соедини со сликата во еден слој. Исто така, може да се соединат сите слоеви што се видливи во моментот (имаат вклучено „око“). Тоа се изведува со наредбата Image → Merge Visible Layers. Корисно е да се примени кога проектот содржи повеќе слоеви, но има потреба сите видливи елементи да се наоѓаат во еден слој. Со наредбата Image → Flatten Image, се соединуваат сите слоеви во еден единствен слој и се отстранува транспарентноста. Најчесто се користи пред зачувување во формати кои не поддржуваат слоеви (на пример JPG).



Слика 4.18. Слика креирана со примена на слоеви

Видливоста на слојот се управува преку иконата со око што се наоѓа покрај името на слојот во панелот Layers. Кога е прикажано окото, слојот е видлив. Кога ќе се исклучи, содржината на тој слој привремено не се гледа, но не е избришана. Ова е корисно кога треба да се споредат различни изгледи на сликата или кога одреден елемент привремено не треба да се прикажува.

Слојот може да се **избрише** со избирање на слојот во панелот Layers и користење на наредбата Layer → Delete Layer или со копчето за бришење во панелот. Ако е потребно да се направи копија од постоечки слој, се користи наредбата Layer → Duplicate Layer. Ова е корисно кога сакаме да се обидеме да направиме промена, а притоа да ја зачуваме оригиналната верзија на слојот.



Слика 4.19: Слика креирана од седум слоја

При работа со повеќе слоеви, секој елемент може посебно да се уредува. На пример, може да се постави фотографија како заднина, врз неа да се додаде текст со алатката Text Tool, а потоа да се додаде форма или друга слика на нов слој. Бидејќи секој елемент е на посебен слој, може едноставно да му се променат положбата, големината, бојата, видливоста или редоследот.

Во ГИМП може да се менува и **непроѕирноста** на слојот преку опцијата Opacity во панелот Layers. Со намалување на вредноста на непроѕирноста, слојот станува попрозирен и низ него може делумно да се гледаат слоевите под него. Оваа можност се користи за намалување на острите граници помеѓу елементите и постигнување постепено прелевање на боите меѓу елементите, порамномерно соединување на елементите или комбинирање повеќе елементи во една композиција.

Откако ќе се заврши **уредувањето**, сликата може да се зачува како работен проект во формат XCF, преку File → Save As, со што се зачувуваат сите слоеви. Ако треба сликата да се користи како обична слика, на пример во документ, презентација или веб-страница, таа се извезува преку File → Export As во формат како PNG или JPG. При извозот слоевите се спојуваат во една конечна слика.



ЗАКЛУЧОК:

За поедноставна обработка на слики се користат слоеви како начин за организирање на одделни елементи во една слика. На секој слој може да се работи независно, така што сликите на другите слоеви не се менуваат. Работата со слоеви се одвива преку панелот Layers во кој активниот слој е означен со сина боја. По обработката на сликата, сите слоеви се соединуваат со наредбата Image>Flatten Image. Еден слој може да се трансформира со следните алатки: наредби за додавање (Layer → New Layer), бришење (Layer → Delete Layer), дуплирање (Layer → Duplicate Layer), промена на редоследот (Layer → Stack → Raise/Lower Layer) и менување на непроѕирноста (Opacity во панелот Layers).

Учениците ќе умеат да ги менуваат видливоста и редоследот на слоевите, со што ќе контролираат кој елемент се прикажува над или под друг. Преку комбинирање на повеќе слоеви, ќе создаваат посложени и покреативни графички решенија.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што се слоеви во GIMP?
2. Зошто се користат слоеви при обработка на слика?
3. Каде се прикажуваат слоевите во GIMP?
4. Како се отвора панелот Layers ако не е видлив?
5. Што претставува активен слој?
6. Како се додава нов слој во GIMP?
7. Зошто е корисно слоевите да се именуваат?
8. Како влијае редоследот на слоевите врз изгледот на сликата?
9. Како може да се промени редоследот на слоевите?
10. За што служи иконата со око покрај слојот?
11. Која е разликата помеѓу сокривање и бришење слој?
12. Кога може да биде корисно дуплирањето на слој?
13. Зошто е добро текстот да биде поставен на посебен слој?
14. Во кој формат се зачувува работната верзија на сликата во GIMP?
15. Која е разликата помеѓу Save As и Export As?
16. Отвори нова слика во GIMP. Креирај најмалку три слоја: заднина, форма и текст. Заднината пополни ја со боја, на вториот слој нацртај правоаголник или друга форма, а на третиот слој внеси краток текст.
17. Со користење на слоеви креирај постер за безбедно однесување на интернет.





ГЕНЕРАТИВНА ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА ЗА ГРАФИКА

Генеративната вештачка интелигенција претставува технологија која може да создава нови содржини врз основа на дадени упатства од корисникот. Во областа на графиката, таа се користи за создавање слики, илустрации, постери, банери, логоа, заднини и други визуелни елементи. Наместо корисникот целосно рачно да ја црта или да ја уредува сликата, тој може да внесе опис, а врз основа на тој опис ВИ алатката да предложи визуелно решение.

Една од најчестите примени на генеративната вештачка интелигенција во графиката е text-to-image. Овој поим означува создавање слика од текстуален опис. Корисникот внесува реченица или повеќе реченици во кои објаснува што сака да биде прикажано на сликата, а алатката генерира слика според тие насоки. На пример, ако се внесе опис „постер за училиштен саем со книги, светли бои и голем наслов“, ВИ алатката може да создаде слика што одговара на тој опис.

Текстуалното упатство што му се дава на ВИ системот се нарекува prompt. Prompt-от треба да биде јасен, конкретен и доволно описен. Во него може да се наведат објектите што треба да се појават, стилот на сликата, боите, распоредот на елементите, големината, атмосферата и намената на графиката. Колку што е попрецизен prompt-от, толку е поголема можноста добиениот резултат да биде поблиску до замислата на корисникот.

На пример, prompt-от „направи слика со природа“ е премногу општ. Подобар prompt би бил: „креирај хоризонтален банер со планински пејзаж, зелена шума, сино небо и простор за наслов во горниот лев агол“. Во вториот пример алатката добива повеќе информации за содржината, боите, распоредот и намената на сликата.



Слика 4.20. Примена на општи инструкции, извор Leonardo.ai



Слика 4.21. Примена на специфични инструкции, извор Leonardo.ai

Постојат повеќе апликации и онлајн алатки со кои може да се генерираат графички содржини со користење на вештачка интелигенција. Такви примери се Adobe Firefly, Canva Magic Media, Microsoft Designer, Midjourney, ChatGPT (со можност за генерирање слики), Leonardo AI, Ideogram и Freepik AI. Овие алатки овозможуваат создавање слики, илустрации, постери, банери и други визуелни содржини врз основа на текстуален prompt. Некои од нив се повеќе насочени кон уметнички слики, други кон графички дизајн и материјали за социјални мрежи, а трети кон уредување и доработка на постоечки слики.

ВИ алатките за генерирање слики може да се користат како помош при дизајнирање. Тие овозможуваат брзо создавање почетни идеи, визуелни предлози и различни варијанти на ист дизајн. Корисникот може да спореди повеќе резултати, да избере најсоодветен и потоа дополнително да го уреди со програма за графичка обработка. На овој начин ВИ не ја заменува целосно креативната работа, туку може да помогне во планирање, инспирација и побрзо изработување на графички решенија.

Генеративната вештачка интелигенција може да помогне и при претворање или приспособување графики од еден вид во друг. Растерската слика е составена од пиксели и најчесто се користи за фотографии и сложени слики со многу детали. Векторската слика е составена од линии, форми и математички дефинирани елементи, па може да се зголеми без да се загуби квалитетот. Со помош на одредени ВИ алатки или алатки со ВИ поддршка, растерска слика може да се поедностави и да се претвори во векторски изглед, на пример како лого, икона или илустрација. Од друга страна, векторската графика може да се изведе како растерска слика кога треба да се користи во документ, презентација или веб-страница.

При користење ВИ алатки важно е да се анализираат резултатите. Генерираната слика не мора секогаш да биде точна, соодветна или квалитетна. Понекогаш може да има неправилности во формите, текстот, пропорциите, боите или во распоредот на елементите. Затоа корисникот треба внимателно да провери дали сликата одговара на зададената цел, дали е јасна, дали е визуелно уредна и дали може да се користи за предвидената намена.

Генерираните графики може да се споредат со рачно креирани графики. Рачно креираната графика овозможува поголема контрола врз секој елемент, бидејќи корисникот сам ги поставува фор-

мите, боите, текстот и распоредот. ВИ графиката, пак, овозможува побрзо добивање идеи и визуелни предлози, но понекогаш бара дополнителна проверка и уредување. Најдобри резултати често се добиваат кога ВИ се користи како помошна алатка, а корисникот го доуредува и го оформува конечниот дизајн.

Во дизајнот, ВИ алатките може да се применат при создавање постери, банери, насловни слики, реклами, илустрации, заднини и визуелни материјали за социјални мрежи. На пример, ученикот може да генерира заднина со ВИ алатка, потоа да ја отвори во програма за обработка на слика, да додаде текст, да промени бои, да отстрани делови од сликата и да го подготви конечниот постер. Така се комбинираат изучените техники за графичка обработка со можностите на вештачката интелигенција.

При работа со ВИ алатки треба да се почитуваат и правилата за одговорно користење. Не треба да се генерираат содржини што навредуваат, измамуваат или погрешно претставуваат личности и настани. Исто така, треба да се внимава дали сликата смее да се користи за јавна објава, за училиштен проект или за комерцијална намена. Корисникот секогаш треба критички да го разгледа добиениот резултат и – ако е потребно – да наведе дека при изработката е користена ВИ алатка.

Генеративната вештачка интелигенција претставува корисна поддршка во современата графичка обработка. Таа овозможува брзо создавање идеи и визуелни решенија, но успешниот дизајн сè уште зависи од знаењето, од процената и од креативноста на корисникот. Затоа е важно корисникот да знае да формулира јасен prompt, да ги анализира добиените резултати и да ги комбинира ВИ алатките со рачните техники за обработка на слика.



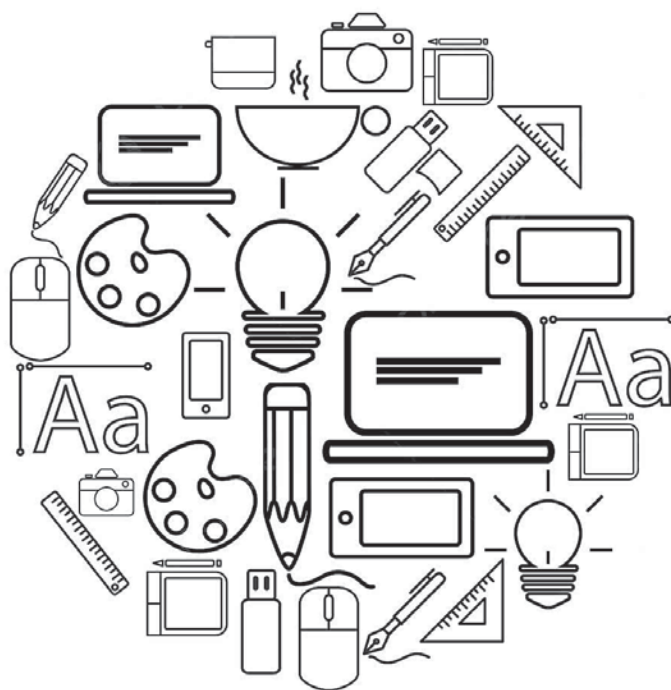
ЗАКЛУЧОК:

Користењето ВИ алатка за создавање дигитални слики е поефикасно ако се формулираат едноставни и јасни инструкции (prompt-ови) кои директно влијаат врз квалитетот на добиените резултати. Преку анализа на генерираните слики се развива критичко размислување и способност за проценка на точноста, креативноста и релевантноста на содржините создадени од ВИ. Споредбата меѓу слики креирани со вештачка интелигенција и рачно изработени графички решенија посочува на предностите, но и на ограничувањата на двата пристапи. ВИ може да биде корисна алатка која го надополнува, но не го заменува човечкото креативно изразување.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што претставува генеративна вештачка интелигенција?
2. Како се користи генеративната вештачка интелигенција во графиката?
3. Што значи поимот text-to-image?
4. Што е prompt?
5. Кои податоци може да се наведат во prompt за генерирање слика?
6. Наведи пример за ВИ алатка со која може да се генерира слика.
7. Зошто треба да се анализира добиената слика од ВИ алатка пред да се употреби?
8. Спореди слика генерирана со ВИ и рачно креирана графика. Кои се предностите и недостатоците на двата начина?
9. Подобри го следниот prompt: „Направи постер за училиште.“ Додај детали за тема, бои, распоред и текст.
10. Осмисли prompt за банер на тема „Дигитална безбедност“.
11. Со помош на ВИ алатка генерирај почетна слика за постер на тема „Дигитална безбедност“. Најпрво формулирај prompt во кој ќе наведеш што треба да содржи сликата, какви бои да се користат и каде треба да има простор за наслов. Потоа анализирај го добиениот резултат и одлучи дали е сликата соодветна за постер. Отвори ја сликата во програма за обработка на слика, додај наслов, краток текст и по потреба направи корекции на големината, бојата или на распоредот. Конечниот постер зачувај го како работен проект и извези го како слика.



ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ТЕМАТА

Заокружи го точниот одговор.

1. Што претставува мултимедија?

- А. Уредување фотографии во графичка програма
- Б. Комбинирање повеќе елементи за пренесување содржина
- В. Зачувување слики во различни формати
- Г. Претворање текст во табела

2. Што е растерска слика?

- А. Слика составена од линии и геометриски форми
- Б. Слика составена од пиксели
- В. Слика што не може да се печати
- Г. Слика што секогаш има прозирна заднина

3. Што е векторска слика?

- А. Слика составена од пиксели
- Б. Слика составена од звук и видео
- В. Слика составена од линии, криви и форми
- Г. Слика што може да се отвори само во GIMP

4. Кој формат најчесто се користи за фотографии?

- А. JPG Б. SVG В. AI Г. DXF

5. Кој формат е најсоодветен за лого што треба да се користи во различни големини?

- А. JPG Б. BMP В. SVG Г. RAW

6. Во кој формат GIMP ја зачувува работната верзија на сликата со слоеви?

- А. JPG Б. PNG В. XCF Г. GIF

7. Што значи text-to-image?

- А. Претворање слика во текст
- Б. Генерирање слика од текстуален опис
- В. Бришење текст од слика
- Г. Зачувување текст како слика

8. Една слика треба да се користи како мала икона на веб-страница, но и како голем знак на постер. Кој тип графика е најсоодветен?

- А. Растерска графика, затоа што е составена од пиксели
- Б. Векторска графика, затоа што може да се зголеми без губење на јасноста
- В. JPG фотографија, затоа што секогаш има мала големина
- Г. GIF датотека, затоа што секогаш е најквалитетна

9. Ако растерска слика со мала резолуција се зголеми многу, што најверојатно ќе се случи?

- А. Ќе стане поостра
- Б. Ќе се претвори во векторска слика
- В. Ќе изгледа заматено или составена од мали квадратчиња
- Г. Ќе добие просирна заднина

10. Во GIMP ученик го уредува текстот, но промените се применуваат на погрешен елемент. Која е најверојатната причина?

- А. Избран е погрешен активен слој
- Б. Сликата е премногу мала
- В. Фонтот е премногу голем
- Г. Документот е зачуван како XCF

11. Ако слојот со текст се постави под слојот со темна заднина, што може да се случи?

- А. Текстот може да не биде видлив
- Б. Текстот автоматски ќе стане поголем
- В. Заднината ќе се избрише
- Г. Сликата ќе се извезе како PNG

12. Која е разликата помеѓу сокривање слој и бришење слој?

- А. Нема разлика
- Б. Сокривањето е привремено, а бришењето трајно го отстранува слојот
- В. Бришењето само ја намалува непроѕирноста
- Г. Сокривањето ја менува големината на сликата

13. Ако е сликата премногу темна, која наредба е најсоодветна за почетна корекција?

- A. Colors → Brightness-Contrast
- Б. File → Export As
- В. Layer → Delete Layer
- Г. Select → None

14. Зошто алатката Healing често дава поприроден резултат од Clone Tool при отстранување мала дамка?

- A. Затоа што автоматски ја брише целата слика
- Б. Затоа што ја приспособува поправката според околината
- В. Затоа што секогаш ја претвора сликата во векторска
- Г. Затоа што додава текст на сликата

15. Кога е подобро да се користи Crop Tool наместо Canvas Size?

- A. Кога треба да се отстрани непотребен дел околу главниот објект
- Б. Кога треба да се додаде нов простор околу сликата
- В. Кога треба да се промени бојата на текстот
- Г. Кога треба да се создаде нов слој

16. Кој prompt е најдобро формулиран?

- A. Направи нешто убаво.
- Б. Слика за училиште.
- В. Креирај хоризонтален банер за училиштен саем на книги, со светла заднина, книги, ученици и простор за наслов.
- Г. Направи боја.

17. Кога е најоправдано да се зачува проектот како XCF?

- A. Кога сликата треба подоцна повторно да се уредува по слоеви
- Б. Кога сликата треба веднаш да се испечати како фотографија
- В. Кога сликата не содржи никакви елементи
- Г. Кога треба да се избрише текстот

18. Треба да се изработи постер со ВИ генерирана заднина, наслов и дополнителна слика. Кој редослед на работа е најсоодветен?

- А. Прво да се извезе постерот, па потоа да се додаде текст
- Б. Да се генерира заднина, да се провери резултатот, да се уреди во графичка програма и да се додаде текст
- В. Да се избрише заднината и да се зачува празна слика
- Г. Да се користи само еден слој за сите елементи

19. Ако треба да се направат две верзии на ист постер, една со лого и една без лого, која можност во GIMP најмногу помага?

- А. Исклучување и вклучување на видливоста на слојот со лого
- Б. Промена на резолуцијата на екранот
- В. Бришење на сите слоеви
- Г. Отворање калкулатор

20. Ученик има фотографија со мала дамка на небото. Која постапка е најсоодветна?

- А. Да ја претвори целата слика во SVG
- Б. Да ја користи алатката Healing или Clone Tool за да ја поправи областа
- В. Да ја намали сликата на 10 пиксели
- Г. Да додаде звук

21. Ако треба да се создаде едноставен постер во GIMP со заднина, форма, слика и текст, како е најдобро да се организираат елементите?

- А. Сите да се постават на еден слој
- Б. Секој главен елемент да биде на посебен слој
- В. Текстот да се избрише по внесувањето
- Г. Сликата да се зачува само како звук

22. Кој план најдобро ја комбинира ВИ алатката со рачна обработка на слика?

- А. ВИ алатката да создаде почетна слика, а корисникот да ја анализира, да ја уреди и да ја при-
способи според задачата
- Б. Корисникот да не проверува ништо и веднаш да ја објави сликата
- В. Да се користи само рачна обработка, без никаква анализа
- Г. Да се користи само еден формат за слики, без оглед на нивната намена.



TEMA 5



ТЕМА 5: ВЕБ-РАЗВОЈ

Основите на веб-развојот се креативен начин да се запознаеш со тоа како функционираат веб-страниците што секојдневно ги користиме. Креирањето веб-страници комбинира креативност и логично размислување. Секоја веб-страница се гради со помош на три основни технологии: HTML, кој ја поставува основната структура, CSS, кој го уредува изгледот и боите, и JavaScript, кој додава интеракција и динамика на страницата.

Кога ќе ги научиш овие основи, ќе можеш да создаваш сопствени веб-страници и да ги претвораеш своите идеи во вистински дигитални проекти. Исто така, важно е да разбереш како да направиш страницата да изгледа добро и на телефон, таблет или на компјутер, како и да биде лесна за користење.

Ова е првиот чекор за секој што сака да започне со веб-развој и можеби во иднина да се занимава со оваа професија.

НОВИ ПОИМИ

- структура на веб-страница, HTML документ, основни HTML ознаки – heading, paragraph, image, link, CSS, селектори, бои, фонтови, заднина, основен распоред на елементите на веб-страницата (layout), основи на адаптивен (responsive) дизајн, приспособување на различни уреди, ВИ инструкции (prompt).

Веќе знаеш:

- Какви можности нудат различните интернет-сервиси.
- Која е улогата на WWW сервисот на интернет.
- Да наведеш примери за Web 1.0, Web 2.0, Web 3.0, Web 4.0, Web 5.0.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- ја објаснуваш основната структура на веб-страница и да применуваш HTML и CSS за креирање и уредување веб-содржини;
- ги применуваш основните принципи на распоред и адаптивен (responsive) дизајн при приспособување веб-страници за различни уреди;
- користиш алатки со вештачка интелигенција за генерирање веб-содржини;
- формулираш инструкции (prompt), да ги анализираш добиените резултати и да уредуваш ВИ-генериран код;
- креираш едноставен веб-проект со комбинирана примена на рачно изработен и автоматски генериран код.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

- Наброј најмалку три различни интернет-сервиси и опиши за што се користат.
- Како влијае WWW-сервисот врз начинот на пристап до информации во споредба со другите интернет-сервиси (на пример е-пошта или FTP).
- Објасни кои се главните разлики помеѓу Web 1.0 и Web 2.0 користејќи сопствени зборови.
- Наведи конкретен пример за Web 3.0 и објасни како ги подобрува тој сервис можностите на интернетот.
- Спореди ги Web 2.0 и Web 4.0 во однос на интеракцијата со корисниците и објасни кои се нивните главни карактеристики.
- Осмисли сопствен пример за Web 5.0 апликација и објасни како би им помогнала таа на корисниците во секојдневниот живот.

5.1



ПОИМ ЗА ВЕБ-СТРАНИЦА

Веб-страница претставува дигитален документ што се прикажува во веб- прелистувач (browser), како што се Google Chrome, Mozilla Firefox или Microsoft Edge. Таа е дел од поширок систем наречен веб-сајт, што претставува збир на повеќе поврзани веб-страници. Секој веб-сајт се состои од една или повеќе веб-страници, меѓусебно поврзани со хиперврски. Поставен е на веб-сервер и има инсталиран софтвер за испорака на веб-страниците.

Секоја веб-страница има своја уникатна адреса, позната како URL (Uniform Resource Locator), преку која корисниците можат да пристапат до неа преку интернет. Веб-страниците најчесто се креирани со користење на јазикот HTML (HyperText Markup Language), (не е програмски јазик) и служи за дефинирање на структурата и содржината на страницата. Тој му дава на прелистувачот податоци за содржината и за структурата на веб-страницата и служи за опишување на елементите и начинот на кој ќе се прикажат тие.

Покрај HTML, често се користат и други технологии како:

- CSS (Cascading Style Sheets) – за стил и дизајн и
- JavaScript – за интерактивност и динамичност.
- Веб-страниците може да се:
- Статички – содржината е фиксна и не се менува често и
- Динамички – содржината се менува во зависност од корисникот или од податоците (на пример, социјални мрежи или веб-продавници).

5.1.1. ОСНОВНА СТРУКТУРА НА ВЕБ-СТРАНИЦА

Секоја веб-страница има логична и техничка структура што овозможува правилно прикажување и функционирање. Основната техничка структура се дефинира со HTML и се состои од неколку главни елементи и се конструира на следниот начин:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Наслов на страницата</title>
</head>
<body>
<h1>Главен наслов</h1>
<p>Ова е пример на текст.</p>
</body>
</html>
```



Овој код го претставува скелетот на една веб-страница. Секој од елементите ги има следните функционалности:

1. `<!DOCTYPE html>`

Овој израз е декларација што му кажува на прелистувачот дека е документот напишан во најновата верзија HTML5 односно дека е веб-страница. Таа е важна за правилно прикажување на страницата.

2. `<html>`

Оваа ознака го означува почетокот на HTML документот. Сè што се наоѓа помеѓу двете `<html>` ознаки е дел од веб-страницата.

3. `<head>`

Во овој дел се внесуваат мета-податоци за страницата, како:

- наслов (`<title>`),
- информации за карактер сет,
- поврзување со CSS датотеки.

Овој дел не се прикажува директно на корисникот.

4. `<meta charset="UTF-8">`

Се користи за прикажување на текстот со кирилично писмо. Многу прелистувачи не се трудат да го препознаат текстот кој е напишан и затоа треба да им се посочи да ја користи Unicode табелата, начин на кодирање UTF-8.

4. `<title>`

Го дефинира насловот на страницата што се прикажува во табот на прелистувачот.

5. `<body>`

Ова е најважниот дел од страницата. Тука се наоѓа сè што корисникот го гледа:

- текст,
- слики,
- видеа и
- линкови.

Покрај техничката структура, веб-страницата има и логичка структура, која овозможува подобра организација на содржината. Оваа структура често вклучува:

1. заглавие (header), односно горниот дел од веб-страницата кој најчесто содржи лого, наслов и навигациско мени;

2. навигација (navigation), мени што им овозможува на корисниците да се движат низ страниците. Се состои од линкови што водат до различни секции или страници;

3. главна содржина (main content), централниот дел, каде што се прикажува основната информација. Овој дел може да содржи текст, слики, видеа и други елементи;

4. странична лента (sidebar), дополнителен дел, кој се наоѓа отстрана и содржи: реклами, дополнителни информации и линкови.

5. подножје (footer), долниот дел од страницата кој најчесто содржи: контакт информации, авторски права и дополнителни линкови.

Добрата структура на веб-страницата има големо значење поради неколку причини:

- Подобра читливост – корисниците поедноставно ја разбираат содржината,
- Подобро корисничко искуство – едноставна навигација и организација,
- Оптимизација за пребарувачи (Search Engine Optimization) – пребарувачите полесно ја анализираат страницата,

- Полесно одржување така што програмерите можат поедноставно да ја уредуваат и подобруваат страницата.

Правилното вгнездување на HTML ознаките е едно од основните правила во веб-развојот. Наједноставно кажано, ознаките функционираат по принципот „со кутии во кутии“ – таа ознака што ќе се отвори последен, треба прв да се затвори (систем на огледало или LIFO - Last In, First Out). Веб-прелистувачите (како Chrome, Edge, Firefox) не го прикажуваат HTML-кодот директно. Тие прво го претвораат во структура наречена DOM (Document Object Model), која изгледа како семејно стебло со родителски и детски елементи (хиерархија).

- Во правилниот пример: `<html>` е родител, а `<body>` е негово дете. Кутијата `<body>` е целосно спакувана внатре во кутијата `<html>`.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Точна структура</title>
  </head>
  <body>
    <h1>Здраво!</h1>
    <p>Ова е <strong>правилно</strong> вгнезден текст.</p>
  </body>
</html>
```



Во неправилен пример (`<html><body>...</html></body>`): Се обидуваме да ја затвориме надворешната кутија (`<html>`), додека внатрешната кутија (`<body>`) сè уште е отворена.

```
<!DOCTYPE html>
<html>
  <body>
  <head>
    <title>Неточна структура</title> </head>
    <h1>Грешка!</h1>
    <p>Ова е <strong>неправилно</p> вгнезден текст.</strong> </body>
</html>
```

Ако е кодот лошо вгнезден, веб-страницата може да изгледа сосема различно на различни прелистувачи или уреди, а и елементите може да се преклопат, да „избегаат“ од своите позиции или текстот да стане нечитлив.



ЗАПОМНИ ШТО НАУЧИ

Веб-страницата претставува основен елемент на интернетот и игра клучна улога во современата комуникација и информирање. Разбирањето на поимот веб-страница и нејзината структура е основа за понатамошно учење во областа на веб-технологиите. Основната техничка структура на веб-страницата се состои од HTML елементи како `<html>`, `<head>` и `<body>`, додека логичката структура вклучува делови како заглавие, навигација, главна содржина и подножје. Овие компоненти заедно овозможуваат веб-страницата да биде функционална, прегледна и корисна за корисниците.



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Која е почетната ознака што му кажува на прелистувачот дека документот е веб-страница напишана во HTML5?
2. Наброј ги трите главни ознаки од кои треба да биде составен секој правилен HTML документ.
3. Каде се сместуваат мета-податоците (како насловот на страницата или кодирањето на знаците) – во `<head>` или во `<body>` секцијата?
4. Објасни ја разликата помеѓу содржината што се наоѓа во `<head>` ознакаот и содржината во `<body>` ознакаот. Што од ова гледа посетителот на веб-страницата?
5. Зошто е важно HTML ознаките да бидат правилно вгнездени (на пример, зошто `<html><body>...</body></html>` е правилно, а `<html><body>...</html></body>` не е)?
6. Со свои зборови опиши ја улогата на ознакаот `<title>`. Каде се прикажува неговата содржина во веб-прелистувачот?
7. Напиши го комплетниот код за празен HTML документ кој ги содржи сите задолжителни структурни елементи, а насловот на страницата во табот на прелистувачот треба да биде „Моја Прва Страница“.
8. Во следниот код има структурна грешка. Каква ќе биде пораката што ќе се прикаже? Пронајди ја и напиши го точниот код.

```
<!DOCTYPE html>
<html>
  <body>
    <head>
      <title>Добредојдовте</title>
    </head>
    <h1>Здраво свет!</h1>
  </body>
</html>
```



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи ја алатката W3C Markup Validation Service. Проучи која е нејзината функционалност.



Markup Validation Service

Check the markup (HTML, XHTML, ...) of Web documents

Validate by URI

Validate by File Upload

Validate by Direct Input

Validate by URI

Validate a document online:

Address:

5.2



ОСНОВИ НА HTML

5.2.1. ОСНОВНИ ОЗНАКИ

За подобро да разбереш како се создава една веб-страница, замисли го HTML кодот како архитектонски план на една кука. Самиот код е составен од ознаки кои му кажуваат на прелистувачот каде да постави прозорец, каде врата, а каде покрив.

Во продолжение ќе ги разгледаме основните HTML ознаки објаснети со практични примери преку кои ќе научиш како да ги применуваш.

1. Ознака за наслови (<h1> до <h6>)

Насловите се користат за организирање на содржината и одредување на нејзината важност. <h1> е најважниот и најголемиот наслов (како наслов на книга), додека <h6> е најмалиот (за под-под-наслови).

- **Практичен пример:** Замисли дека правиш веб-страница за рецепти. Насловот на самиот рецепт ќе биде во <h1>, а поднасловите како „Состојки“ или „Начин на подготовка“ ќе бидат во <h2>.

```
<h1>Рецепт за Палачинки</h1>
```

```
<h2>Потребни состојки</h2>
```

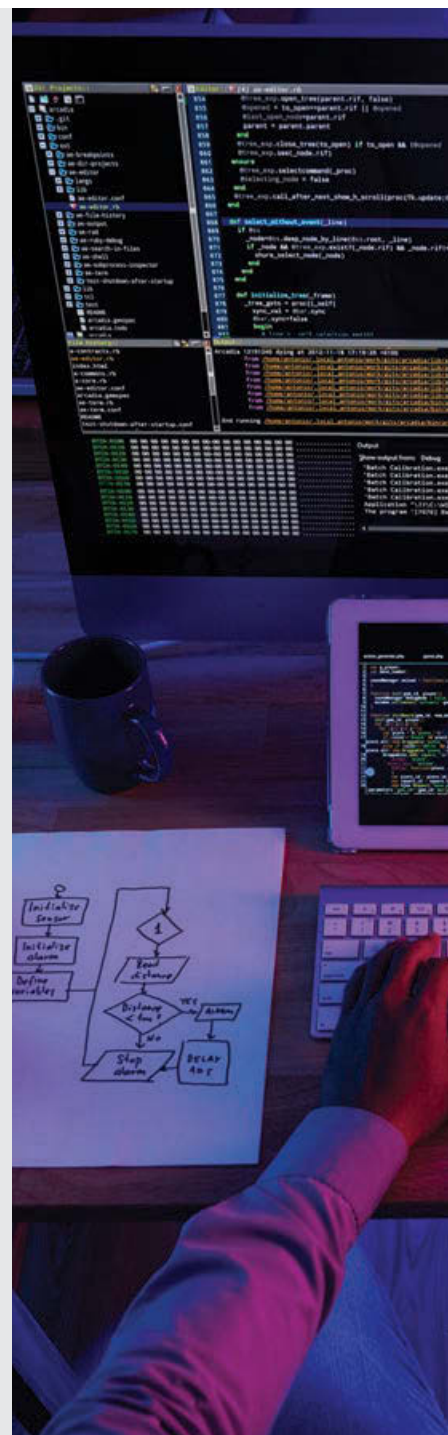
2. Ознаки за текст и наслови

- **параграф (<p>)** — се користи за пишување обичен текст или ставови за да се прикажат како параграф. Секогаш кога отвораш нова <p> ознака, прелистувачот автоматски остава малку празен простор пред и по него за текстот да биде читлив.

На пример, текстот подолу ќе биде прикажан како посебен параграф.

```
<p>Палачинките се омилен десерт на сите генерации. Се подготвуваат брзо и лесно.</p>
```

- * **прекин
** — (break) се користи за присилно префрлување на текстот во нов ред. Оваа ознака е една од ретките што **нема** завршна ознака (се пишува сама).
- * **ознаката ... **, го прави текстот **задебелен (Bold)** и е корисно за истакнување важни зборови.
- * **ознаката ... **, го прави текстот *закосен (Italic)*.



3. Ознака за линкови (<a>)

Ознаката <a> (anchor) служи за поврзување на веб-страница со друга веб-страница на интернет или со друг документ. За да знае прелистувачот каде води линкот, се користи атрибутот href. На пример, кога сакаш да ги упатиш посетителите на Google за да побараат повеќе идеи за фил за палачинки, го пишуваш следниот код:

```
<a href="https://www.google.com">Побарајте идеи на Google</a>
```

4.Ознака за слики ()

Веб-страниците би биле неинтересни без визуелни елементи. Ознаката внесува слика на страницата. Важно е да знаеш дека оваа ознака нема завршна ознака (се затвора сама) и ги користи атрибутите src (патека до сликата) и alt (текст што се појавува ако сликата не се вчита). За прикажување превкусна слика од готовите палачинки се користи следниот код:

```

```

Кога ќе ги примениш сите овие ознаки во една логична целина внатре во главното тело (<body>), го добиваш следниот едноставен, но комплетен HTML код:

```
<!DOCTYPE html>
<html>
<head>
  <title>Мојот прв рецепт</title>
</head>
<body>
  <h1>Рецепт за Брзи Палачинки</h1>
  <p>Ова е највкусниот и наједноставниот рецепт за палачинки кој можете да го направите за само 15 минути.</p>
  
  <p>Доколку ви се допадна овој рецепт, посетете го нашиот партнерски сајт за повеќе идеи:</p>
  <a href="https://www.google.com">Повеќе рецепти тука</a>
</body>
</html>
```

5.Ознаки за креирање листа со точки или бројки:

- — Неозначена листа со точки,
- — Нумерирана листа со бројки 1, 2, 3...,
- — Елемент во листата. Секогаш се става внатре во или .

Може да се употребат за креирање листа на омилени предмети кои ќе се прикажат на твојата веб-страница.

```
<h1>Добредојдовте во моето училиште</h1>
<p>Ова се моите <strong>омилени</strong> предмети:</p>
<ul>
  <li>Математика</li>
  <li>Веб-дизајн</li>
</ul>
```


6. Ознака за дефинирање блок - <div> (скратено од division – оддел, сектор).

Таа е несемантички блок-елемент во HTML и служи како универзален празен контејнер (кутија) за групирање на други елементи со цел нивно полесно организирање, структурирање и стилизирање со помош на CSS или измена со JavaScript. На пример, може да се користи за да се направи картичка за кориснички профил:

```
<div class="profil-karticka">
  
  <h2>Марко Марковски</h2>
  <p>Професија: Веб-дизајнер</p>
</div>
```



ДАЛИ ЗНАЕШ?

Дали знаеш за потребата од поставување „празен простор“ во текстот? Оставањето празни места прелистувачот ги распознава само како едно празно место. Тој ги собира сите празни места во едно. Ако сакаш технички да направиш повеќе празни места едно по друго, треба да користиш специјална ознака за карактер, (non-breaking space).



ИСТРАЖИ

Истражи во кои случаи се јавува потреба за дефинирање празен простор и како се употребува ознаката за повеќе места.



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО:

1. Објасни ја разликата помеѓу логичката структура на една веб-страница (она што го гледа корисникот) и нејзината техничка структура (она што го гледа прелистувачот во кодот).
2. Која е улогата на тагот <head>, а која на тагот <body> внатре во HTML документот? Што се случува ако некој текст за прикажување се стави во <head> наместо во <body>?
3. Имаш задача да креираш онлајн мени за ресторан. Напиши го HTML кодот кој ќе направи нумерирана листа () со три јадења по твој избор.
4. Разгледај го следниот дел од код:

```
<p>Добредојдовте во мојот <strong>блог за програмирање.</p></strong>
```

Пронајди ја грешката во овој код, објасни што е погрешно според правилата за вгнездување (nesting) и напиши го правилниот синтаксички облик.

5. Анализирај како ќе го прикаже прелистувачот текстот напишан во кодот како:

```
<h1>Учиме HTML заедно!</h1>
```

Која карактеристика на HTML ознаките за текст доаѓа до израз во овој пример?

6. Прегледуваш едноставен HTML документ во кој програмерот ја поставил содржината на следниов начин:

На самиот врв, главниот и најважен наслов на статијата го означил со ознака <h3>.

Малку подолу, еден помал поднаслов го означил со ознака <h1>.

На крајот, за еден обичен, долг пасус со текст, тој искористил серија од пет поединечни ознаки <h1> (за секоја реченица посебно), со образложение дека сакал тој текст да биде многу важен и упадлив.

Евалуирај го ваквиот начин на користење на основните HTML ознаки. Прочени какви проблеми во логичката хиерархија создава ваквата структура и како ќе влијае тоа врз пребарувачите (како Google) кои се обидуваат да сфатат која е главната тема на страницата. Предложи што треба да се измени во кодот.

7. Прочени го кодот и определи какви технички и логички грешки се направени:

```
<div>
```

```
<p>Погледнете ги нашите најпознати експонати:</p>
```

```
<ul> <a href="https://muzej.mk/monaliza">
```

```
<li>Мона Лиза </li>
```

```
</a>
```

```
<li>Статуата на Давид</li>
```

```

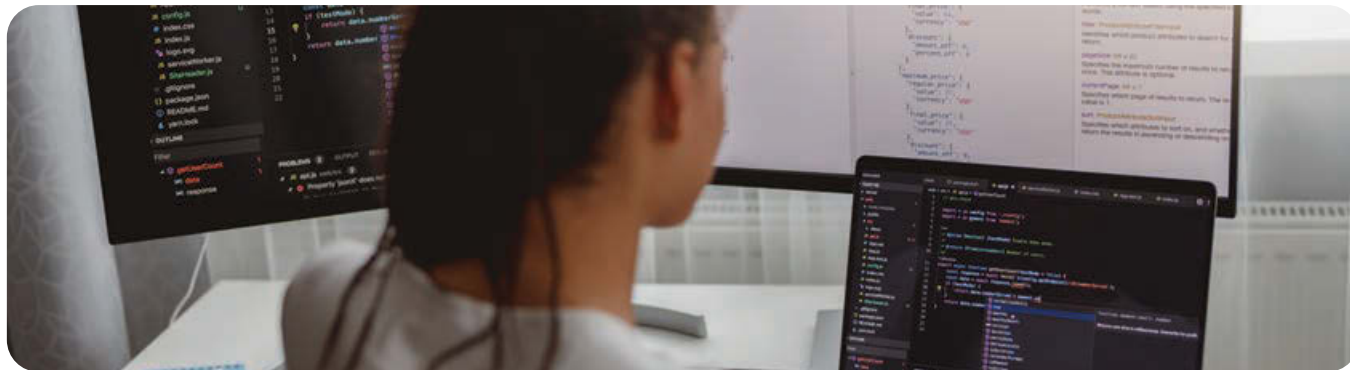
```

```
<li>Свездена ноќ</li>
```

```
</ul>
```

```
</div>
```

Предложи решение за правилно прикажување на листата на веб-страницата на музејот.



ИСТРАЖИ

Како роботите на Google (Web Crawlers) ја скенираат хиерархијата на насловите (<h1> до <h6>)?



ЗАПОМНИ ШТО НАУЧИ

Секој HTML-документ се состои од два дела: заглавие (head) и тело (body). Во заглавието на документот се пишуваат информациите за страницата кои не се прикажуваат како содржина на веб-страницата. Содржината се пишува во телото. Првата ознака `<html>` му кажува на прелистувачот дека тоа е почеток на HTML-документот, а последната `</html>` ознака е дека е тоа крај. Заглавието се оградува со ознаките `<head>` и `</head>` а телото со `<body>` и `</body>`. `<!DOCTYPE>` се користи како декларација за прелистувачот да ја прикаже веб-страницата исправно. Во основниот код потребно е да се стави и ознака со која се дефинира начинот на кодирање, која се става во заглавието `<meta charset="UTF-8">`. Основните ознаки за уредување текст се дадени во следната табела:

Ознака (Tag)	Име / Значење	Главна функција (Улога)
<code><!DOCTYPE html></code>	Декларација на документ	Му соопштува на прелистувачот дека се користи најновата HTML5 верзија.
<code><html></code>	Коренски елемент	Го обвиткува целиот код и ги означува почетокот и крајот на HTML документот.
<code><head></code>	Заглавие на документ	Содржи скриени информации и мета-податоци (како <code><meta charset="UTF-8"></code> и <code><title></code>).
<code><body></code>	Тело на документ	Го содржи целиот видлив дел од страницата (текст, слики, линкови) што ги гледа корисникот.
<code><h1></code> до <code><h6></code>	Наслови	Дефинираат наслови со строга хиерархија (од најважниот <code><h1></code> до најмалиот <code><h6></code>).
<code><p></code>	Пасус (Paragraph)	Служи за организирање и прикажување на обичен блок-текст.
<code><a></code>	Хиперврска (Линк)	Поврзува различни веб-страници преку атрибутот href (дестинација на линкот).
<code></code>	Слика	Вметнува слика на страницата преку изворот src и содржи задолжителен опис alt.
<code></code> / <code></code>	Неозначена / Означена листа	Креираат листи со точки (<code></code>) или листи со бројки (<code></code>), користејќи <code></code> за секоја ставка.
<code><div></code>	Оддел / Кутија (Division)	Универзален празен контејнер што служи за групирање и идно стилизирање на елементите.



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО:

- Која од следниве линии код претставува ПРАВИЛНО вгнездување (nesting) на HTML ознаките?
 - А) `<p>Учиме модерен <strong>HTML код.</p></strong>`
 - Б) `<p>Учиме модерен <strong>HTML код.</strong></p>`
 - В) `<strong><p>Учиме модерен HTML код.</strong></p>`
 - Г) `<strong><p>Учиме модерен HTML код.</p></strong>`
- Што ќе прикаже прелистувачот на екранот ако во VS Code го напишеш следниов текст: `<p>Здраво студенти!</p>` (со 5 празни места)?
 - А) „Здраво студенти!“ (точно 5 празни места, затоа што HTML ги зачувува сите)

- Б) „Здравостуденти!“ (ќе ги спои зборовите комплетно бидејќи празниот простор се брише)
- В) „Здраво студенти!“ (само едно празно место, поради карактеристиката *White-space Collapse*)
- Г) Прелистувачот ќе пријави синтаксичка грешка и воопшто нема да го прикаже текстот.

3. Која е главната улога на тагот `<div>` во HTML?

- А) Служи за автоматско задебелување и зголемување на текстот на екранот.
- Б) Работи како универзален празен контејнер (кутија) за групирање на други елементи.
- В) Се користи исклучиво за внесување слики и линкови на веб-страницата.
- Г) Автоматски ја преведува содржината на страницата на англиски јазик.

4. Кој од следниве тагови спаѓа во групата на „Празни елементи“ (Void Elements) кои немаат завршен таг и се затвораат сами?

- А) `<p>`
- Б) `<h1>`
- В) `<img>`
- Г) `<a>`

5. Зошто е задолжително да ја напишеш линијата `<meta charset="UTF-8">` во `<head>` делот на твојот код, иако VS Code самиот по себе ги зачувува датотеки во тој формат?

- А) За да му кажеш на прелистувачот (Chrome, Safari...) точно кој стандард на карактери да го користи за кирилицата да се прикаже без хиероглифи.
- Б) За да ѝ кажеш на програмата VS Code дека имаш дозвола да пишуваш код на македонски јазик.
- В) За да се активираат автоматските скратеници (Emmet) за побрзо пишување код.
- Г) Со тоа се дефинира главниот наслов кој се појавува на најгорниот таб од прелистувачот.

6. Замисли дека правиш веб-страница и сакаш да поставиш слика од логото на твоето училиште. Напиши ја комплетната HTML линија за вметнување слика, и накратко објасни зошто е важно да го наведеш `alt` атрибутот во таа ознака (наведи барем една причина).



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

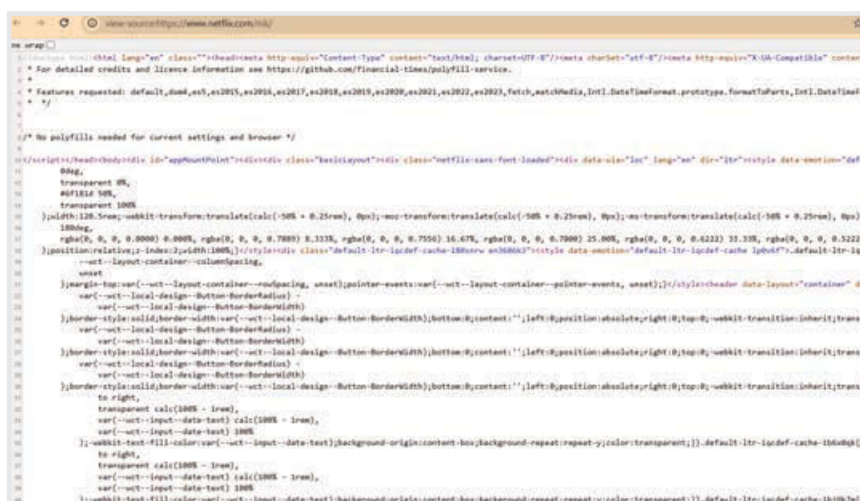
Модерниот HTML не е само за прикажување текст и слики. Тој доаѓа со вградени напредни технологии (API) кои во минатото бараа тешки програми, а денес се активираат со неколку линии код:

- 1. Геолокација (Geolocation):** Преку HTML и малку JavaScript можеш да побараш дозвола од корисникот за сајтот да знае точно каде се наоѓа на мапата.
- 2. Локално складирање (Local Storage):** HTML5 му овозможува на твојот сајт да зачува податоци директно во прелистувачот на корисникот (како „Dark Mode“ прекинувачот или производите во кошничката), па дури и ако корисникот го згасне компјутерот и се врати утре, сајтот ќе го памети неговиот избор.
- 3. Аудио- и видеостриминг:** Таговите `<audio>` и `<video>` имаат напредни контроли кои овозможуваат да направиш сопствен плеер како YouTube или Spotify директно во прелистувачот, без никакви надворешни додатоци.

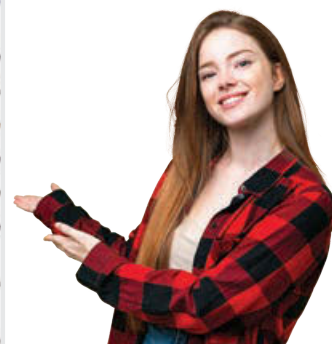


5.2.2 УРЕДУВАЧИ ЗА ПИШУВАЊЕ HTML КОД

Веб-страницата е документ кој содржи HTML код и има наставка .html. Отвори ја твојата омилена веб-страница, кликни со десен клик каде било и од менито одбери View page source. Ќе се прикаже HTML кодот на страницата кој изгледа како текстуален документ кој може да се напише во текст едитор како што е Notepad или Notepad++. Не се препорачува пишување код во текст процесор (Microsoft Word, Apple Mac-Pages, и др.) затоа што тие креираат датотеки што прелистувачот не може да ги прочита. HTML кодот може да се пишува и во други едитори како што е Visual Studio Code кој треба да се инсталира и е бесплатен или во онлајн едитори како што е CodePen.io, OneCompiler и други. Во овој учебник за изучување на HTML ќе се користи Visual Studio Code (VS Code) едиторот со отворен код и претставува индустриски стандард за веб-развој. При пишување код нуди паметни сугестии, автоматски ги прикажува достапните датотеки и ја прикажува веб-страницата во прелистувач кој е го отвора во самиот VS Code.



Слика 1: Изглед на HTML код



5.2.3. ИНСТАЛАЦИЈА НА HTML ЕДИТОР VISUAL STUDIO CODE

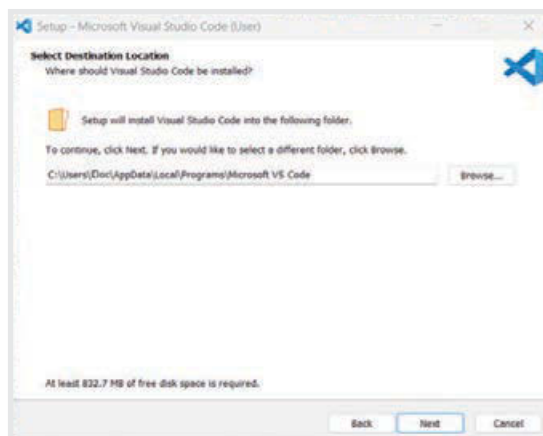
Инсталацијата на Visual Studio Code (VS Code) е мошне брза и едноставна постапка. Отвори ја официјалната веб-страница на Visual Studio Code и одбери верзија која одговара на твојот оперативен систем. Кликни на потребната верзија и почекај неколку секунди додека инсталационата датотека (со екстензија .exe) се преземе на твојот компјутер.

1. Најди ја преземената датотека (на пример, VSCodeUserSetup-x64.exe) во твојот компјутер и кликни двапати (double-click) на неа за да започне процесот на инсталација.
2. Се активира волшебник кој те води низ постапката. Прво се појавува прозорец со Договорот за лиценца (License Agreement). Прочитај го и одбери ја опцијата "I accept the agreement" и потоа кликни на копчето Next (Следно).



Слика 2: Прифаќање услови од договор

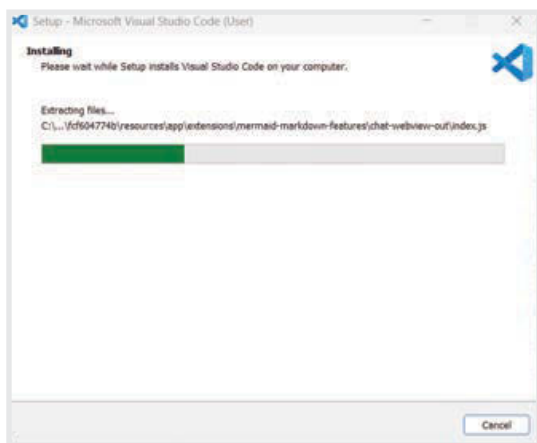
3. Потоа одбери локација на компјутерот каде што ќе ја зачуваш датотеката.



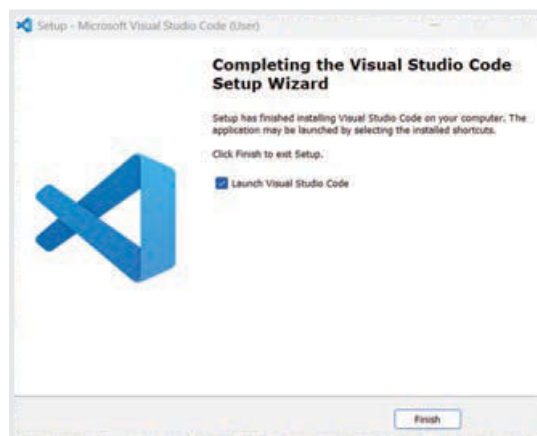
Слика 3: Определување локација за инсталација

4. Кога ќе се отвори прозорецот со наслов "Select Additional Tasks" имаш можност да ги одбереш следните опции:
- Create a desktop icon: За да добиеш кратенка (икона) од VS Code директно на твојот десктоп за брзо отворање.
 - Add "Open with Code" to Windows Explorer... (Има две вакви опции, одбери ги двете). Ова е корисна опција; подоцна, кога ќе имаш папка со HTML проекти, ќе можеш само да кликнеш со десен клик на папката и да избереш „Open with Code“, со што целиот проект веднаш се отвора во едиторот.
 - Register Code as an editor for supported file types: Ова овозможува со двоен клик на кој било .html или .css датотека, тој автоматски да се отвора во VS Code.
 - Add to PATH: Оваа опција е стандардно штиклирана и овозможува програмата да работи правилно со системот.

Откако ќе ги одбереш овие полиња, кликни **Next** за да започне процесот на инсталација.

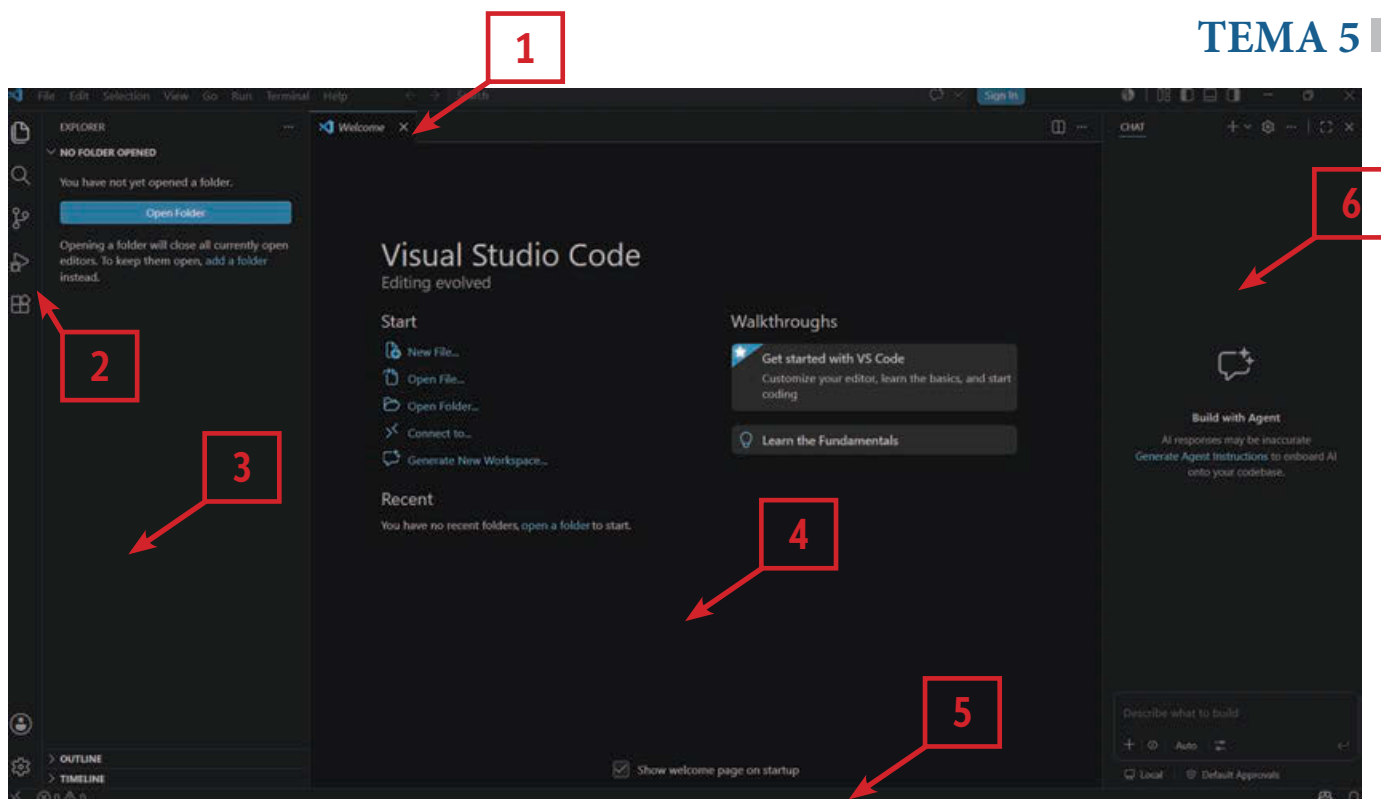


Слика 4: Процес на инсталација



Слика 5: Завршена инсталација на VS Code

5. Процесот на инсталација трае околу 10 до 20 секунди додека зелената линија не се исполни докрај. Со кликање на копчето Finish во едиторот(управувачот) се вклучува почетниот работен прозорец.



Слика 6: Почетен работен прозорец на VS Code

1. Почетен таб: „Welcome“ (Добредојдовте)

На самата средина на екранот ќе се отвори голем таб наречен Welcome. Ова е твојата почетна станица каде што се наоѓаат кратенки за:

- **New File** – за креирање нов, празен документ.
- **Open Folder** – за отворање на папка од компјутерот (најважната опција за програмерите).
- **Clone Git Repository** – за напредни корисници кои преземаат код од интернет.

2. Лента со алатки на крајниот лев раб (Activity Bar)

Ова е тесната, вертикална лента со икони на самиот лев раб од прозорецот. Овие икони се твоите главни „режими на работа“:

- **Explorer (Икона со две папки)**: Ова ќе го користиш 95% од времето. Кога ќе кликнеш тука, се отвора страничниот прозорец каде што ги гледаш твоите папки и датотеки.
- **Search (Луна)**: За пребарување и менување зборови низ сите датотеки одеднаш.
- **Source Control (Гранка/јазол)**: Наменето за Git и верзионирање на кодот.
- **Run and Debug (Бубачка со триаголник)**: За тестирање и барање грешки во кодот.
- **Extensions (Четири коцки од кои едната лета)**: Твојата „продавница“ за бесплатни додатоци. Од тука подоцна ќе инсталираме корисни алатки за полесно пишување код.

3. Страничен прозорец (Sidebar)

Овој простор се наоѓа веднаш десно од лентата со алатки. Ако ти е вклучена опцијата *Explorer*, тука ќе ја гледаш структурата на твојот проект – кои HTML датотеки ги имаш, дали имаш папки за слики и слично. На почетокот, додека немаш отворено папка, тука стои големо сино копче **„Open Folder“**.

4. Главен простор за уредување (Editor)

Ова е најголемиот, централен дел од екранот. Кога ќе кликнеш на некоја датотека (на пример index.html), неговата содржината ќе се отвори токму тука. Ова е твоето „платно“ каде што ќе го пишуваш

и менуваш HTML кодот. Можеш да отвориш повеќе датотеки одеднаш и тие ќе се подредат во табови најгоре, исто како во Chrome.

5. Долна лента за статус (Status Bar)

Во долниот раб од прозорецот има една тенка лента (најчесто сина или виолетова). Таа ти дава моментални информации за датотеката во кој работиш: во кој ред се наоѓаш, кој програмски јазик е активиран (долу десно треба да пишува HTML) и какво е енкодирањето (стандардно пишува UTF-8). Прозорецот е направен со цел да не ти го одвлекува вниманието, лево е организацијата на датотеките, а во центарот на прозорецот е самиот код.

6. Работен простор за работа со ВИ асистент.

5.2.4. РАБОТА СО ЕДИТОР

Постапката за почеток во VS Code е мошне едноставна и брза. Кога првпат ја отвораш програмата, најважно е да запомниш дека **HTML датотеки не можат да „висат“ сами во воздухот** – тие секогаш треба да бидат уредно зачувани во сопствена папка (фолдер) на компјутерот. Подолу се објаснети чекорите еден по еден:

Чекор 1: Направи папка за проектот

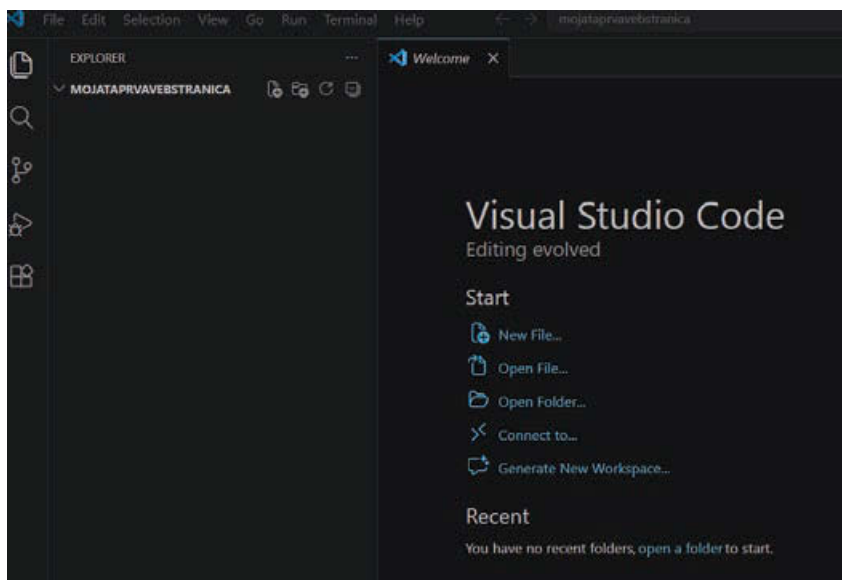
Пред воопшто да го отвориш **Visual Studio Code (VS C)**, направи нова, празна папка на твојот компјутер. Именувај ја едноставно, на пример: `moj-prv-proekt` (најдобро е да користиш мали латинични букви без празни места и име кое ќе асоцира на содржината на веб-страницата).

Чекор 2: Отвори ја папката во VS Code

Сега отвори го VS Code. Имаш два начини да ја внесеш папката внатре:

1. Преку менито **File -> Open Folder...** и избери ја папката што ја направи во Чекор 1.
2. Или едноставно **влечи ја и пушти ја (Drag, Drop)** папката со глумчето директно во празниот простор на VS Code.

Зошто е ова важно? На овој начин VS Code знае дека сè што ќе работиш отсега па натаму припаѓа на тој конкретен проект. Левата страна од екранот станува твојот работен простор (Explorer).

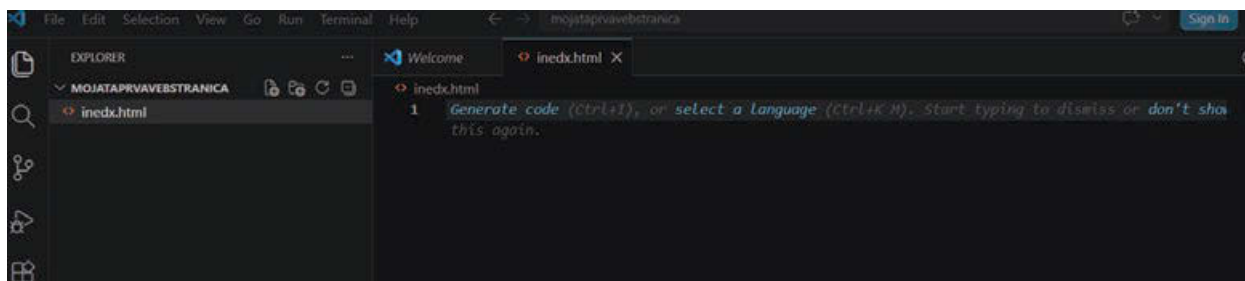


Слика 7: Креирана папка во Visual Studio Code

Чекор 3: Креирај нова HTML датотека

На левата страна, во делот каде што го пишува името на твојата папка, кликни на малата икона за **New File** (изгледа како лист хартија со знакот плус).

Напиши го името на датотеката: **index.html** и притисни Enter.



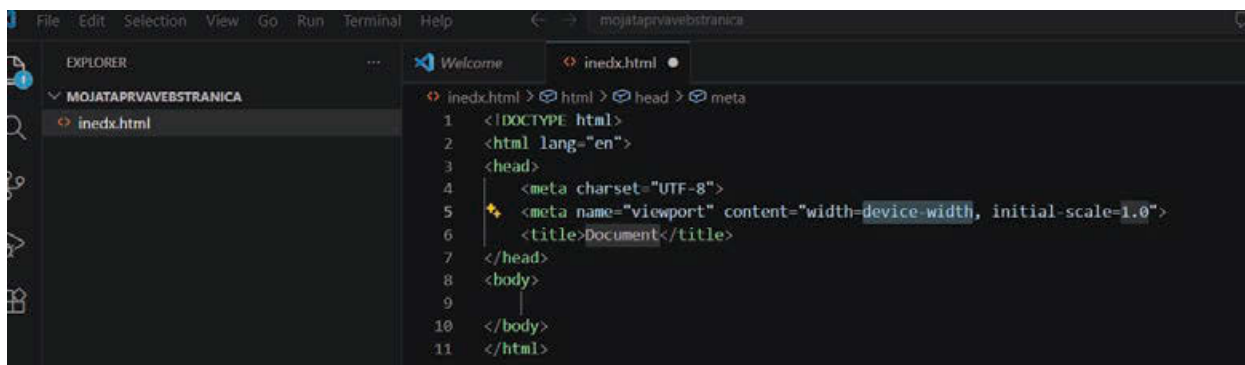
Слика 8: Именувана датотека во Visual Studio Code

Важно: Името треба да завршува со .html за компјутерот да сфати дека ова е веб-страница, а не обичен текстуален документ. Името index е светски стандард за главната (почетна) страница на секој сајт.

Чекор 4: Активирај го почетниот скелет

Сега откако е датотеката отворена и празна:

1. Во првата линија напиши само еден извичник: **!**
2. Ќе се појави мало мени (Emmet кратенка). Притисни **Enter** или **Tab**.
3. VS Code сам ќе го генерира целиот задолжителен почетен код (скелетот со `<html>`, `<head>`, `<body>` и ознаката за UTF-8).



Слика 9: Креиран основен скелет на веб-страница

Чекор 5: Напиши го твојот прв код

Прелистувачот го прикажува на екранот само она што се наоѓа помеѓу почетната `<body>` и завршната `</body>` ознака.

Постави го курсорот под првата ознака `<body>` и напиши:

```
<body>
  <h1>Здраво Свет!</h1>
  <p>Ова е мојата прва веб-страница напишана во VS Code.</p>
</body>
```

Чекор 6: Зачувај го кодот (Save)

Горе на табот каде што пишува index.html, ќе забележиш едно мало бело крукче. Тоа значи дека „Имаш напишано нов код кој сè уште не е зачуван!“.

- Одбери ја кратенката на тастатурата: **Ctrl + S** (за Windows) или **Cmd + S** (за Mac).
- Белото крукче ќе исчезне и ќе се претвори во знакот икс (x). Сега твојот код е безбедно зачуван на хард дискот.

Со ова, твојот прв код е комплетен и подготвен за прегледување.

Има неколку начини за прегледување на веб-страницата, но подолу ќе бидат објаснети три.

Начин 1: Отворање директно од папка. Ова е наједноставниот начин кој функционира на секој компјутер и не бара интернет ниту инсталирање дополнителни додатоци.

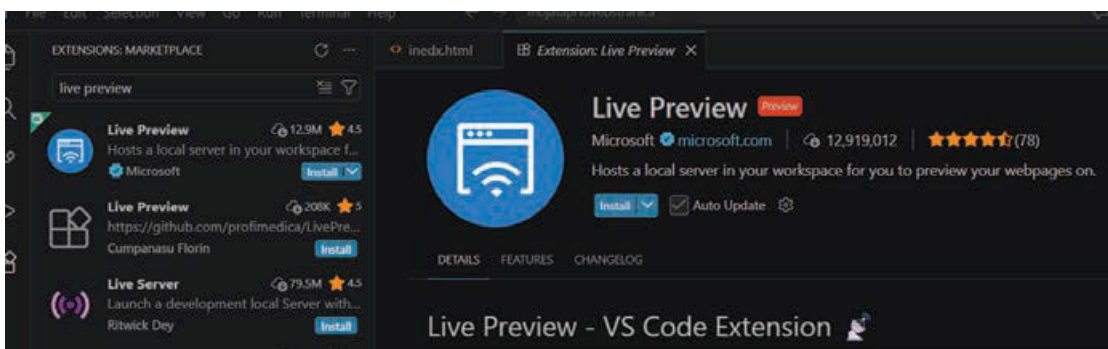
- Отвори ја папката на твојот компјутер каде што го зачува index.html фајлот.
- Кликни два пати на датотеката index.html.
- Компјутерот автоматски ќе ја отвори датотеката во твојот стандарден прелистувач (Chrome, Edge, Firefox).

Важна забелешка за овој начин: Секогаш кога ќе измениш нешто во кодот во VS Code, прво треба да ја зачуваш промената со **Ctrl + S** и потоа треба рачно да го отвориш прелистувачот и да кликнеш на копчето **Refresh** или да одбереш **F5** на тастатурата за да ги видиш промените.

Начин 2: Со додаток „Live Preview“ (преглед директно во VS Code)

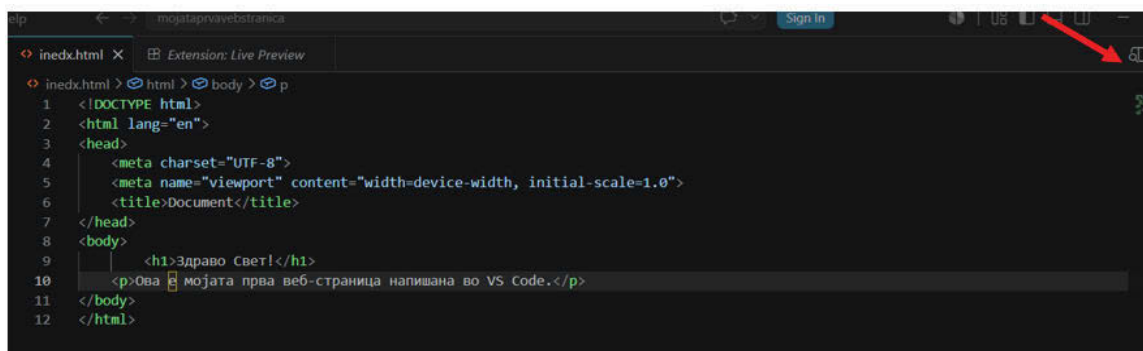
Ако немаш два монитори или не сакаш да го делиш екранот на две половини со Chrome, Microsoft направи додаток кој ти отвора минипрелистувач директно внатре во самата програма.

1. Одбери **Extensions** (четирите коцки лево) и побарај **Live Preview** (официјален додаток од Microsoft).
2. Кликни **Install**.



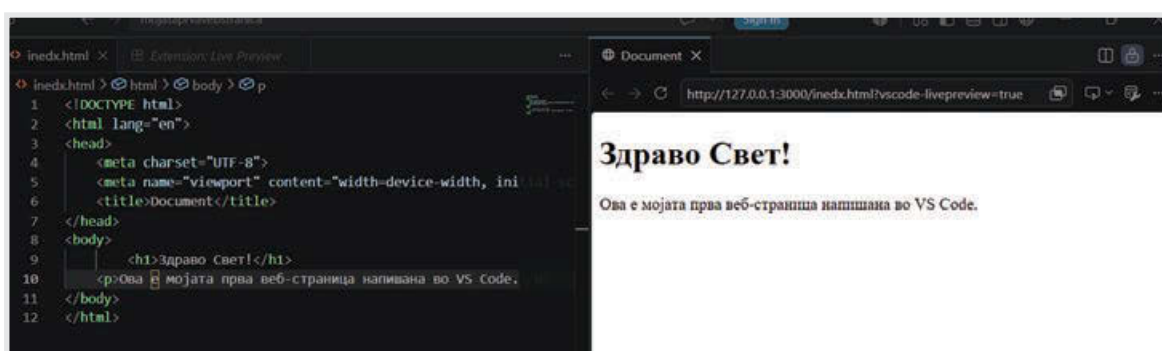
Слика 10: Инсталација на Live Preview

3. Штом ќе заврши инсталацијата, копчето ќе се смени во запчаник (што значи дека е инсталиран). Од тој момент, додатокот е засекогаш вграден во твојот VS Code и е подготвен за користење со еден клик на иконата со мал екран во горниот десен агол
4. Кога ќе се вратиш во твојот index.html код, во горниот десен агол на екранот ќе забележиш нова икона што личи на **мал компјутерски екран со лупа**.



Слика 11: Инсталиран Live Preview

- Клики на таа икона и екранот на VS Code ќе се подели на два дела: лево ќе ти биде кодот, а десно веднаш ќе се појави твојата страница во живо. Секое зачувување веднаш се прикажува таму.



Слика 12: Работен простор и приказ на веб-страница

Начин 3: Преку ВИ асистентот на десната страна

Бидејќи веќе го имаш ВИ прозорчето на десната страна, можеш да ја искористиш и неговата моќ за брз преглед.

- Селектирај го целиот твој HTML код.
- Прашај го ВИ асистентот во чатот: „Како ќе изгледа овој код кога ќе се рендерира?“ или „Show me a preview of this HTML“.
- Многу од современите ВИ агенти (како Copilot или Cursor) имаат вградено копче „**Preview**“ директно во нивниот прозорец, со чиј клик ти покажуваат визуелна слика од тоа како изгледа дизајнот на страницата во тој момент.

За почетно учење најдобар е Начин 2 (Live Preview) бидејќи сè е на едно место и немаш потреба да движиш со глумчето надвор од VS Code.





ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО:

1. Која комбинација на тастатурата е светски стандард во VS Code за зачувување на промените во твојот код?

- А) Ctrl + C
- Б) Ctrl + S
- В) Ctrl + V
- Г) Ctrl + Z

2. Што означува иконата со четири коцки (каде што едната коцка е одвоена) на крајната лева вертикална лента во VS Code?

- А) Мени за пребарување на зборови низ кодот
- Б) Мени за работа со папки и фајлови (Explorer)
- В) Продавница за инсталирање на додатоци (Extensions), како *Live Preview*
- Г) Делот каде што се отвора вештачката интелигенција за разговор

3. Наместо да користиш додатоци за прикажување, ја отвораш твојата веб-страница со два клика директно од папката на компјутерот во прелистувачот. Откако ќе допишеш нов HTML код во VS Code и ќе го зачуваш со Ctrl + S, кој е следниот задолжителен чекор за да ја видиш промената на екранот во Chrome?

- А) Треба целосно да го затвориш прелистувачот и повторно да го отвориш со два клика од папката.
- Б) Треба рачно да кликнеш на копчето Refresh (Освежи / Вчитај повторно) во прелистувачот или да го притиснеш копчето F5 на тастатурата.
- В) Не треба ништо, страницата сама ќе се смени во истиот момент штом ќе одбереш Ctrl + S.
- Г) Треба да ја избришеш старата HTML датотека од папката и да направиш нова.

4. Наместо рачно да го пишуваш целиот почетен скелет на HTML документ, кој знак (кратенка) треба да го напишеш во празен фајл за VS Code (преку Emmet) сам да го генерира тој код со еден притисок на Enter?

- А) Прашалник (?)
- Б) Хаштаг (#)
- В) Извичник (!)
- Г) Коса црта (/)



5. Кога работиш со вградениот ВИ асистент (на десната страна од екранот), кој е најефикасниот начин да му кажеш на агентот да ти објасни или да поправи само една специфична линија од твојот код?

- А) Треба да го препишеш целиот код со рака во прозорчето за разговор.
- Б) Доволно е со глумчето да ја селектираш (означиш) таа линија код на левата страна, и потоа да го поставиш прашањето во десното ВИ прозорче.
- В) Треба да го снимиш фајлот во посебен формат .txt и да му го испратиш на асистентот.
- Г) ВИ асистентите не можат да видат што пишуваш, па требаш да му ја кажеш точната линија (на пр. „Линија 14“).

6. Кој е точниот редослед на чекори што треба да ги направиш за да започнеш нов проект во празен VS Code, така што програмата веднаш да почне да ти помага (да ги бои таговите и да дава предлози)?

- А) Прво пишуваш код, па правиш нова датотека, потоа на крајот го ставаш во папка.
- Б) Прво зачувуваш празен текст на десктоп, па го преименуваш во папка, па го отвораш Chrome.
- В) Прво правиш нова датотека, па пишуваш извичник, па дури на крајот мислиш каде ќе го зачуваш на компјутерот.
- Г) Прво отвораш папка на проектот (**Open Folder**), внатре креираш нова датотека (**New File**) и веднаш го зачувуваш со наставката **.html**.
- Д) Редоследот воопшто не е важен, VS Code сам знае што сакаш да пишуваш уште пред да го вклучиш компјутерот.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Ако често пишуваш една иста структура (на пример, скелет за картичка, табела со специфични класи или мени), нема потреба постојано да ја препишуваш или да правиш Copy/Paste од стари проекти. Можеш да направиш **твоја сопствена Emmet кратенка**. Кликни на запчаникот долу лево (Settings) и избери **User Snippets**. Потоа избери **html.json** и таму можеш да ја дефинираш твојата кратенка. На пример, ако дефинираш кратенка мојакартичка, во моментот кога ќе го напишеш тоа во празен HTML фајл и ќе притиснеш Tab, VS Code автоматски ќе го вчита целиот твој предефиниран код со наслови, класи и слики. Исто така, кога твојата HTML датотека ќе има 500 или 1000 линии, „скролањето“ нагоре-надолу станува кошмар. Напредните програмери користат паметна навигација.

1. **Breadcrumbs (Трошки леб):** Најгоре над кодот секогаш гледаш патека (на пр. src > index.html > body > div.container > table). Можеш да кликнеш на која било ознака во таа патека за веднаш да „скокнеш“ директно во него, без скролање.
2. **Outline Мени:** Во долниот лев агол (под папките) има мени наречено Outline. Тоа ти го прикажува HTML документот како семејно стебло. Ако кликнеш на некој таг во тоа дрво, едиторот веднаш те носи таму.

5.2.5. ПРИМЕНА НА HTML ОЗНАКИ

Во HTML јазикот, за да се додадат дополнителни карактеристики на ознаки се користат атрибути. Ако самите ознаки се скелетот на веб-страницата (кажуваат каде има наслов, каде параграф), атрибутите им даваат дополнителни информации на тие ознаки. Тие секогаш се пишуваат внатре во почетната ознака (никогаш во завршната). Се пишуваат во форма: име_на_атрибут="вредност" одонсно имаат име и вредност. Името на атрибутот се пишува со мали букви а вредноста на атрибутот треба секогаш да биде помеѓу наводници („“) или полунаводници ('). Почесто се користат наводниците. Полунаводниците се користат кога во вредноста има наводници, на пример `ime='булевар „8 Септември“ '`.

Исто така, за ознака за линк (`<a>`) да знае каде да нè „однесе“, го користиме атрибутот `href`.

```
<a href="https://www.google.com">Оди на Google</a>
```

Се користат и други атрибути за хиперврска, `target="_blank"` – го отвора линкот во **нов таб**, за корисникот да не ја напушти твојата страница а **title** – дава мал текстуален совет (tooltip) кога ќе го задржиш глумчето над линкот.

```
<a href="https://mk.wikipedia.org" title="Оди на Википедија">Посети ја Википедија</a>
(едноставна хиперврска)
```

```
<a href="https://www.google.com" target="_blank">Отвори го Google во нов таб</a>
(хиперврска што се отвора во нов таб)
```

Ознака за слика (`<img>`) е специфична бидејќи е самостојна (нема завршна ознака). За да прикаже слика, ги користи атрибутите `src` (извор на сликата) и `alt` (алтернативен текст):

```

```

Постојат атрибути и за ширина и големина на сликата, **width и height** со кои големината на сликата изразена во пиксели.

```

```

Еден од најчестите елементи на интернет е кога сакаш **кликот на самата слика да те однесе на некој линк** (како кога кликуваш на лого за да се вратиш на почетната страница).

За да го направиш ова, едноставно ја вгнездуваш ознаката `<img>` внатре, помеѓу почетната и завршната ознака на `<a>`.

```
<a href="https://www.youtube.com" target="_blank">
  
</a>
```

Сите овие ознаки може да се комбинираат во еден код и да се креира интересна страница. На пример, со следниот код се креира страница со слика и хиперврска која води до повеќе информации за сликата:

```
<h1>Слика од мало кученце</h1>



<br> <a href="https://mk.wikipedia.org/wiki/Куче" target="_blank">Прочитај повеќе за кучињата на
Википедија</a>
```


За креирање табели со HTML се користат посебни атрибути за соединување на ќелиите. Кога правиме табела, користиме една главна ознака во која се вгнездуваат неколку помошни ознаки за редови и колони:

Главни ознаки за табела:

- **<table>** – Ја започнува и ја завршува целата табела.
- **<tr>** (Table Row) – Прави нов **хоризонтален ред**.
- **<td>** (Table Data) – Прави обична **ќелија (колона)** во тој ред каде што стојат податоците.
- **<th>** (Table Header) – Прави **ќелија за наслов** (текстот автоматски станува задебелен и центриран).

Ако креираме распоред на часови и еден предмет се изучува на два часа тогаш треба да соединиме две колони или два реда. За ова ги користиме следниве два атрибути кои се пишуваат исклучиво внатре во **<td>** или **<th>** ознаките:

Атрибутот **colspan** (Спојување на колони)

Овој атрибут му кажува на полето да се „растегне“ хоризонтално на повеќе колони одеднаш.

- **Пример:** Ако имаш блок часови по Информатика, наместо да правиш две посебни полиња, правиш едно вакво:

```
<td colspan="2">Информатика (блок час)</td>
```

Атрибутот **rowspan** (Спојување на редови)

Овој атрибут го „растегнува“ полето вертикално, односно надолу низ редовите.

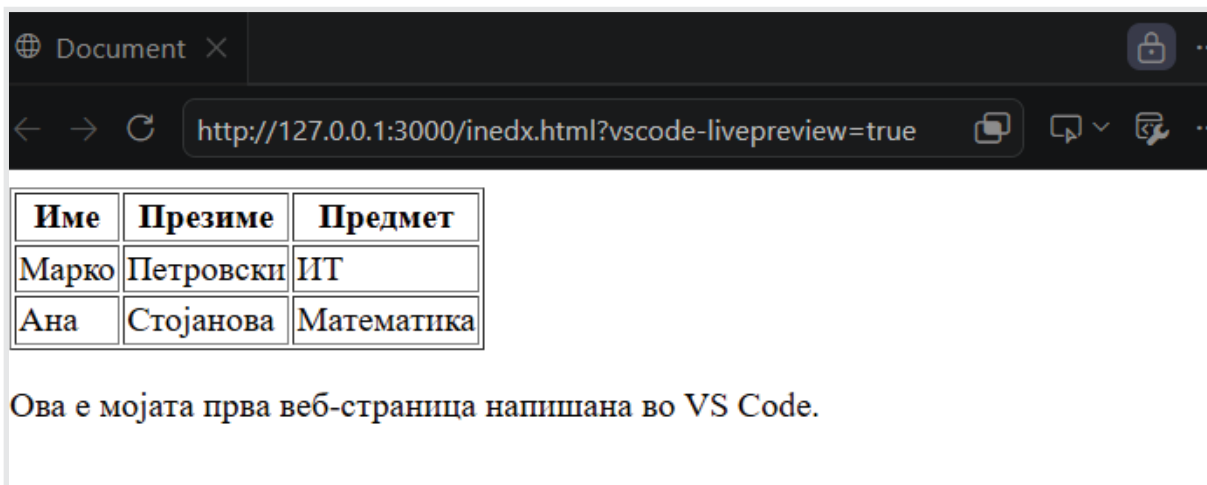
- **Пример:** Ако секој ден во исто време имаш „Голем одмор“, можеш да ги споиш редовите за деновите:

```
<td rowspan="3">Голем одмор</td>
```

Целосен код за поврзување на сите овие ознаки и нивните атрибути:

```
<table border="1"> <tr>
<th>Име</th>
<th>Презиме</th>
<th>Предмет</th>
</tr>
<tr>
<td>Марко</td>
<td>Петровски</td>
<td>ИТ</td>
</tr>
<tr>
<td>Ана</td>
<td>Стојанова</td>
<td>Математика</td>
</tr>
</table>
```





Слика 13: Распоред креиран со HTML

Знаењето за креирање табели може да го искористиме за креирање форма за пријавување на курс која ќе содржи табела, внесови и соединување. Овој пример е уште понапреден бидејќи користи табела без рамка (невидлива мрежа) за совршено да ги порамни полињата за внесување, а на крајот користи colspan за да го рашири копчето низ двете колони.

<h2>Пријава за курсот по HTML ознаки</h2>

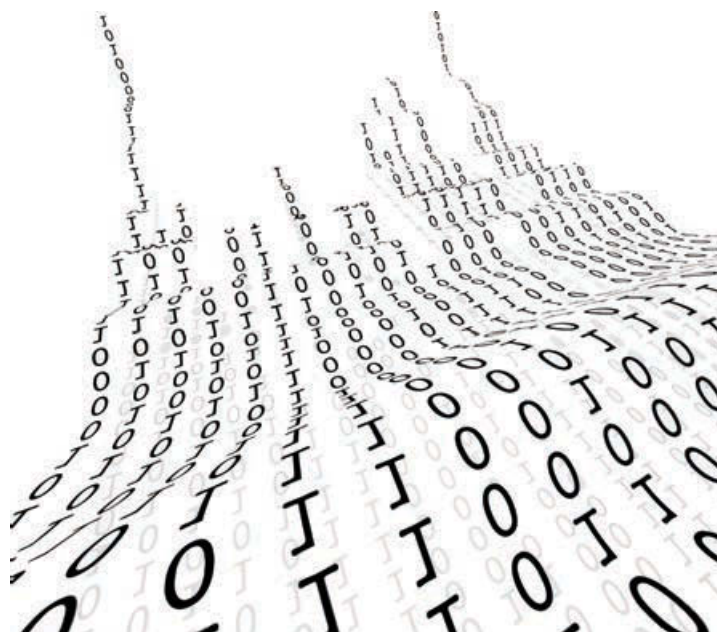
<p>Ве молиме пополнете ги задолжителните полиња за да го резервирате вашето место.</p>

```
<table>
<tr>
  <td><b>Име и презиме:</b></td>
  <td><input type="text" placeholder="Внесете го вашето име..." required></td>
</tr>

<tr>
  <td><b>Е-пошта:</b></td>
  <td><input type="email" placeholder="пример@емаил.ком" required></td>
</tr>

<tr>
  <td><b>Датум на раѓање:</b></td>
  <td><input type="date"></td>
</tr>

<tr>
  <td colspan="2" style="text-align: center;">
    <br>
    <input type="submit" value="Испрати ја пријавата">
  </td>
</tr>
</table>
```



Во табелите подолу прикажани се сите ознаки и атрибути со нивните описи кои се користат во HTML.

Ознака	Што означува?	Опис / Како се користи?	Има ли завршна ознака?
<h1> до <h6>	Наслови	<h1> е најголем главен наслов, <h6> е најмал поднаслов.	Да (на пр. </h1>)
<p>	Параграф	Се користи за пишување обичен текст.	Да (</p>)
<a>	Хиперлинк	Прави линк на кој може да се кликне.	Да ()
	Слика	Вметнува слика на страницата.	Не (самостојна е)
<table>	Табела	Ја започнува и ја затвора целата табела.	Да (</table>)
<tr>	Ред во табела	Прави нов хоризонтален ред внатре во табелата.	Да (</tr>)
<td>	Поле (ќелија)	Обично поле со податоци внатре во редот.	Да (</td>)
<th>	Насловно поле	Поле за наслов на колона (текстот е задебелен).	Да (</th>)

	Нов ред	Го спушта текстот или елементот во нов ред.	Не (самостојна е)

Табела 2: Ознаки во HTML

Атрибут	На која ознака се става?	Што контролира? (Објаснување)	Пример како се пишува
href	На ознаката за линк (<a>)	Кажува каде (на кој линк) да нè однесе кога ќе кликнеме.	href="https://google.com"
target	На ознаката за линк (<a>)	Вредноста "_blank" кажува линкот да се отвори во нов таб.	target="_blank"
src	На ознаката за слика ()	Ја кажува точновата патека или линк од каде се зема сликата.	src="kuche.jpg"
alt	На ознаката за слика ()	Алтернативен текст кој ја опишува сликата со зборови.	alt="Црно кученце"
width	На слика () или видео	Ја одредува ширината на елементот во пиксели.	width="300"
colspan	На полињата во табела (<td> / <th>)	Спојува повеќе колони хоризонтално (на пр. за двоен час).	colspan="2"
rowspan	На полињата во табела (<td> / <th>)	Спојува повеќе редови вертикално во табелата.	rowspan="2"
border	На ознаката за табела (<table>)	Додава видлива рамка (линии) околу табелата.	border="1"
title	На која било ознака	Прикажува мало текстуално облаче кога ќе го оставиш глумчето озгора.	title="Повеќе инфо"

Многу лесно се креира едноставна презентација на производ или книга. Се користи наслов, параграф со опис, слика од корицата, табела со технички детали за книгата и линк за купување што се отвора во нов таб.

<h1>Книга: Дигитален Свет</h1>

<p title="Краток опис на содржината">Ова е возбудлива книга за почетоците на компјутерската технологија и развојот на интернетот.</p>

<h3>Технички детали</h3>

<table border="1">

<tr>

<th>Карактеристика</th>

<th>Информација</th>

</tr>

<tr>

<td>Автор</td>

<td>Марко Петровски</td>

</tr>

<tr>

<td>Јазик</td>

<td>Македонски</td>

</tr>

<tr>

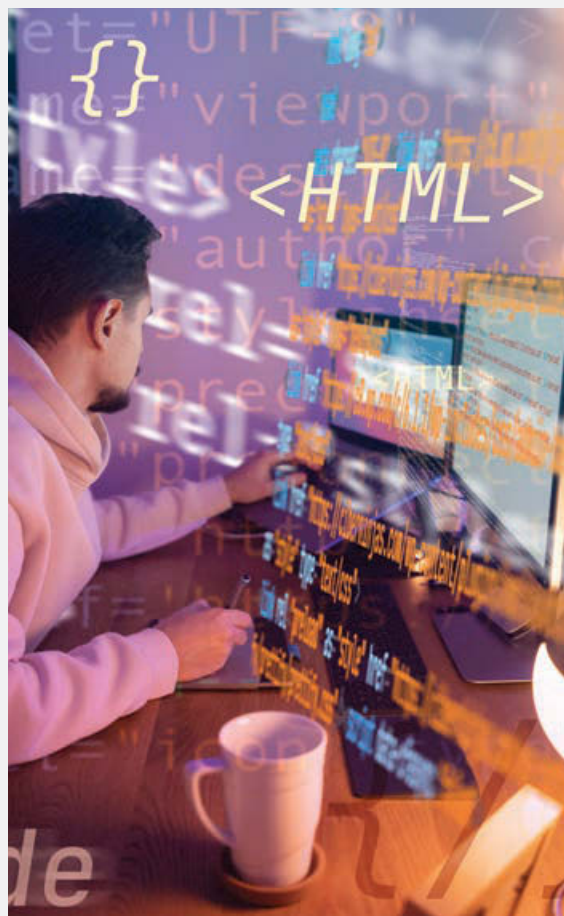
<td>Број на страници</td>

<td>240 страници</td>

</tr>

</table>

Кликувај тука за да ја нарачаш книгата



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО:

1. Каде СЕКОГАШ се пишуваат HTML атрибутите?

- А) На самиот крај на документот, по ознаката </html>
- Б) Внатре во почетната ознака (opening tag)
- В) Внатре во завршната ознака (closing tag)
- Г) Помеѓу почетната и завршната ознака, заедно со текстот

2. Која од следниве ознаки за табела се користи за креирање нов хоризонтален ред?

- А) <table>
- Б) <td>
- В) <tr>
- Г) <th>

3. Со свои зборови, објасни која е разликата помеѓу HTML ознака (tag) и HTML атрибут (attribute). Како соработуваат тие за да прикажат нешто на екранот?

4. Ако на една слика ѝ го зададеме атрибутот `width="300"`, а го изоставиме атрибутот `height`, што ќе се случи со висината на сликата во прелистувачот?

- А) Сликата нема да се прикаже воопшто бидејќи недостасува висината.
- Б) Висината ќе стане 0 пиксели и сликата ќе биде невидлива.
- В) Прелистувачот сам ќе ја приспособи висината пропорционално за да не се искриви сликата.
- Г) Сликата ќе се растегне преку целиот екран по вертикала.

5. Сакаш да поставиш слика (`logotip.png`) која кога корисникот ќе кликне на неа, ќе го однесе на веб-страницата `https://www.google.com` во сосема нов таб. Напиши го точниот HTML код за оваа намена.

6. Кој од следниве кодови правилно прикажува линк со алтернативен опис кога ќе се помине со глумчето над него?

- А) `<a src="https://sajtedu.mk" alt="Училиште">Оди на сајтот</a>`
- Б) `<a href="https://sajtedu.mk" title="Посети го нашето училиште">Оди на сајтот</a>`
- В) `<a target="https://sajtedu.mk">Оди на сајтот</a>`
- Г) `<link href="https://sajtedu.mk" title="Училиште">Оди на сајтот</link>`

7. Разгледај го следниов HTML код за табела:

HTML

```
<table border="1">
<tr>
<td>Македонски</td>
<td colspan="2">Математика (двоен час)</td>
</tr>
<tr>
<td>Англиски</td>
<td>ИТ</td>
<td>Ликовно</td>
</tr>
</table>
```

Анализирај го кодот и објасни зошто оваа табела визуелно ќе се искриви (деформира) во прелистувачот. Каде е грешката во броењето на полињата?

8. Замисли дека правиш веб-страница за лица со оштетен вид кои користат софтверски „читачи на екран“ (Screen Readers). Кој од следниве два чекори е клучен за нив и зошто?

- А) Секогаш да се користи атрибутот `target="_blank"` на линковите.
- Б) Секогаш да се пишува прецизен и описен текст во атрибутот `alt` на ознаката `<img>`.
- В) Табелите секогаш да имаат `border="1"`.
- Г) Сите наслови да бидат исклучиво `<h1>`.

9. Дизајнирај HTML код за наједноставна онлајн визит-карта (профил). Кодот треба да содржи: твојот омилен наслов, еден параграф за тебе, твоја слика и еден линк до твојот омилен сајт. Внимавај на правилното отворање и затворање на ознаките и точната синтакса на атрибутите.

10. Замисли дека треба да направиш распоред за утрешниот ден во училиште, каде што 3-тиот и 4-тиот час ти се блок-часови (двоен час) по истиот предмет. Напиши го само HTML кодот за тој специфичен ред (`<tr>`) од табелата, користејќи го соодветниот атрибут за спојување.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Ознаката `<a>` со атрибутот `href` не служи само за прелистување од една веб-страница на друга. Таа може да комуницира со оперативниот систем на компјутерот или телефонот на корисникот.

Врска за директно испраќање е-пошта (`mailto:`): Ако сакаш кога некој ќе кликне на линкот веднаш да му се отвори програмата за е-пошта (како Outlook или Gmail) со веќе напишана адреса, го користиш ова:

```
<a href="mailto:kontakt@skolo.mk">Испрати ни е-пошта</a>
```

Врска за директно јавување (`tel:`): Ова е генијално за корисници кои го гледаат сајтот од мобилен телефон. Кога ќе кликнат на линкот, телефонот автоматски почнува да го врти бројот:

```
<a href="tel:+3892123456">Јави се веднаш</a>
```

Ако имаш многу долга веб-страница и сакаш со еден клик корисникот да „скокне“ најдолу на контактот, користиш комбинација од `href` и атрибутот `id`:

```
<a href="#kontakt-sekcija">Скокни до контакт</a>
```

```
<h2 id="kontakt-sekcija">Тука се нашите податоци</h2>
```

Експериментирај со паметни слики со атрибутот `loading="lazy"`, со ознаки за напреден текст кој компјутерот го „разбира“, атрибутот `download` и други.

5.2.6 CSS – СТИЛОВИ НА ВЕБ-СТРАНИЦИ

Додека HTML е одговорен за структурата (скелетот) на веб-страницата, CSS (Cascading Style Sheets) е јазикот кој е одговорен за нејзиниот изглед, дизајн и убавина. Без CSS, сите веб-страници би изгледале како обичен црно-бел текст во Notepad со сини линкови.

CSS опишува како треба да се прикажат HTML ознаките на екранот. Постојат три начини на кои можеме да додадеме CSS во нашиот HTML документ:

- **Инлајн (Inline):** Директно внатре во HTML ознаката преку атрибутот `style`. (Пример: `<p style="color: red;">`).
- **Внатрешен (Internal):** Внатре во самиот HTML документ, во специјална ознака `<style>` сместена во делот `<head>`.
- **Надворешен (External) – Професионален начин:** Кодот се пишува во посебен фајл со наставка `.css` (на пример: `style.css`), кој потоа се поврзува со HTML преку ознаката `<link>`.

За да обоиме или промениме некој елемент, прво треба да му кажеме на компјутерот кој елемент го менуваме. Тоа го правиме со помош на **селектори**.

Основни селектори:

- **Селектор по ознака (Element selector):** Ги избира сите ознаки од тој тип на страницата.

```
p {  
  color: blue; /* Ги прави сите параграфи сини */  
}
```

- **Селектор по класа (Class selector):** Таргетира група елементи кои го имаат атрибутот class. Секогаш почнува со **точка (.)**.

```
.crven-tekst {
  color: red;
}
```

- **Селектор по идентификатор (ID selector):** Таргетира само еден уникатен елемент со атрибутот id. Секогаш почнува со **тараба (#)**.

```
#glaven-naslov {
  font-size: 30px;
}
```

Во CSS, бојата на текстот се менува со наредбата color. Боите може да ги дефинираме на три начини:

1. Со **име на бојата**: red, blue, green, orange.
2. Со **HEX код** (со тараба): #ff0000 (црвена), #ffffff (бела), #000000 (црна).
3. Со **RGB вредности**: rgb(255, 0, 0).

```
h2 {
  color: #2c3e50; /* Темносива нијанса */
}
```

Уредувањето на текстот е клучно за убав изглед на веб-страницата. Најважни наредби се:

- **font-family:** Избор на стил на буквите (на пр. Arial, Verdana, Times New Roman). Се препорачува секогаш да се стави генеричко семејство како sans-serif (без украси) или serif (со украси) како замена ако компјутерот го нема точниот фонт.
- **font-size:** Големина на буквите, најчесто изразена во пиксели (px).
- **font-weight:** Задебелување на текстот (bold за задебелен, normal за обичен).

```
body {
  font-family: Arial, sans-serif;
  font-size: 16px;
}
```

Заднината на целата страница (ознаката <body>) или на некој специфичен елемент (како табела или параграф) се уредува со:

- **background-color:** Ја менува бојата на заднината.
- **background-image:** Поставува слика како заднина преку линк (url('slika.jpg')).

```
body {
  background-color: #f4f4f4; /* Светлосива нежна заднина */
}
```

Секој елемент во HTML (наслов, слика, параграф) компјутерот го гледа како една **невидлива правоаголна кутија**. Распоредот и просторот околу тие кутии го контролираме со три клучни наредби:

- **padding (Внатрешна маргина):** Просторот **внатре** во кутијата, помеѓу текстот и нејзината рамка.
- **border (Рамка):** Линијата околу елементот (може да биде со дебелина, стил и боја, на пр. 2px solid black).

- **margin (Надворешна маргина):** Просторот **надвор** од кутијата, кој прави растојание помеѓу овој елемент и другите елементи околу него.

```
.kutija {  
  border: 1px solid #ccc;  
  padding: 20px; /* Текстот внатре ќе биде одвоен од рамката */  
  margin: 15px; /* Кутијата ќе се оддалечи од другите елементи */  
}
```

Во следниот пример е прикажано како се користат наредбите за промена на бојата на буквите и заднината.

```
<!DOCTYPE html>  
<html lang="mk">  
<head>  
  <meta charset="UTF-8">  
  <title>Едноставен CSS пример</title>  
</head>  
<body style="background-color: #f0f0f0; font-family: Arial;">  
  
  <h1 style="color: blue;">Мојот прв стилизиран наслов</h1>  
  
  <p style="background-color: red; color: white; padding: 10px;">  
    Овој текст има црвена заднина и бели букви!  
  </p>  
  
</body>  
</html>
```

Која од наредбите го дефинира јазикот на кој ќе се прикаже текстот? На каков начин е додаден CSS?

Кога сакаме да стилизираме конкретни (специфични) елементи на страницата, подобро е наместо сите одеднаш секој одделно да се дефинира. За таа цел, се користат два различни типа на „означувачи“ (селектори): **Класа (Class)** и **Идентификатор (ID)**.

```
<!DOCTYPE html>  
<html lang="mk">  
<head>  
  <meta charset="UTF-8">  
  <title>Селектори по Класа и ИД</title>  
<style>  
  /* Селектор за КЛАСА (почнува со точка) */  
  .vazno-izvestuvanje {  
    background-color: yellow;  
    padding: 8px;  
  }
```

```

border: 1px solid orange;
}

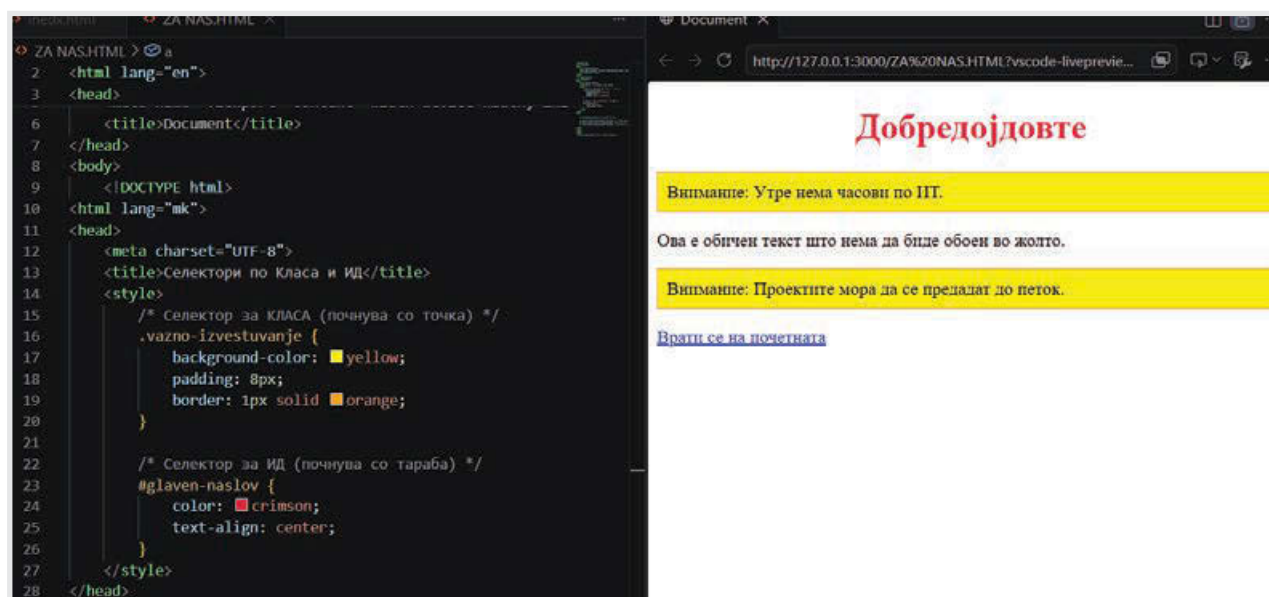
/* Селектор за ИД (почнува со тараба) */
#glaven-naslov {
  color: crimson;
  text-align: center;
}
</style>
</head>
<body>

<h1 id="glaven-naslov">Добредојдовте</h1>

<p class="vazno-izvestuvanje">Внимание: Утре нема часови по ИТ.</p>
<p>Ова е обичен текст што нема да биде обоен во жолто.</p>
<p class="vazno-izvestuvanje">Внимание: Проектите треба да се предадат до петок.</p>

</body>
</html>

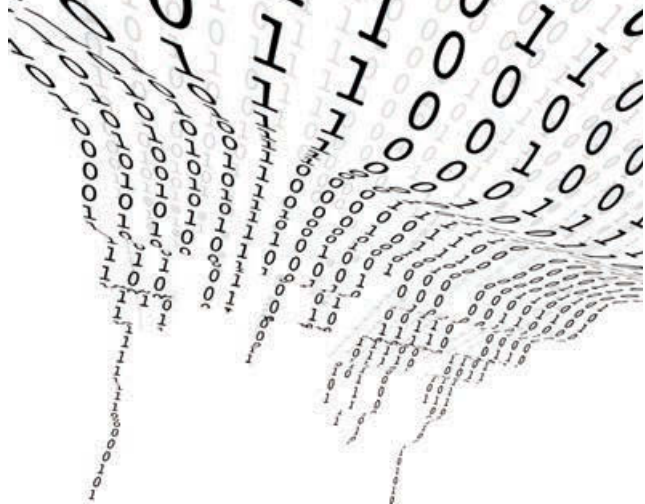
```



Слика 13: Стилизирање со класа и идентификатор



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО:



1. Што означува скратеницата CSS?

- A) Computer Style Sheets
- Б) Creative Style Systems
- B) Cascading Style Sheets
- Г) Colorful Style Software

2. Со кој симбол започнува селекторот за КЛАСА (class) во CSS?

- A) Со тараба (#)
- Б) Со точка (.)
- B) Со извичник (!)
- Г) Со прашалник (?)

3. Објасни ја разликата во дејството помеѓу селекторот `p` и селекторот `#glaven-paragraph`. Кој елемент на страницата ќе го промени секој од нив?

4. Ако на еден елемент му доделиш `background-color: black;` и намерно ја заборавиш наредбата `color`, зошто текстот во тој елемент ќе стане невидлив за корисникот?

5. Напиши го точниот CSS код со кој сите наслови од типот `<h2>` на веб-страницата ќе добијат зелена боја на текстот, големина од 24 пиксели и фонт Arial.

6. Кој од следните CSS кодови правилно поставува црна рамка со дебелина од 2 пиксели околу една слика?

- A) `img { margin: 2px black; }`
- Б) `img { border: 2px solid black; }`
- B) `img { padding: 2px solid; }`
- Г) `img { line: 2px black; }`

7. Разгледај ја следнава ситуација: Имаш две кутии една под друга. Сакаш да направиш поголемо растојание (празен простор) помеѓу нив. Која наредба од „Box Model“-от треба да ја искористиш и зошто наредбата `padding` нема да ти заврши работа во овој случај?

8. Твој соученик го напишал целиот CSS код преку инлајн стилови (директно во секоја HTML ознака со `style="..."`). Проектот има 5 различни веб-страници. Оценете го овој пристап: Што ќе се случи ако професорот побара од него утре да ја смени заднината на сите 5 страници од сива во бела? Дали надворешниот CSS фајл би бил подобар избор?

9. Дизајнирај комплетен CSS стил за класа наречена `.karticka-proizvod`. Кутијата треба да има светлосива заднина, внатрешно растојание (`padding`) од 15 пиксели, тенка рамка и текстот внатре да биде со фонт Verdana.

10. Напиши краток HTML и CSS код каде што ќе креираш еден главен наслов кој ќе има уникатен идентификатор `id="naslov-акција"`. Во CSS делот, направи овој наслов да биде со задебелени букви (**bold**) и да има жолта заднина (`background-color`).

5.2.7. РАСПОРЕД И АДАПТИВЕН (RESPONSIVE) ДИЗАЈН

Кога креираме веб-страница, не е доволно само да ги обоиме елементите. Клучно е да знаеме како да ги распоредиме на екранот (лево, десно, едно до друго) и како тој распоред автоматски да се приспособи кога страницата се гледа на компјутер, на таблет или на мобилен телефон. Тоа се постигнува со помош на соодветен распоред и адаптивен дизајн.

Распоред претставува структура или „скелет“ на веб-страницата. Тој одредува каде ќе стои логото, каде менито, каде главниот текст, а каде страничната лента (sidebar).

Во CSS, елементите стандардно се редат еден под друг (блок елементи како <div>, <h1>, <p>). За да направиме поинаков распоред, ја користиме наредбата **display**. Најсовремен и наједноставен начин за подредување елементи е **Flexbox** (Flexible Box). Кога на една родителска кутија ќе ѝ ја дадеме наредбата `display: flex;`, сите нејзини внатрешни елементи автоматски се подредуваат **едно до друго (во колони)**, наместо еден под друг.

```
.kontejner {
  display: flex;
  justify-content: space-between; /* Ги оддалечува елементите рамномерно лево и десно */
}
```

Респонзивен веб-дизајн (Responsive Web Design) е пристап во креирањето веб-страници кој овозможува нивниот изглед и содржината автоматски да се приспособуваат на големината на екранот на уредот од кој се пристапува – без разлика дали станува збор за паметен телефон, таблет, лаптоп или голем десктоп монитор.

Наместо да се прават посебни верзии на страницата за мобилни телефони и за компјутери, со респонзивниот дизајн се користи еден ист код кој флексибилно се прередува и се собира или се раширува во зависност од просторот.

Главни принципи на респонзивниот дизајн:

1. Флуидни мрежи (Fluid Layouts): Наместо да користиме фиксни димензии во пиксели (на пр. `width: 900px;`), користиме проценти (`width: 100%;` или `width: 50%;`). Така елементите се шират или се стеснуваат во зависност од големината на екранот.
2. Флуидни слики: За сликата никогаш да не излезе надвор од екранот на телефонот, ѝ го даваме следново златно правило во CSS:

```
img {
  max-width: 100%;
  height: auto;
}
```

Зошто е клучен за современите веб-страници?

- Доминација на мобилниот сообраќај: Повеќе од половина од глобалниот интернет-сообраќај доаѓа од мобилни уреди. Ако вашата страница изгледа лошо или е тешка за навигација на телефон, губите огромен број потенцијални посетители.
- Подобро корисничко искуство (UX): Корисниците не сакаат постојано да зумираат, да скролаат хоризонтално или да притискаат ситни копчиња. Респонзивниот дизајн нуди чист преглед, лесно читлив текст и едноставна навигација на секој уред.

- Важност за SEO (оптимизација за пребарувачи): Пребарувачите како Google користат „mobile-first indexing“. Тоа значи дека Google примарно ја оценува мобилната верзија на вашата страница кога одлучува на кое место ќе ја рангира во резултатите од пребарувањето.
- Полесно одржување и економичност: Наместо да трошите време и пари за развој и ажурирање на две различни веб-страници (една за десктоп, друга за мобилен), одржувате само еден систем кој работи секаде.

Во денешно време, респонзивниот дизајн веќе не е само „додаток“ или луксуз, туку е основен стандард за секоја професионална веб-страница која ќе биде успешна и видлива на интернетот.

Најважната алатка во CSS за приспособување на веб-страницата се вика **Media Queries** (медиумски барања). Тие работат како филтри или услови: „Ако е екранот помал од 600 пиксели, тогаш примени ги овие нови CSS правила“. Замисли дека на компјутер имаш три елементи-кутии поставени една до друга (flex). На мобилен уред екранот е тесен и тие елементи-кутии ќе се „нагмечат“. Со помош на Media Query, им кажуваме да се наредат **една под друга** на мобилниот уред.

```
/* Стандарден CSS за компјутери (елементите се еден до друг) */
```

```
.glavno-meni {
  display: flex;
  flex-direction: row;
}
```

```
/* Специјално правило за мобилни телефони (екрани потесни од 600px) */
```

```
@media (max-width: 600px) {
  .glavno-meni {
    flex-direction: column; /* Ги вртиме елементите да се наредат еден под друг */
  }
}
```

Секој елемент/кутија во HTML го има таканаречениот Box Model (модел на кутија). Дури и да не гледаш рамка, компјутерот за секој елемент пресметува четири работи:

- **Содржина (content):** Самиот текст или слика.
- **Внатрешна маргина (padding):** Празниот простор околу текстот за тој да не биде залепен за рабовите.
- **Рамка (border):** Линијата околу елементот (може да биде невидлива, а може и да ја обоиме).
- **Надворешна маргина (margin):** Просторот што ја оддалечува таа кутија од соседните кутии за да не се допираат.

Кога дизајнираме веб-страница, ние всушност само си играме со овие кутии – ги шириме, ги стеснуваме, ги боиме нивните заднини и одлучуваме дали ќе стојат во иста линија или ќе бидат една под друга!

Во примерот подолу е прикажан адаптивен дизајн на веб-страница. Искористи го кодот во VSC едиторот и тестирај ја големината на прозорецот со намалување и зголемување за да воочиш како функционираат display: flex; и @media.

```

<!DOCTYPE html>
<html lang="mk">
<head>
  <meta charset="UTF-8">
  <title>Краток Адаптивен Дизајн</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      padding: 20px;
    }

    /* КОНТЕЈНЕР: На компјутер ги реди картичките хоризонтално (во ред) */
    .grupacija-kutii {
      display: flex;
      flex-direction: row;
      gap: 15px; /* Растојание помеѓу кутиите */
    }

    /* КАРТИЧКА: Стил за секоја поединечна кутија */
    .karticka {
      background-color: white;
      padding: 20px;
      border: 1px solid #ddd;
      flex: 1; /* Им кажува на кутиите да ја поделат ширината еднакво */
    }

    /* МЕДИУМСКО ПРАШАЊЕ: За екрани помали од 600 пиксели (мобилни) */
    @media (max-width: 600px) {
      .grupacija-kutii {
        flex-direction: column; /* ГИ ВРТИ КУТИИТЕ ЕДНА ПОД ДРУГА */
      }
    }
  </style>
</head>
<body>

  <h2>Нашиот Адаптивен Распоред</h2>

  <div class="grupacija-kutii">
    <div class="karticka">

```



```

    <h3>Кутија 1</h3>
    <p>Ова е содржината на првата кутија.</p>
  </div>
  <div class="karticka">
    <h3>Кутија 2</h3>
    <p>Ова е содржината на втората кутија.</p>
  </div>
  <div class="karticka">
    <h3>Кутија 3</h3>
    <p>Ова е содржината на третата кутија.</p>
  </div>
</div>

</body>
</html>

```



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО:

1. Која CSS наредба се користи за да се активира флексибилниот модел на распоред (Flexbox)?
 - A) display: block;
 - Б) display: flex;
 - B) position: static;
 - Г) float: left;
2. Што се постигнува со користење на CSS правилото @media (Media Queries)?
 - A) Се додаваат видео- и аудиофајлови на страницата.
 - Б) Се применуваат различни CSS стилови во зависност од големината на екранот.
 - B) Се менува јазикот на веб-страницата.
 - Г) Се поврзува страницата со социјалните мрежи.
3. Зошто е подобро да користиме проценти (%) наместо фиксни пиксели (px) за ширината на главните контејнери кога дизајнираме респонзивна веб-страница?
4. Објасни ја разликата во однесувањето на елементите кога flex-direction е поставено на row наспроти кога е поставено на column.
5. Напиши го CSS правилото кое ќе овозможи сите слики () на страницата да бидат респонзивни (да не ја надминуваат ширината на својот родител и да го задржат соодносот на големината).
6. Кој од следниве кодови правилно таргетира уреди чија максимална ширина на екранот е 768 пиксели (како таблети)?
 - A) @media (width: 768px)
 - Б) @media (max-width: 768px)
 - B) @screen (max-device: 768px)
 - Г) @media (min-width: 768px)

7. Ученик направил распоред каде што менито за навигација и главната содржина стојат едно до друго на компјутер. Кога го отвора сајтот на мобилен, содржината се „стуткува“ и текстот не може да се прочита. Анализирај го овој проблем и кажи кој принцип на респонзивен дизајн е прекршен и како треба да се промени распоредот на мобилен.

8. Замисли дека треба да избереш стратегија за развој на нов веб-сајт за училиштето. Едната опција е да направиш две посебни верзии (една комплетна за компјутер и една посебна за мобилен со различен линк), а втората опција е да направиш една респонзивна страница со Media Queries. Оцени го ефектот од двете опции од аспект на одржување и промена на информациите во иднина.

9. Креирај едноставен CSS код за класа .kutija-blok која на компјутери ќе зафаќа точно 33.3% од ширината на екранот, но со помош на Media Query направи на мобилни уреди (под 600 пиксели) нејзината ширина автоматски да стане 100%.

10. Дизајнирај HTML структура и CSS код за едноставно заглавие (Header) на сајт кое содржи Наслов (лево) и Линкови (десно). На компјутер тие треба да стојат во иста линија (едно до друго) со помош на Flexbox, а кога ќе се стесни екранот под 480 пиксели, насловот и линковите треба да се центрираат и да се наредат еден под друг.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Кога правиме респонзивен дизајн, честопати процентите (%) не се доволни, затоа што зависат од големината на „родителската“ кутија. Затоа, професионалците користат мерни единици кои се врзани директно за големината на целиот екран (Viewport):

vw (Viewport Width): 1vw е еднакво на 1% од **ширината** на екранот. Ако ставиш width: 50vw;, елементот секогаш ќе зафаќа точно половина од екранот, без разлика на кој уред се отвора.

vh (Viewport Height): 1vh е еднакво на 1% од **висината** на екранот. Ова е одличен начин за правење на почетни („Hero“) секции кои треба да го исполнат целиот екран по висина кога ќе се вчита сајтот:

```
.pocetna-sekcija {
  height: 100vh; /* Ја зафаќа целата висина на екранот */
}
```

Еден од најголемите предизвици во респонзивниот дизајн е фонтот. Ако е насловот преголем на компјутер, на мобилен ќе се „скрши“ во три реда. Наместо да пишуваш десетици Media Queries само за фонтови, во модерната CSS постои функцијата clamp().

Таа прима три вредности: минимална, препорачана (флуидна) и максимална големина.

```
h1 {
  font-size: clamp(20px, 5vw, 45px);
}
```

Додека е Flexbox одличен за редување елементи во една насока (или само во ред, или само во колона), CSS Grid е дизајниран за комплетен дводимензионален распоред (и редови и колони одеднаш). Со Grid можеш да направиш распоред како во вистински весник или магазин.

```
.magazin-raspored {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr); /* Прави точно 3 еднакви колони */  
  grid-gap: 20px;  
}
```

А на мобилен, со само едно медиумско барање, едноставно го менуваш бројот на колони:

```
@media (max-width: 600px) {  
  .magazin-raspored {  
    grid-template-columns: 1fr; /* На мобилен сè се претвора во 1 колона */  
  }  
}
```

Понекогаш, некоја голема табела или комплицирана слика едноставно нема смисла да се прикажува на мобилен телефон бидејќи само ќе му го забави интернетот на корисникот и ќе го „нагмечи“ просторот. Со Media Queries можеш целосно да сокриеш одреден елемент на мобилен, а да го оставиш видлив на компјутер:

```
/* На мобилен, сокриј ја страницната лента со реклами */  
@media (max-width: 600px) {  
  .stranicna-reklama {  
    display: none;  
  }  
}
```

5.2.8. ГЕНЕРАТИВНА ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА ЗА ВЕБ-РАЗВОЈ

Технологијата на генеративна вештачка интелигенција (Generative AI) направи револуција во начинот на кој се креираат веб-страниците. Алатките како ChatGPT, Claude, Microsoft Copilot и специјализираните додатоци во VS Code (како GitHub Copilot) сега можат да пишуваат комплексен HTML и CSS код за неколку секунди.

Оваа содржина ќе ти помогне да разбереш како функционира овој процес, како правилно да му даваш наредби на компјутерот и како критички да го анализираш кодот што ќе го создаде вештачката интелигенција за тебе.

1. Поим за инструкција (prompt) и формулирање јасни инструкции

Еден од најважните концепти во работата со вештачка интелигенција е **prompt (промпт)**. Промпт е текстуална инструкција или наредба што корисникот ја внесува во ВИ алатката за да добие специфичен резултат (во нашиот случај, веб-код).

Ако ѝ кажеш на ВИ: „Направи ми веб-страница за училиште“, резултатот ќе биде премногу општ и едноставен. За да добиеш професионален код, треба да користиш структуриран и детален промпт.

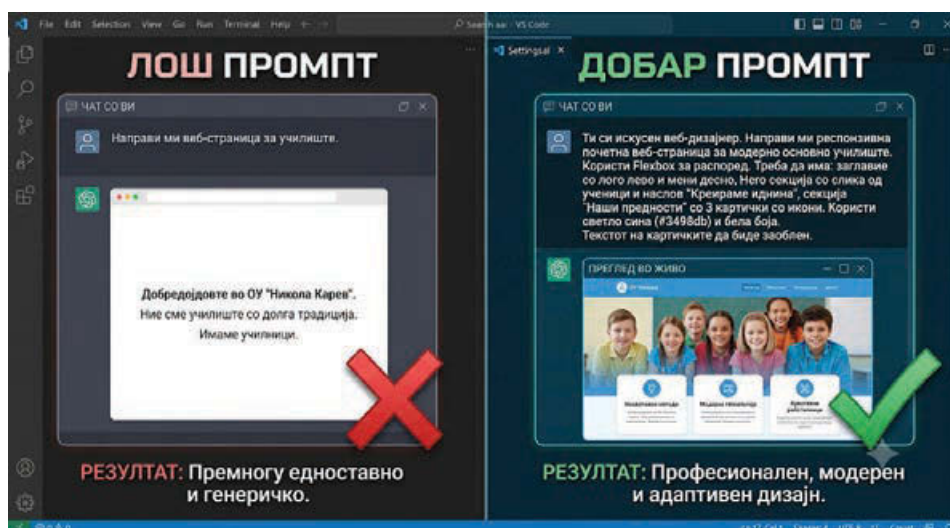
Како да формулираш јасно прецизен промпт за веб-развој?

Еден квалитетен промпт треба да ги содржи следните елементи:

- **Улога (Role):** Дефинирај ја улогата на ВИ (на пр. „Ти си искусен веб-дизајнер“).
- **Контекст/Цел:** Што точно е замислата за страницата.
- **Технички барања:** Кои јазици да ги користи (HTML, CSS со внатрешен/надворешен стил, Flexbox распоред).
- **Специфични детали:** Бои, број на секции, респонзивност за мобилни уреди.

Пример за лош промпт: „Напиши ми HTML за контакт веб-страница.“

Пример за одличен промпт: „Ти си професионален веб-програмер. Напиши ми комплетен HTML и внатрешен CSS код за адаптивна веб-страница «Контакт». Страницата треба да содржи форма со полиња за Име, Е-пошта и Порака, како и копче за испраќање. Користи модерен дизајн со темносива заднина (#2c3e50), бел текст и користи Flexbox за да ја центрираш формата на средина од екранот. На мобилни уреди (под 600px), формата треба да зафаќа 95% од ширината.“



Слика 14: Пример за општ и специфичен промпт

2. Автоматско креирање веб-страници и анализа на резултатите

Постапката за работа со вештачка интелигенција (ВИ) асистент при креирање веб-страници се одвива во неколку клучни чекори. За да добиеш квалитетен код, процесот не е само „копирај-залепи“, туку вклучува планирање, наредби и проверка.

Еве го целиот процес чекор по чекор:

- Планирање на структурата (пред да почнеш со пишување со код)

Пред воопшто да ја отвориш ВИ алатката, треба да имаш јасна претстава што сакаш да направиш. Скицирај го распоредот (layout), каде сакаш да стои главното мени, колку елементи-кутии со содржина ќе има и кои бои ќе ги користиш.

- Формулирање на јасен промпт (инструкција)

Ова е најважниот чекор. Наместо кратки и нејасни реченици, на ВИ асистентот треба да му формулираш детален опис.

- **Постави улога:** „Ти си искусен веб-програмер.“
- **Дефинирај ја целта:** „Направи ми веб-страница за...“
- **Кажи ги техничките правила:** „Користи HTML5 и внатрешен CSS во <style> ознака. Распоредот направи го со flexbox. Кодот треба да биде адаптивен (responsive) за мобилни уреди со помош на медиумски барања.“

- Генерирање и преземање на кодот

ВИ асистентот ќе го напише кодот за неколку секунди. Твоја задача е:

1. Да го прегледаш кодот уште во самиот чет-прозорец.
2. Да го копираш соодветниот блок со код.
3. Да го префрлиш во твојата нова датотека (на пример index.html) во VS Code и да го зачуваш.

- Анализа и тестирање во прелистувач (Проверка на резултатот)

Откако ќе го зачуваш фајлот, отвори го во прелистувач (Chrome, Edge). Сега правиш критичка анализа на сработеното:

- **Изглед:** Дали се боите и фонтовите онакви какви што побара?
- **Структура:** Дали се HTML ознаките правилно логички поставени (наслови, параграфи, кутии)?
- **Адаптивност (Responsive):** Намали го прозорецот за да видиш дали се префрлуваат правилно елементите еден под друг на помал екран.

- Итерација и рачна доработка (Поправање грешки)

Речиси никогаш првиот одговор од ВИ не е 100% совршен. Овде настапуваш ти како програмер:

- **Ако има мала грешка:** Поправи ја самиот рачно во VS Code (на пример, смени ја бојата од црвена во сина во CSS делот). Ова е најдобриот начин за учење.

- **Ако има поголема грешка или недостасува цела секција:** Врати се кај ВИ асистентот и дај му дополнителен промт, на пример: „Кодот е добар, но додај уште едно копче под текстот кое ќе биде зелено.“

Со овој кружен процес, вештачката интелигенција ти помага брзо да ја завршиш тешката и механичка работа, а ти да ја задржуваш целосната контрола врз логиката и финалниот изглед на веб-страницата.

За да станеш добар корисник на ВИ, треба да ги знаеш предностите и недостатоците на двата пристапи. Еве една директна споредба:

Карактеристика	Рачно напишан код (Hand-coded)	ВИ-генериран код (AI-generated)
Брзина на изработка	Бавно (бара рачно пишување на секоја линија и заграда).	Екстремно брзо (цели страници се готови за неколку секунди).
Контрола и разбирање	Максимална. Точно знаеш зошто секоја линија код е таму и лесно ја поправаш ако има грешка.	Лимитирана. Ако е кодот предолг, тешко е да разбереш каде настанал проблемот ако се расипе нешто.
Оптимизација и логика	Кодот е чист, напишан точно за твоите потреби, без непотребни повторувања.	Може да содржи „ѓубре-код“ (непотребни наредби кои ВИ ги додала без реална потреба).
Учење на материјалот	Најдобар начин за учење. Напредокот доаѓа преку правење и поправање сопствени грешки.	Не помага во учењето ако само слепо се копира и се лепи без читање.

Вештачката интелигенција е **извонреден асистент, но лош господар**. Најдобриот начин да ја користиш е да ја искористиш нејзината брзина за да ти направи брз скелет (шаблон) од страницата, а потоа ти рачно да го прегледаш, да го исчистиш и да го приспособиш кодот според твоите прецизни желби и правила!

- Анализа на постоечки HTML/CSS код според структура и изглед

Кога вештачката интелигенција ќе го генерира кодот врз основа на твојот промт, работата не е завршена. Како иден корисник на ВИ, треба да го анализираш резултатот. Вештачката интелигенција не е безгрешна – таа честопати генерира код кој изгледа убаво, но во заднината може да содржи сериозни пропусти.

На што да внимаваш при анализа на ВИ кодот?

- **Функционалност:** Дали функционираат сите линкови? Дозволува ли формата внесување текст?
- **Приспособеност (Адаптивност):** Движи го прелистувачот лево-десно. Додаде ли ВИ соодветни @media барања за уреди или кодот се искривува на мобилен?
- **Чистина на кодот:** Дали се класите јасно именувани (на пр. .karticka-proizvod) или се збунувачки?

За да знаеш дали ВИ завршила добра работа, треба да го „прочиташ“ кодот и да ја анализираш неговата **структура (HTML)** и **изгледот (CSS)**.

Да замислиме дека ВИ го генерирала следниов дел од код:

HTML

```
<div class="karticka">
  <h2 class="naslov">Добредојдовте</h2>
  <p>Прочитајте ги најновите вести тука.</p>
</div>
```

CSS

```
.karticka {  
    background: #fff;  
    padding: 20px;  
    border-radius: 5px;  
}  
  
.naslov {  
    color: red;  
}
```

Анализа на структурата (HTML): Кодот е логичен. Имаме контејнер правоаголник (div) кој внатре содржи еден наслов и еден параграф. Ова овозможува сите елементи да се движат и да се уредуваат заедно како една целина.

Анализа на изгледот (CSS): Кутијата има бела заднина, заоблени агли и внатрешно растојание (padding) за да не биде текстот залепен до рабовите. Насловот има специфична црвена боја. Структурата и изгледот се убаво разделени.



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО:

1. Кој поим се користи за текстуалната инструкција или наредба што корисникот ја внесува во ВИ алатката за да добие код?

- А) Скрипта (Script)
- Б) Промпт (Prompt)
- В) Таг (Tag)
- Г) Коментар (Comment)

2. Која е главната улога на медиумските барања (@media) кои ВИ асистентот ги генерира во CSS кодот?

- А) Да овозможат страницата да се приспособува на различни големини на екрани.
- Б) Да го забрзаат вчитувањето на сликите на сајтот.
- В) Да го поврзат веб-сајтот со социјалните мрежи.
- Г) Да пронајдат логички грешки во HTML структурата.

3. Зошто промптот „Направи ми веб-страница за патики“ се смета за нецелосен и лош промпт кога работиме со ВИ асистент?

4. Објасни што се случува со изгледот на една веб-страница на мобилен телефон ако ВИ асистентот генерирал код со фиксни ширини во пиксели (px) наместо проценти (%) или флуиден распоред.

5. Напиши детален и структуриран промпт кој од ВИ асистентот ќе побара да креира навигациско мени (Header) со три линкови, темнозелена заднина и кој користи Flexbox за подредување.

6. Ако во кодот генериран од ВИ асистент ја забележиш следнава линија: `<meta name="viewport" content="width=device-width, initial-scale=1.0">`, за кој уред ВИ асистентот се осигурал дека страницата ќе се прикаже правилно?

- А) Само за десктоп компјутери со големи монитори.
- Б) Само за принтери кога страницата се печати на хартија.

- В) За сите уреди, вклучувајќи ги и мобилните телефони.
- Г) Само за прелистувачи што не поддржуваат CSS.

7. Ученик го користи ВИ асистентот за да генерира форма за регистрација. Откако ќе го ископира кодот во VS Code, забележува дека копчето „Испрати“ стои најгоре на страницата наместо најдолу под полињата. Анализирај во кој дел од кодот (HTML или CSS) е направена логичката грешка во структурата и како би ја поправила.

8. Разгледај го следниов CSS код генериран од ВИ: `.tekst { color: blue; font-size: 16px; font-style: italic; }`. Ученикот барае кодот да нема закосени букви. Опиши која наредба од овој код треба да се избрише или да се промени за да се исполни барањето.

9. Оцени ги предностите и недостатоците при користење ВИ-генериран код во споредба со рачно напишан код од аспект на учење и разбирање на програмирањето кај почетници.

10. Замисли дека ВИ асистентот ти генерирал одличен распоред со две картички поставени една до друга на компјутер, но кога го тестираш кодот во прелистувачот, сфаќаш дека тој заборавил да додаде респонзивност за мобилни уреди. Креирај го (напиши го) CSS медиумското барање што недостасува за да ги спуштиш картичките со класа `.karticka` една под друга на екрани под 600 пиксели.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Вештачката интелигенција не го гледа само промптот што во моментот му го пишуваш, туку го памети и претходниот разговор. Тој простор за паметење се нарекува контекстен прозорец. Ако сакаш да знаеш повеќе, најдобриот начин да го искористиш ова е да ѝ дадеш на ВИ претходно дефинирани правила пред воопшто да побараш код.

На пример, првиот промпт во денот може да биде формулиран:

„Во текот на денешната сесија, јас ќе барам од тебе HTML и CSS код. Твоја задача е никогаш да не користиш застарени CSS наредби (како `float`), секогаш да користиш `Flexbox`, и коментарите во кодот секогаш да бидат на македонски јазик.“

Со ова, ти го тренираш асистентот да работи по твои стандарди уште пред да почне со креирање.

2.5.9. УРЕДУВАЊЕ И ПРИСПОСОБУВАЊЕ НА ВИ КОД

Да разгледаме еден пример за креирање веб-страница со ВИ алатка. Веќе споменавме дека VS Code има ВИ асистент. Отвори го VS Code и напиши ја следната инструкција:

Ти си професионален веб-дизајнер и кодер. Напиши ми комплетен, чист и лесен за разбирање HTML и внатрешен CSS код за веб-страница на училиштен клуб наречен „ИТ и Гејминг Клуб Код-Елит“.

Страницата треба да ги исполнува следните технички и дизајнерски стандарди:

1. Структура (HTML):

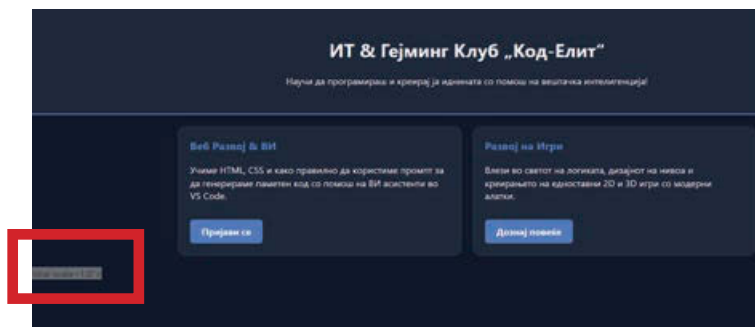
- Да има заглавие (header) со главен наслов и поднаслов кој го опишува клубот.
- Да има главен контејнер со две картички (кутии) за содржина: една за „Веб-развој и ВИ“ и една за „Развој на игри“.
- Секоја картичка треба да има наслов, краток текст и едно копче (линк) за акција.

2. Изглед и распоред (CSS):

- Користи модерна темна тема (темносива или темносина заднина) со многу светли букви за добар контраст и лесно читање.
- Главниот распоред на картичките направи го со Flexbox (display: flex), така што на компјутерски екран кутиите ќе стојат хоризонтално, една до друга, со еднаква ширина и убаво растојание меѓу нив.
- Картичките треба да имаат заоблени агли (border-radius) и дискретни рамки.

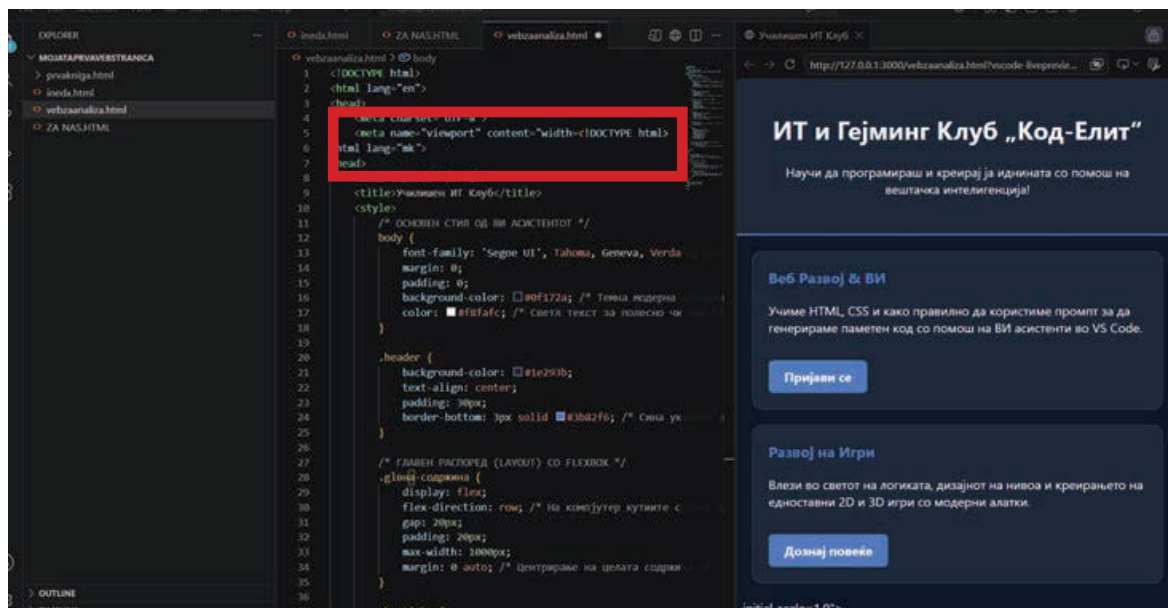
3. Адаптивност (Responsive):

- Вклучи медиумско барање (@media) за мобилни уреди со максимална ширина од 768 пиксели.
- Кога ќе се отвори страницата на мобилен, Flexbox распоредот треба да се смени во колона (column), така што картичките автоматски ќе се спуштат една под друга за да бидат прегледни на тесен екран.
- Генерираниот код зачувај го со наставка index.html во VS Code и отвори го во прелистувач за да го видиш резултатот. Забележувај дека една линија од кодот се појавува на веб-страницата, initial-scale=1.0">.

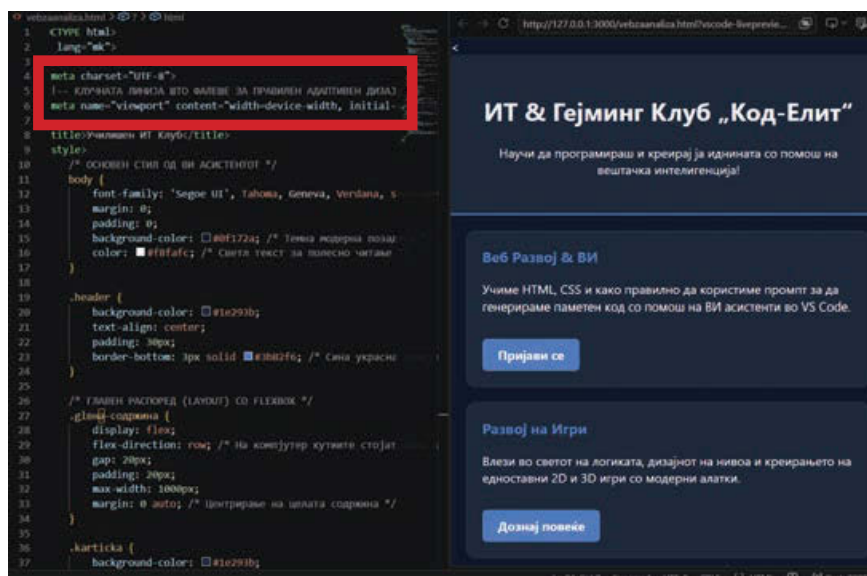


Тоа значи дека ВИ асистентот погрешил некаде. Анализирај за што служи и каде треба да се напише оваа наредба. Побарај од ВИ асистентот да го коригира кодот. Ако не успее, коригирај рачно под <head> линијата со наредбата

<meta name="viewport" content="width=device-width, initial-scale=1.0">



Зачувај ги промените, кликни F5 и веб-страницата ќе изгледа коректно.



Побарај нови измени на веб-страницата, на пример: „Во овој код сакам да ја сменам главната тема. Преправи го CSS-от така што заднината ќе биде темнозелена, а главните сини копчиња и линии ќе станат неонски жолти.“ или: „Анализирај ја структурата на кутиите во овој код и додај уште една, трета кутија со наслов 'Сајбер безбедност' која ќе го има истиот стил и ќе се вклопи во Flexbox распоредот.“ Може да се побара и проверка на адаптивноста на веб-страницата, „Погледни го медиумското барање на дното. Како можам да го сменам кодот за на мобилни уреди текстот во заглавието (header) да стане помал и да се центрира?“



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО:

1. Ученик/чка побарал/а од ВИ асистент да направи респонзивна страница. Во кодот што го добил, линијата за viewport е напишана со грешка: `<meta name="viewport" content="width=device-width", initial-scale=1.0">>` Што ќе се случи кога ќе се отвори оваа страница во прелистувач?

- А) Страницата воопшто нема да се вчита и ќе покаже празен екран.
- Б) Најдолу на страницата ќе се појави вишок текст како `initial-scale=1.0">>`.
- В) Целиот CSS стил ќе се избрише и текстот ќе стане црвен.
- Г) Формата за контакт автоматски ќе се сокрие.

2. Разгледај го овој CSS код кој треба да ги подреди картичките една до друга:

```
.grupacija-kutii {
  display: block;
  flex-direction: row;
  gap: 15px;
}
```

3. Која наредба е погрешна и спречува Flexbox распоредот да профункционира правилно?

- А) `gap: 15px;`
- Б) `flex-direction: row;`
- В) `display: block;`
- Г) Името на класата `.grupacija-kutii` има печатна грешка.

4. Погледни го следниов HTML код генериран од ВИ асистент:

```
<div class="karticka">
  <h3>Веб Развој</h3>
  <p>Учиме да кодираме со помош на ВИ.
</div>
```

Објасни која структурна грешка постои во овој дел од кодот и како може да влијае таа врз изгледот на останатите елементи под неа.

5. Во делот за медиумски барања, ВИ асистентот ја направил следнава грешка во синтаксата:

```
@media max-width: 768px {
  .glavna-sodrzina {
    flex-direction: column;
  }
}
```

Зошто ова медиумско барање не функционира во прелистувачот и кои знаци недостасуваат во првата линија за да биде точен кодот?

6. Ученик/чка сака да го приспособи ВИ кодот за да ја смени бојата на заднината на картичката. ВИ асистентот го напишал следниов код:

```
.karticka {
  background-color: #0f172a;
  padding: 20px;
  border-radius: 12px;
}
```

Кој синтаксички знак недостасува во овој код и кое CSS правило нема воопшто да се примени (ќе биде игнорирано од прелистувачот)?

7. Која наредба треба да ја избришеш или да ја промениш за да го исполниш барањето кодот да нема закосени букви (италик)?

```
.kopce {
  display: inline-block;
  background-color: #3b82f6;
  color: white;
  font-style: italic;
}
```

8. Разгледај го овој обид на ВИ асистентот да направи респонзивен распоред за мобилни телефони:

```
.karticka {
  width: 400px;
}
```

```
@media (max-width: 600px) {
  .karticka {
    width: 400px;
  }
}
```

Анализирај го кодот и објасни зошто овој дизајн нема да биде адаптивен на екрани со ширина од 360 пиксели (стандарден мобилен) и како треба да се уреди вредноста на width во медиумското барање.

9. Ученик/чка го користи следниов код за да креира наслов на страницата:

```
<h1 id="glaven-naslov" class="glaven-naslov">ИТ Клуб Код-Елит</h1>
```

Ако во CSS делот има две правила: едното таргетира со .glaven-naslov { color: red; }, а другото со #glaven-naslov { color: blue; }, со која боја ќе се прикаже насловот и зошто? Објасни го принципот на специфичност на селекторите.

10. ВИ асистентот генерирал код каде што две картички треба да бидат една до друга на компјутер, а една под друга на мобилен. Но, кодот изгледа вака:

```
.glavna-sodrzina {
  display: flex;
  flex-direction: column;
}
```

```
@media (max-width: 768px) {
  .glavna-sodrzina {
    flex-direction: column;
  }
}
```

Оцени ја функционалноста на овој код за двата уреди (компјутер и мобилен) и предложи ги точните вредности за flex-direction за да го поправиш распоредот.

11. Ова е HTML код за две картички во кој ВИ асистентот направил логичка грешка при копирањето на класите:

```
<div class="glavna-sodrzina">
  <div class="karticka">
    <h3>Веб Развој</h3>
    <p>Учиме кодирање со ВИ.</p>
  </div>
  <div class="glavna-sodrzina">
    <h3>Развој на Игри</h3>
    <p>Креираме 2D и 3D игри.</p>
  </div>
</div>
```

Креирај го точниот HTML код со тоа што ќе ја пронајдеш и ќе ја поправиш погрешната класа во втората кутија за да го добие таа истиот стил како првата картичка.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Пристапот „Прво мобилен“ (Mobile-First) преку ВИ

Повеќето почетници бараат од ВИ да направи страница за компјутер, па потоа ја поправаат за мобилен. Но, во модерната ИТ индустрија се користи пристапот Mobile-First. Тоа значи дека страницата прво се дизајнира за најмалиот екран, а потоа се раширува за големите монитори.

Кога бараш код од ВИ со овој пристап, инструкциите ги менуваш вака:

Бараш основниот CSS да биде напишан во една колона (за мобилен телефон).

Бараш медиумските барања да користат min-width (минимална ширина) наместо max-width.

На тој начин, со @media (min-width: 1024px), вештачката интелигенција ќе додаде правила кои ќе се активираат дури откако екранот ќе стане голем како на компјутер, што резултира со многу побрзо вчитување на страницата на телефоните.

Автоматско чистење на „AI губре-код“ (Refactoring)

Вештачката интелигенција има тенденција да пишува премногу код. Честопати за една иста работа ќе измисли три различни класи или ќе додаде CSS наредби кои прелистувачот и онака ги подразбира. Процесот на чистење и оптимизација на тој код се нарекува рефакторирање (Refactoring).

Откако ВИ ќе ти генерира код, напредна техника е да му го вратиш истиот тој код со промпт за оптимизација:

„Ова е кодот што го генерираше. Сега рефакторирај го: спој ги дуплираните CSS класи, отстрани ги непотребните маргини што се повторуваат и направи ја HTML структурата пократка и поедноставна за читање.“

Интеграција во VS Code со паметни додатоци

Напредните програмери не го копираат кодот од прелистувач (како ChatGPT), туку користат додатоци директно во самиот уредувач VS Code (како GitHub Copilot или Codeium).

Овие алатки имаат пристап до сите фајлови во твојата папка одеднаш. Тоа значи дека тие не само што генерираат нов код, туку можат да го анализираат твојот стил на пишување и автоматски да ти предложат следна линија код уште пред да почнеш да ја пишуваш, со притискање на само едно копче (Tab).

5.2.10 КРЕИРАЊЕ ВЕБ-ПРОЕКТ

Откако научи како работи вештачката интелигенција и како се прават приспособувања во кодот, време е да го погледнеме целиот процес на создавање на една веб-страница, од првата идеја на хартија па сè до финалниот изглед.

Професионалното креирање на веб-страница секогаш се одвива низ 4 клучни чекори кои мора да се следат по прецизен редослед.

1. Планирање (Што ќе правиме и за кого?)

Планирањето е темелот на веб-страницата. Пред да се напише макар една линија код, мора да се одговорат следниве прашања:

- **Која е целта на страницата?** (Дали е портфолио, училиштен сајт или онлајн продавница?)
- **Која е целната публика?** (Дали ја правиме за тинејџери, за наставници или за гејмери?)
- **Што сè ќе содржи?** Се прави листа на потребни делови: заглавие, картички со производи, галерија со слики, контакт форма.

2. Структура (Како ќе биде организиран кодот?)

- Структурата се однесува на скелетот на страницата, односно на логичкото подредување на HTML ознаките. Професионалните дизајнери во оваа фаза цртаат **жичена рамка (Wireframe)** – едноставна скица со правоаголници која покажува каде ќе стои секој елемент.

Во кодот, ова го претвораме во чист, семантички HTML5:

- Користиме `<header>` за менито најгоре.
- Користиме контејнери (`<div>` или `<section>`) со јасни класи за да ги групираме елементите.
- Внимаваме на хиерархијата на насловите (`<h1>` за главен наслов, `<h2>` и `<h3>` за поднаслови на картичките).

3. Дизајн (Како ќе изгледа страницата?)

Откако имаме цврста структура, преминуваме на CSS стилизирање. Дизајнот ги одредува визуелниот идентитет и корисничкото искуство.

Овде ги применуваме сите изучени CSS правила:

- **Распоред (Layout):** Го користиме Flexbox за елементите да се шират и да се подредуваат паметно.
- **Типографија и бои:** Избираме фонтови што лесно се читаат и палета од 2 до 3 усогласени бои (на пример, темна модерна тема со контрастни сини детали).
- **Простор:** Користиме маргини (margin) и внатрешни растојанија (padding) за да не бидат елементите згуснати и „залепени“ едни до други.
- **Адаптивност (Responsive):** Преку медиумските барања (@media) осигуруваме дека дизајнот ќе се приспособи од три колони на компјутер во една колона на мобилен.

4. Интеграција на содржините (Полнење на страницата со вистински податоци)

Интеграцијата е финалниот чекор каде што „лажниот“ или генеричкиот текст што ни го дал ВИ асистентот го заменуваме со вистински, реални информации:

- Го пишуваме финалниот текст во параграфите (`<p>`).

- Ги заменуваме привремените линкови со точни патеки до реални слики и надворешни страници.
- Ги поврзуваме копчињата со соодветните дејства.

Дури откако сите 4 чекори ќе бидат завршени по овој редослед, можеме да кажеме дека имаме комплетна, функционална и професионално изработена веб-страница.

За подобро да ги усвоиш фазите за подготовка, најдобро е да изработиш своја лична веб-страница (портфолио) каде што ќе ги прикажеш сите вежби, игри или веб-страници што си ги направил или планираш да ги правиш во иднина. Ова е нешто што професионалните програмери секогаш го имаат.

Архитектура и елементи што треба да ги содржи веб-страницата:

1. Структура и семантички HTML тагови

Страницата треба да биде поделена на јасни логички делови:

- `<header>`: Најгоре, со неговото име и краток поднаслов (на пр. „Иден веб-развојник“).
- `<main>`: Главниот дел, каде што ќе биде содржината.
- `<section>` со класа `.proekti`: Овде ќе стојат картичките за неговите проекти.
- `<footer>`: Најдолу, со текст за авторски права.

2. Распоред (Layout) со Flexbox

Во секцијата за проекти треба да има **3 картички** (на пример: „Училиштен сајт“, „Мојата прва игра“, „ВИ Вежба“).

- Овие 3 картички мора да бидат ставени во еден родителски контејнер кој ќе има `display: flex;` и `flex-direction: row;` за на компјутер да стојат убаво подредени во еден ред.

3. Адаптивен дизајн (@media барања)

Страницата мора да изгледа совршено и на мобилен телефон.

- Треба да се додаде `@media (max-width: 768px)` каде што Flexbox распоредот ќе се смени во `flex-direction: column;` за картичките да се спуштат една под друга.
- Во `<head>` делот задолжително да се стави точната линија за `viewport` за да нема вишок текст на екранот.

Како да се искористи ВИ асистентот за овој проект? (Процес на работа)

Учениците треба да го водат проектот преку следниве 3 чекори со ВИ во VS Code:

- **Прв промпт (Генерирање на скелетот):** Ученикот и пишува на ВИ: „Ти си мој ментор. Напиши ми HTML5 структура и почетен CSS за мое лично ИТ портфолио. Сакам темносива заднина, со заглавие, главен дел со 3 картички за проекти поставени со Flexbox и подножје. Вклучи и респонзивност за мобилен под 768px.“
- **Тестирање и пронаоѓање грешки (Анализа):** Ученикот го пушта кодот во прелистувач. Проверува дали ВИ заборавил некоја точка-запирка (;), дали текстот се искривува или дали случајно се појавил некој закосен текст (италик) кој не го сака.
- **Втор промпт (Приспособување и доработка):** Откако ќе го анализира кодот, ученикот бара измена: „Кодот е супер, но сега во втората картичка смени го насловот во ‘Мојот ВИ Асистент’“

Проект', а бојата на копчињата на сите картички направи ја да биде светлосина со заоблени агли.“

- **Персонален е:** Ученикот се претставува себеси и се чувствува како вистински програмер.
- **Ги содржи сите стандарди:** Бара размислување за структура (HTML), естетика и респонзивност (CSS), и правилна комуникација со технологијата (Промпт инженеринг).
- **Лесно се проверува:** Како наставник/оценувач, веднаш можете да видите дали Flexbox-от работи кога ќе го намалите прозорецот на прелистувачот.



ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО:

1. Ученик/чка сака да направи веб-страница за ИТ клубот, но пред да почне со кодирање, истражува кои информации им се најпотребни на соучениците и прави листа на секции. Во која фаза од изработката на проектот се наоѓа ученикот во тој момент?

- А) Дизајн
- Б) Структура
- В) Планирање
- Г) Интеграција на содржини

2. Кога велиме дека правиме „жичена рамка“ (Wireframe) или едноставна скица со правоаголници за да одредиме каде ќе стојат насловите, сликите и картичките, ние всушност ја дефинираме:

- А) Заднинската боја на сајтот
- Б) HTML структурата на страницата
- В) Брзината на вчитување на серверот
- Г) Врската со надворешните PDF документи

3. Разгледај ја следнава ситуација: Ученикот го завршил HTML кодот и сега сака да ги подреди картичките со проекти со помош на Flexbox, да ги заобли аглите на кутиите и да додаде медиумски барања за мобилен телефон. Објасни која фаза од изработката на проектот ја извршува ученикот во овој чекор и кој јазик го користи.

4. Што претставува фазата на „Интеграција на содржини“ и што точно треба да направи ученикот со линковите кои ВИ асистентот ги генерирал како href="#" за неговото портфолио да стане функционално?

5. Ученик го направил следното поврзување во својот код за да ја прикаже својата контакт форма: `<a href="C:/Users/Ucenik/Desktop/IT_Klub/kontakt.html" class="kopce">Отвори</a>`. Анализирај зошто е погрешно поставена оваа патека (апсолутна наместо локална) и како ќе влијае тоа ако некој друг се обиде да ја отвори страницата на својот компјутер.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ



Планирањето во веб-развојот вклучува и креирање на детална корисничка мапа (User Flow) која точно го предвидува секое движење и клик на посетителот низ страницата. Во оваа напредна фаза, професионалците користат специјализирани дигитални алатки како Figma или Adobe XD за да создадат интерактивен прототип кој изгледа и реагира како вистински сајт уште пред да се напише првата линија HTML код.

Дополнително, се планира и таканаречената SEO стратегија (оптимизација за пребарувачи), со која однапред се одредуваат клучните зборови во насловите и метаознаките за страницата подоцна да биде лесно пронајдена од страна на прелистувачите.

