Тодор Димов

ИНФОРМАТИКА

за I (прва) година гимназиско образование

Со Одлука за одобрување и употреба на учебник по наставниот предмет Информатика за I (прва)година во средно гимназиско образование бр. 26-420/3 од 07.04.2026 година, донесена од Националната комисија за учебници.

Скопје, 2026

Информатика

за I (прва) година гимназиско образование

Издавач:
ДПТУ „СТУДЕНТСКИ СЕРВИС“ ДОО - Скопје

Автор:
Тодор Димов

Рецензенти
Проф. д-р Газмен Џафери
Проф д-р Атанас Христов
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

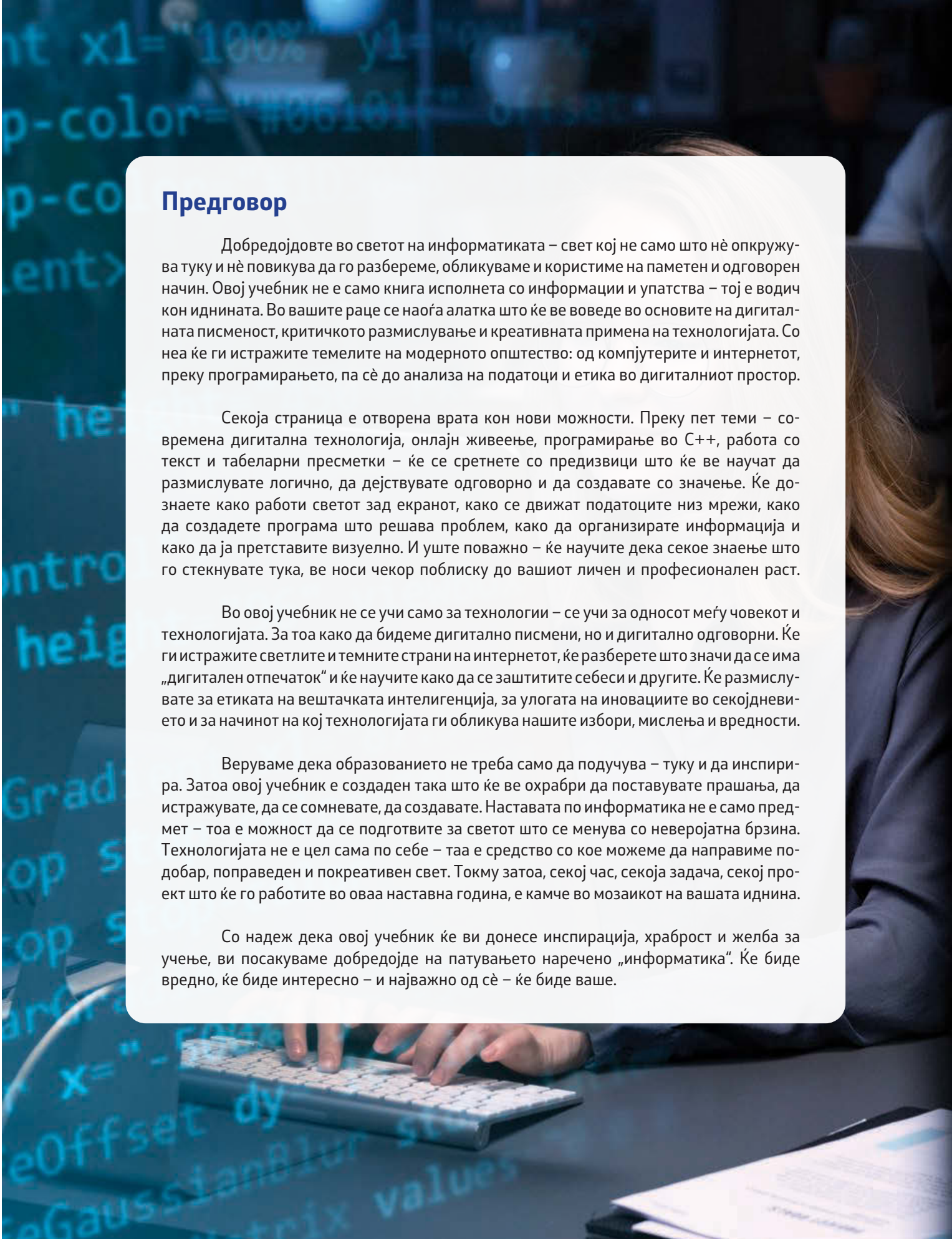
CIP - Каталогизација во публикација
Национална и универзитетска библиотека «Св. Климент Охридски», Скопје
004(075.3)

ДИМОВ, Тодор
Информатика за I (прва) година гимназиско образование / Тодор Димов.
- Скопје : Студентски сервис, 2026. - 263 стр. : илустр. ; 26 см

ISBN 978-608-4777-46-5

COBISS.MK-ID 69196549

СОДРЖИНА



Предговор

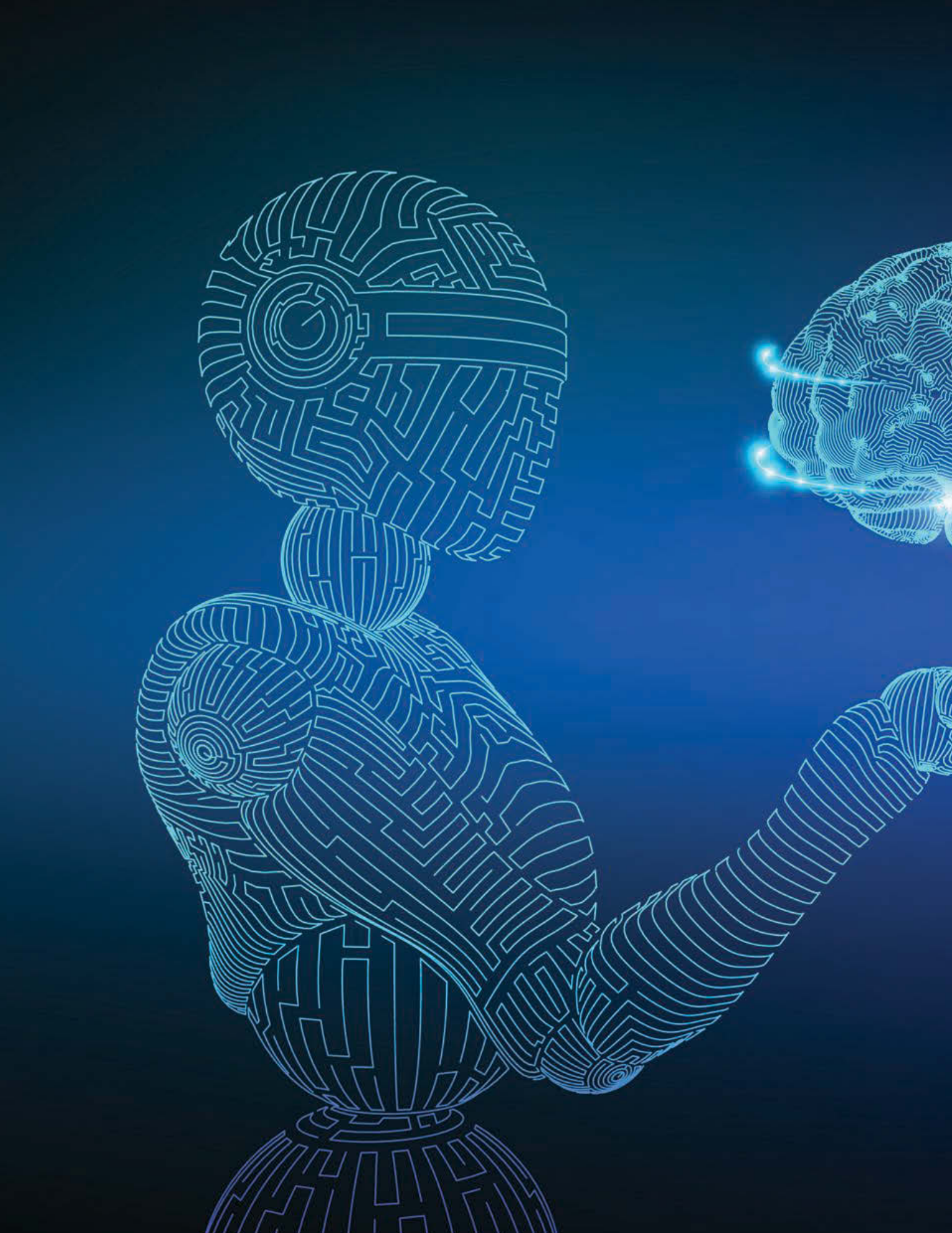
Добредојдовте во светот на информатиката – свет кој не само што нè опкружува туку и нè повикува да го разбереме, обликуваме и користиме на паметен и одговорен начин. Овој учебник не е само книга исполнета со информации и упатства – тој е водич кон иднината. Во вашите раце се наоѓа алатка што ќе ве воведе во основите на дигиталната писменост, критичкото размислување и креативната примена на технологијата. Со неа ќе ги истражите темелите на модерното општество: од компјутерите и интернетот, преку програмирањето, па сè до анализа на податоци и етика во дигиталниот простор.

Секоја страница е отворена врата кон нови можности. Преку пет теми – современа дигитална технологија, онлајн живеење, програмирање во C++, работа со текст и табеларни пресметки – ќе се сретнете со предизвици што ќе ве научат да размислувате логично, да дејствувате одговорно и да создавате со значење. Ќе дознаете како работи светот зад екранот, како се движат податоците низ мрежи, како да создадете програма што решава проблем, како да организирате информација и како да ја претставите визуелно. И уште поважно – ќе научите дека секое знаење што го стекнувате тука, ве носи чекор поблиску до вашиот личен и професионален раст.

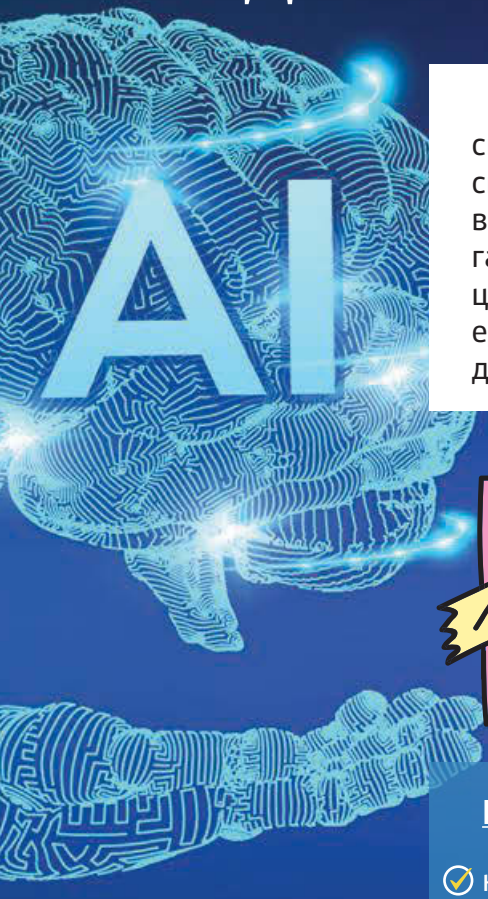
Во овој учебник не се учи само за технологии – се учи за односот меѓу човекот и технологијата. За тоа како да бидеме дигитално писмени, но и дигитално одговорни. Ќе ги истражите светлите и темните страни на интернетот, ќе разберете што значи да се има „дигитален отпечаток“ и ќе научите како да се заштитите себеси и другите. Ќе размислувате за етиката на вештачката интелигенција, за улогата на иновациите во секојдневието и за начинот на кој технологијата ги обликува нашите избори, мислења и вредности.

Веруваме дека образованието не треба само да подучува – туку и да инспирира. Затоа овој учебник е создаден така што ќе ве охрабри да поставувате прашања, да истражувате, да се сомневате, да создавате. Наставата по информатика не е само предмет – тоа е можност да се подготвите за светот што се менува со неверојатна брзина. Технологијата не е цел сама по себе – таа е средство со кое можеме да направиме подобар, поправеден и покреативен свет. Токму затоа, секој час, секоја задача, секој проект што ќе го работите во оваа наставна година, е камче во мозаикот на вашата иднина.

Со надеж дека овој учебник ќе ви донесе инспирација, храброст и желба за учење, ви посакуваме добредојде на патувањето наречено „информатика“. Ќе биде вредно, ќе биде интересно – и најважно од сè – ќе биде ваше.



ТЕМА 1: Современа дигитална технологија



Современата дигитална технологија опфаќа широк спектар на алатки и системи што користат електронски сигнали и податоци за обработка, складирање и пренесување информации кои имаат значење за луѓето и за организациите. Таа е сржта на комуникацијата, автоматизацијата и иновациите во различни аспекти од животот, од едноставни алатки како компјутери и паметни телефони до напредни апликации.

Нови поими

- Creative Commons лиценци,

општа вештачка интелигенција,

генеративна вештачка интелигенција.

Веќе знаеш да...

- набројуваш и опишуваш основни делови на компјутерски систем и да ги наведуваш нивните основни функции;
- ја објаснуваш улогата на меморијата и процесорите во компјутерот;
- набројуваш различни видови меморија;
- ги објаснуваш, со свои зборови, функциите на хардверските уреди;
- ги објаснуваш разликите помеѓу различните уреди, десктоп компјутер, лаптоп, таблет, смартфон;
- ги образложиш поимите вирус и антивирусна програма;
- ги почитуваш основните правила за етичко користење компјутер.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Истражи го твојот компјутер или лаптоп или користи слика ако немаш пристап до вистински уред! Потоа, креирај табела во програма за обработка на текст како прикажаната подолу и пополни ја следејќи го примерот во првиот ред!

Дел од компјутерот	Опис (со свои зборови)	Функција
CPU (процесор)	Тој ги извршува сите наредби и „мисли“ што треба да направи компјутерот.	Ги извршува сите инструкции од програмите и управува со работата на другите делови.
RAM (меморија)		
Тврд диск (HDD/SSD)		
…		

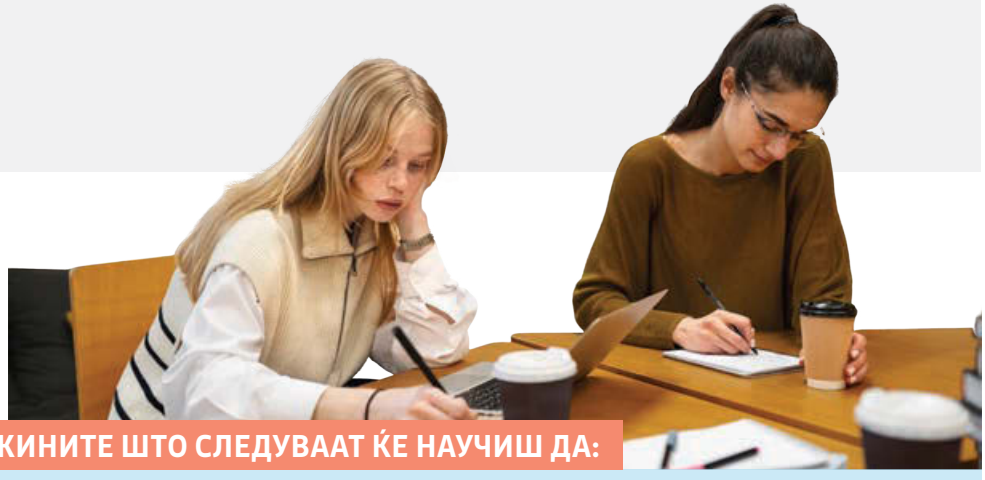
2. Искористи ја истата табела, додај уште две колони и измени ги насловите во колоните како примерот подолу за да направиш споредба меѓу различни уреди! Дополни ја табелата со други уреди кои ги знаеш!

Уред	Користење (каде?)	Преносливост Да/Не	Големина на екран	Батерија (работни часови)
Десктоп				
Лаптоп				
Таблет				
Смартфон				
…				

3. Направи постер во електронска форма на тема – „Како да се заштитиме од вируси“ со следниве елементи:

- Кратко објаснување што е компјутерски вирус.
- Три совети како да се заштитиш.
- Име на една антивирусна програма.

4. Напиши пет правила за етичко користење на компјутер и интернет! Можеш да ги напишеш како „Не треба да...“ или „Секогаш треба да...“

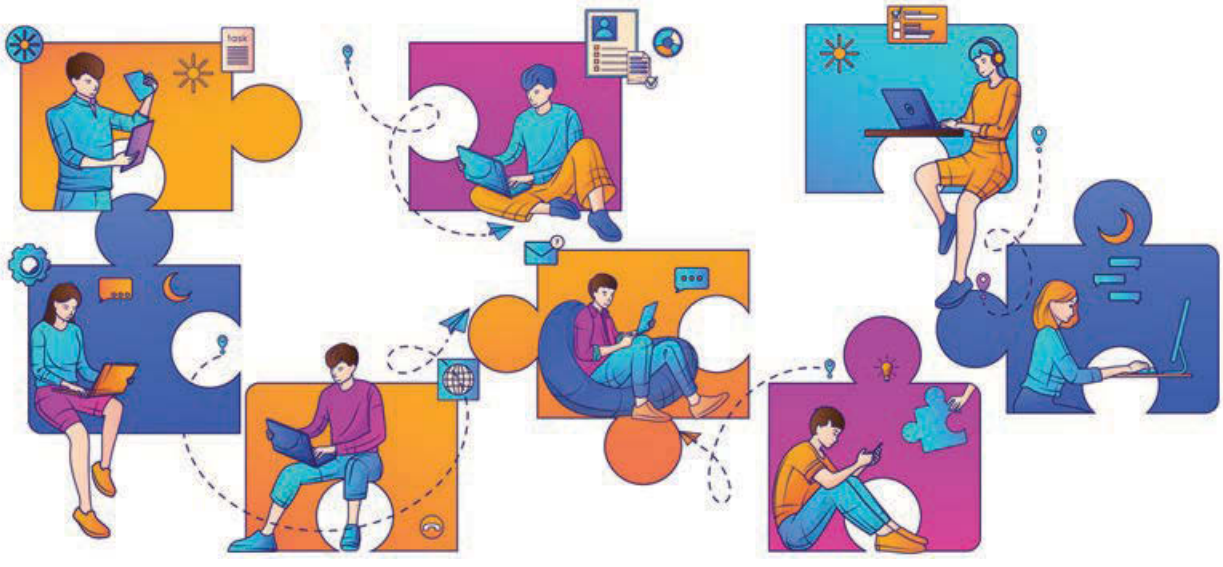


ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- идентификуваш модел на компјутерски систем во зависност од намената;
- ги наведуваш и да ги опишуваш современите дигитални технологии;
- ги објаснуваш значењето и улогата на дигиталните технологии во современото општество;
- користиш соодветен софтвер за различна намена;
- прилагодуваш ставки на оперативен систем;
- ја објаснуваш вештачката интелигенција преку нејзините карактеристики.

1.1 Улогата на дигиталната технологија во секојдневниот живот

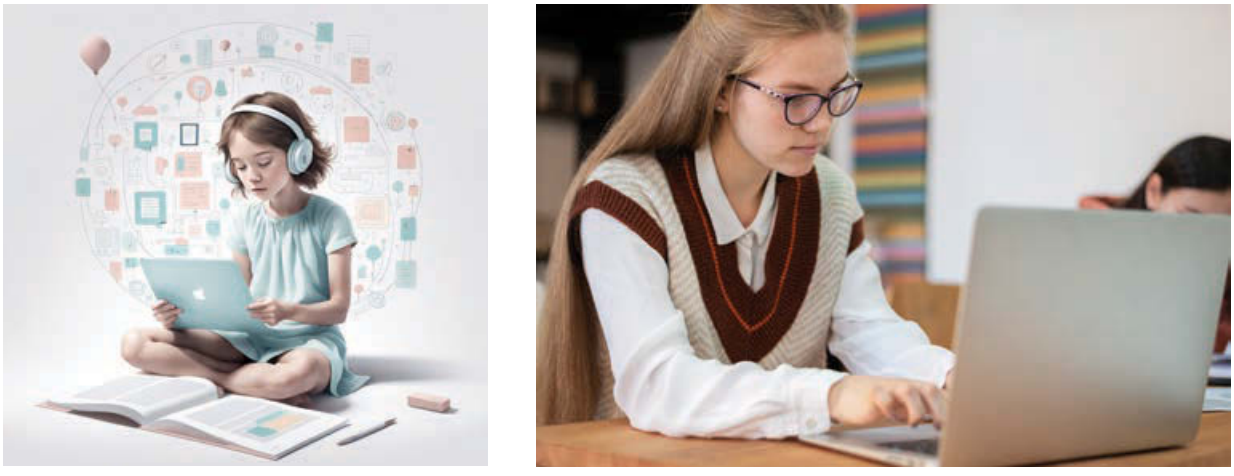
Во современото општество, дигиталната технологија е присутна во речиси секој аспект од животот. Таа овозможува автоматизација, поврзување и обработка на податоци на начини кои до неодамна биле незамисливи. Во домот, на работното место, во образованието и во здравството, дигиталната технологија обезбедува алатки што го олеснуваат секојдневието и го зголемуваат квалитетот на животот. Со нејзина помош, информациите се добиваат побрзо, комуникацијата се врши во реално време, а задачите што барале часови работа, сега се извршуваат за неколку секунди дури и милисекунди. Разбирањето на нејзината улога е суштински дел од дигиталната писменост и е предуслов за активно учество во модерното општество.



Слика 1 *Дигиталната технологија во секојдневниот живот*

Во домаќинствата, дигиталната технологија се користи преку уреди како паметни телефони, телевизори, гласовни асистенти, сензори за температура и системи за безбедност. Така, со помош на апликации, корисниците можат да ги контролираат осветлувањето, греењето или домашната забава. Паметните уреди не само што го зголемуваат комфорот, туку и штедат енергија, време и ресурси преку автоматизирани сценарија и управување на далечина.

Во образованието, учениците и наставниците користат платформи за е-учење, дигитални табли, симулации и алатки за споделување образовни содржини. Дигиталната технологија го проширува пристапот до знаење, овозможува учење од дома, а мултимедијалните содржини ја прават наставата подинамична и интерактивна, така што учениците се повеќе ангажирани и фокусирани на учењето. Преку дигитални дневници, родителите можат да го следат напредокот во учењето на своите деца, а со помош на видеоконференции, се одржуваат состаноци и консултации без физичко присуство.



Слика 2 *Дигиталната технологија во образованието*

Во здравството, медицинските установи користат електронски досиеја на пациенти, дигитални дијагностички уреди и телеконсултации. Пациентите закажуваат термини онлајн, добиваат резултати преку интернет и користат мобилни апликации за следење на здравствената состојба. Вештачката интелигенција се применува во анализа на медицински податоци, додека роботизираните системи се користат во хирургијата или во негата.

Речиси на секое работно место дигиталните алатки овозможуваат комуникација, тимска соработка, организација на задачи и автоматизација на процеси. Работата од далечина, споделувањето документи преку облак и видеоконференциските системи сè почесто се користат во институциите и во компаниите. Дигитализацијата овозможува зголемена ефикасност, но бара и нови вештини од работниците, како што се кибербезбедност, анализа на податоци и дигитална комуникација.

Во транспортот, дигиталната технологија се користи за навигација, управување со сообраќајот, системи за следење на возила и автоматизирани плаќања. Паметните апликации за јавен превоз овозможуваат следење на возниот ред во реално време, додека навигациските системи ја оптимизираат маршрутата на движење и го намалуваат времето на патување. Автомобилите со автономно управување, кои користат сензори, камери и алгоритми за ВИ, веќе се тестираат на патиштата и претставуваат иднина на мобилноста.

Во трговијата, купувањето и плаќањето преку интернет се во постојан раст. Електронската трговија, мобилното банкарство и дигиталните паричници овозможуваат трансакции без готовина, од секое место и во секое време. Преку анализа на податоците, продавниците добиваат увид во навиките на потрошувачите и понудата ја прилагодуваат според нивните интереси. Истовремено, алгоритмите за препораки, кои се базираат на вештачка интелигенција, влијаат врз тоа што и како купува клиентот, отворајќи нови прашања за етика и приватност.

Во јавната администрација, услугите сè почесто се одвиваат во дигитална форма. Електронското закажување, онлајн плаќањето на сметки и пристапот до документи преку интернет овозможуваат побрза и поефикасна комуникација со институциите. Дигиталната технологија ја намалува потребата за физичко присуство, ги скратува бирократските процедури и го зголемува транспарентниот пристап до информации. Сепак, овие промени бараат развој на дигитална писменост кај граѓаните и воведување безбедносни мерки за заштита на личните податоци.

Негативните влијанија од дигиталната технологија исто така се присутни и не смеат да бидат занемарени. Прекумерната употреба на екрани доведува до зависност, и предизвикува нарушувања во спиењето и социјална изолација. Приватноста и безбедноста се изложени на ризик поради злоупотреба на лични податоци и кибернапади. Дополнително, постои опасност од дигитална нееднаквост, при што оние без пристап до технологија остануваат во информациска и образовна празнина. Од тие причини, дигиталната технологија мора да се користи одговорно и со критичка свест.

Дигиталната технологија станува составен дел од секојдневниот живот, овозможувајќи побрза комуникација, попаметно управување со ресурсите, подобар пристап до услуги и нови форми на учење и работа. Со нејзиното користење се добиваат значајни придобивки, но истовремено се јавуваат и нови предизвици поврзани со етиката, безбедноста и здравјето. Разбирањето на овие аспекти овозможува технологијата да се користи свесно, одговорно и ефективно, во служба на човекот и општеството.

ПРИМЕР 1

Е-учење во практиката

Во едно училиште, наставникот користи дигитална платформа за учење преку која учениците добиваат материјали, решаваат тестови и добиваат повратни информации во реално време.

Оваа платформа овозможува и родителите да го следат напредокот на своето дете, а наставникот добива автоматизирани извештаи за резултатите.

ПРИМЕР 2

Паметен дом

Во паметниот дом, преку мобилна апликација се контролираат осветлувањето, температурата и безбедносниот систем. Кога никој не е дома, системот автоматски го намалува греењето, активира аларм и испраќа известувања при необични движења.

ПРИМЕР 3

Телемедицина

Пациент со хронична болест користи апликација што го мери крвниот притисок и го испраќа резултатот директно до матичниот лекар. При промени над дозволените граници, системот автоматски го алармира лекарот и закажува онлајн консултација.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Објасни три начини на кои се користи дигиталната технологија во домаќинството!

2. Наведи примери за употреба на дигитална технологија во образованието!

3. Објасни како помага дигиталната технологија во здравството!

4. Што е телемедицина? Објасни со пример!

5. Напиши позитивни и негативни ефекти од користењето мобилни телефони!

6. Кои опасности се јавуваат од прекумерна зависност од дигитални уреди?

7. Што се подразбира под дигитална нееднаквост? Како може да се намали?

8. Истражи како се користи дигиталната технологија во јавната администрација во нашата држава!

ПРАКТИЧНА ЗАДАЧА

Замисли дека си ИТ администратор во едно училиште. Дадена ти е задача да избереш соодветен компјутерски систем за секоја од следниве ситуации. За секој случај, објасни зошто го избра тој систем.

Сценарио 1: Канцелариска работа во училишната администрација

▸ Ти треба систем за обработка на документи, е-пошта и управување со податоци за учениците.

Сценарио 2: Видеомонтажа и графички дизајн за училишниот мултимедијален клуб

▸ Ти треба систем кој ќе може да обработува мултимедија со висока резолуција.

Сценарио 3: Училница за програмирање и учење основи на вештачката интелигенција

▸ Ти треба систем кој поддржува развој на софтвер, работа со големи податоци и модели за машинско учење.

Сценарио 4: Информациски киосk за посетители на училиштето

▸ Ти треба систем кој работи постојано, едноставен е за користење и нема потреба од тастатура или глушец.

За секое сценарио, избери еден од следниве типови компјутерски системи:

Десктоп компјутер, лаптоп, сервер, работна станица, вграден систем (embedded system), таблет/интерактивен екран.

Објасни го својот избор за секое сценарио со две до три реченици.

ЗАКЛУЧОК

Дигиталната технологија претставува неизбежен дел од секојдневниот живот и значително го обликува начинот на кој учиме, работиме и комуницираме. Таа овозможува побрз пристап до информации, полесна комуникација и поефикасно извршување на различни активности. Сепак, покрај бројните предности, како што се зголемената продуктивност, креативност и олеснето учење, постојат и одредени недостатоци. Тие најчесто се поврзани со ризиците за приватноста, можното прекумерно користење, изложеноста на дезинформации и зависноста од технологија. Затоа, важно е да развиеш критичко разбирање за улогата на дигиталната технологија, нејзините придобивки и потенцијалните опасности. Таквата свест им помага на луѓето да ја користат технологијата на одговорен, безбеден и свесен начин, со што ќе ги зајакнуваат своите дигитални вештини и компетенции за живот и работа во современото дигитално општество.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во современите дигитални технологии спаѓаат и виртуелната реалност (VR, Virtual Reality) и проширената реалност (AR, Augmented Reality) со кои се создаваат интерактивни дигитални искуства, но функционираат на различни начини. Имаат цел целосно да го „заменат“ реалниот свет со компјутерски генерирана околина.

Виртуелната реалност (VR) е симулација на различни активности и за неа се потребни:

- **VR очила/шлем (Headset)** – прикажува 3D-слика пред очите на корисникот.
- **Сензори за движење** – ги следат движењата на главата и на телото.
- **Контролери** – кои овозможуваат интеракција со виртуелниот свет.
- **Аудиосистем** – кој обезбедува 3D-звук за реално искуство.

Функционира на тој начин што кога ќе ги ставиш очилата, гледаш дигитален свет што реагира на твоите движења. Секое вртење на главата или движење на рацете се следи и се прикажува во реално време. Програмите користат графички мотори (како Unity или Unreal Engine) за да создадат реалистични 3D-средини.

Проширената реалност (Augmented Reality) го надградува реалниот свет со дигитални елементи (слики, текст, 3D-објекти). Користи информации во реално време како што се текст, слика, звук или други виртуелни елементи интегрирани во реалниот свет и по тоа се разликува од виртуелната реалност. AR интегрира и додава вредност на интеракцијата на корисникот со реалниот свет наспроти симулациите. Потребни се камера и екран – најчесто на мобилен телефон, таблет или AR очила, сензори и GPS – за препознавање на околината и позицијата и софтвер за препознавање на објекти/простор – за да ги „залепи“ дигиталните елементи врз реалниот свет. Функционира така што камерата го снима реалниот свет, софтверот го анализира просторот и додава дигитални објекти (на пример, 3D-модел на нова работна маса во твојата соба). Корисникот гледа на екранот комбинација од реална и дигитална слика.

Истражи како се користи дигиталната технологија во „паметни градови“. Кои технологии се применуваат за управување со сообраќајот, јавната безбедност, отпадот и енергијата? Проучи како големите светски градови користат вештачка интелигенција за подобрување на урбаното живеење. Како може VR и AR да помогнат за подобрување на квалитетот на животот во градот? Дали може да најдат примена и во рурални средини?

1.2 Компјутерски систем и неговите компоненти

Како се поврзува дигиталната технологија со компјутерскиот систем?

Дигиталната технологија е зависна од компјутерските системи. За функционирање на ВИ, за дигиталното учење, автоматизацијата и анализата на податоци, потребни се компјутерски системи кои ја извршуваат ВИ технологијата. На пример, ВИ во образованието работи преку компјутерски систем кој користи алгоритми за анализа на резултати, препознавање обрасци и прилагодување на наставата според ученикот. Дигиталната технологија ги проширува функциите на компјутерските системи. Преку ВИ и машинското учење, компјутерските системи добиваат нова функционалност: не само обработка, туку и самостојно учење, препознавање, автоматско одлучување (на пример: прифаќање на кандидат во образовна установа). Компјутерските системи се платформа за дигитална трансформација. Едукативните системи користат компјутерски системи како инфраструктура за дигитална технологија – за администрирање, следење напредок, персонализирана настава, итн.

Компјутерскиот систем се состои од хардвер и софтвер. Хардверот ги вклучува физичките компоненти како куќиште, монитор, тастатура, глувче, тврд диск, слушалки, микрофон, печатач и проектор. Софтверот е збир на програми кои го контролираат хардверот и овозможуваат извршување задачи. Влезните уреди (како тастатура и глувче) овозможуваат внес на податоци, додека излезните уреди (како монитор и печатач) прикажуваат резултати. Сите извршуваат различна функција и функционираат на различен начин.

Во куќиштето е сместена централната процесорска единица **CPU** (Central Processing Unit) која извршува инструкции и управува со сите процеси во системот преку своите компоненти. Работи на тој начин што ги чита, ги обработува и ги извршува инструкциите од програмите, прави пресметки и донесува одлуки. Составена е од:

- **ALU (Аритметичко-логичка единица):** која врши математички и логички операции.
- **CU (Контролна единица):** која управува со протокот на податоци и инструкции.

Податоците што ги обработува централната процесорска единица, CPU, привремено се сместуваат во работната меморија, наречена **RAM** (Random Access Memory). Таа функционира така што привремено ги складира податоците и програмите што се во тековна употреба. Работи на следниов начин: кога отвораш апликација, таа се вчитува во RAM за побрз пристап до неа. Сè се брише од RAM кога ќе се исклучи компјутерот.

Податоците трајно се сместуваат и се зачувуваат во трајна меморија (HDD/SSD). Во неа долгорочно се складираат различни видови податоци (документи, програми, слики). Постои разлика меѓу HDD/SSD, HDD (Hard Disk Drive) е механички диск, побавен, но поевтин, а SSD (Solid State Drive) е побрз, без подвижни делови, поскап.

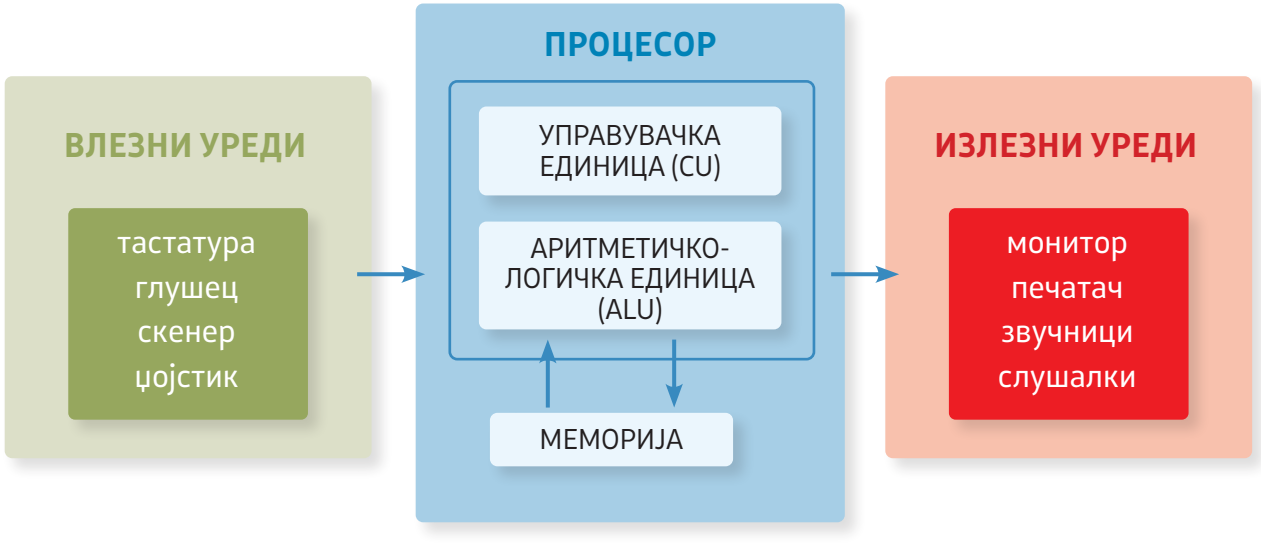
Ова претставува основа на компјутерската архитектура која е предложена од Фон Нојман уште во 1945 година и опишува како функционира еден компјутер.

1.3 Фон-Нојмановата архитектура и компоненти на компјутерот

Фон Нојмановата архитектура претставува основа врз која се засноваат модерните компјутерски системи. Овој концепт е предложен во средината на 20 век од страна на унгарско-американскиот математичар Џон фон Нојман. Според овој модел, компјутерот се дефинира како уред што ги извршува инструкциите однапред запишани во својата меморија. Фон Нојмановата структура го одредува начинот на кој функционираат основните делови на компјутерот – процесорот, меморијата, уредите за влез и излез – и како тие меѓусебно соработуваат преку комуникациски канали. Иако технологијата напреднала, основните принципи дефинирани со овој модел сè уште се применуваат во современите компјутери.

Фон Нојмановата архитектура се базира на неколку клучни компоненти: централната процесорска единица (CPU), главната меморија, уредите за влез и излез и комуникациските патеки наречени магистрали (bus). Процесорот ја контролира работата на целиот систем и се состои од две главни поединици – аритметичко-логичката единица (ALU) и контролната единица (CU). Аритметичко-логичката единица извршува пресметки и логички операции, додека контролната единица ги интерпретира инструкциите и ги координира сите активности во системот.

Главната меморија служи за чување податоци и инструкции што се моментално потребни за извршување. Оваа меморија е привремена (волатилна), што значи дека нејзината содржина се губи по исклучување на компјутерот. Во неа се чуваат и програмите што се извршуваат, како и резултатите од пресметките. Меморијата се организира во адреси и секој податок се лоцира со помош на уникатна адреса.

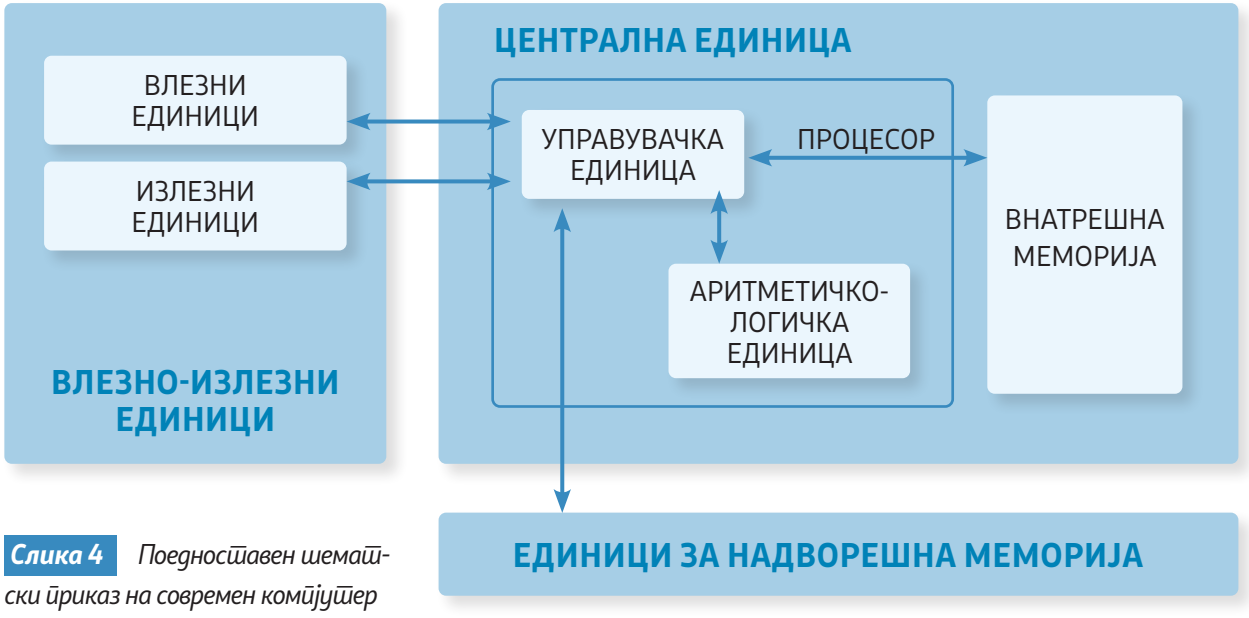


Слика 3 *Поедноставен шематски приказ на Фон-Нојмановиот модел на компјутерски систем*

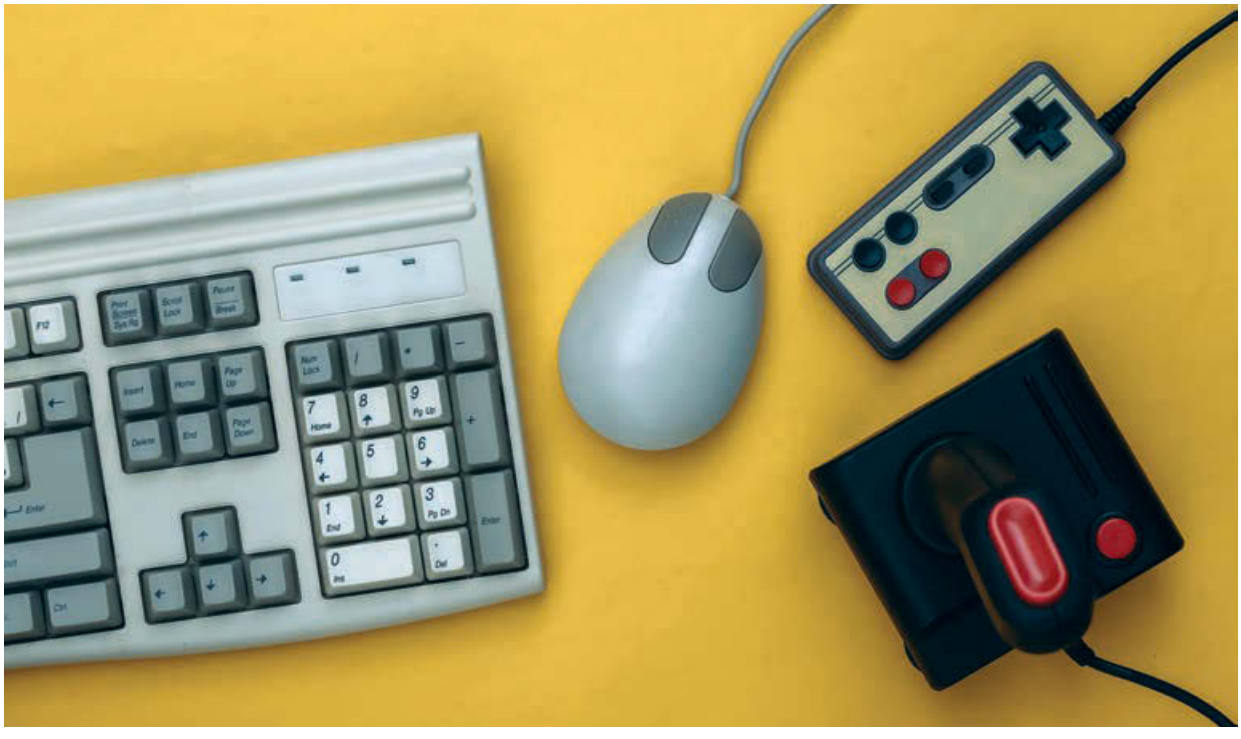
Еден од клучните принципи на Фон Најмановата архитектура е тоа што и податоците и инструкциите се чуваат во истата меморија. Овој пристап овозможува поголема флексибилност, бидејќи инструкциите може да се модифицираат при извршување, но истовремено претставува и потенцијален безбедносен ризик ако нема јасна разграниченост меѓу код и податоци. Токму овој модел го направил возможно програмирањето, бидејќи програмите може да бидат запишани, пренесени, чувани и обработени како кој било друг податок. Но, инструкциите се преземаат и извршуваат една по една, што создава тесно грло каде што процесорот не може да презема инструкции и податоци истовремено. Ова е познато како „тесно грло“ на Фон Најман. За разрешување на овој недостаток, современите процесори користат оперативна внатрешна меморија, односно техника за зголемување на перформансите на процесорот со разделување на извршувањето на инструкциите во секвенцијални фази (како преземање, декодирање, извршување, пристап до меморија, запишување) и обработка на различни инструкции истовремено. На тој начин се овозможува повеќе инструкции да се извршуваат во различни фази истовремено, со што значително се зголемува вкупниот проток во компјутерот.

Влезните уреди овозможуваат внесување информации во компјутерот – како тастатура, глушец, микрофон, скенер. Излезните уреди служат за прикажување или пренос на резултатите од обработката – како монитор, печатач, звучник. Постојат и комбинирани уреди што истовремено служат и како влез и како излез, на пример, екрани на допир и мрежни адаптери.

Магистралите се електронски патеки што ги поврзуваат сите компоненти на системот. Постојат три основни типа: адресна магистрала (за адресирање мемориски локации), податочна магистрала (за пренос на податоци) и контролна магистрала (за пренос на сигнали за контрола). Овие магистрали овозможуваат брза и синхронизирана комуникација меѓу процесорот, меморијата и периферните уреди, со што се одржува функционалноста на системот.



Слика 4 *Поедноставен шематски приказ на современ компјутер*



Слика 5 *Влезни уреди*

Централната процесорска единица (CPU) функционира како на компјутерот и нејзината брзина во голема мера го одредува вкупниот перформанс на системот. Современите процесори содржат милијарди транзистори и користат повеќе јадра (cores), што овозможува паралелна обработка на информации. Сепак, основниот логички принцип на редоследно извршување на инструкциите, како што го дефинирал Фон Најман, останува валиден и денес.

Главната меморија се мери во гигабајти (GB), а за најбрза обработка се користи и кеш-меморија која се наоѓа во самите процесори. Податоците патуваат помеѓу процесорот и меморијата со огромна брзина, но токму оваа брзина претставува едно од главните ограничувања во модерните компјутери – познато како „Фон Најманово тесно грло“ (Von Neumann bottleneck). Овој термин се користи за да се опише ограничената пропусна моќ на комуникација меѓу меморијата и процесорот, што ја забавува вкупната брзина на системот.

Фон Најмановата архитектура го поставува темелот на сите модерни дигитални компјутери. Иако технологијата драматично напредува, основните принципи остануваат непроменети: инструкциите и податоците се чуваат во заедничка меморија, процесорот ги извршува чекор по чекор, а сите компоненти соработуваат преку комуникациски магистрали. Разбирањето на оваа архитектура е основа за секое понатамошно учење во информатиката и овозможува подобро разбирање на начинот на кој функционираат компјутерите.

ЗАКЛУЧОК

Разбирањето на начинот на кој функционира компјутерскиот систем според архитектурата на Фон Нојман ти овозможува да сфатиш како компјутерот ги прима, обработува и зачувува податоците. Со опишување на улогата на процесорот, меморијата и уредите за внес и изнесување, развиваш основна, но суштинска дигитална писменост.

Во исто време, споредувањето на различни компјутерски системи и правилното категоризирање на влезните и излезните уреди ти помага подобро да разбереш како соработуваат различните делови на компјутерот. На тој начин се оспособуваш да избереш соодветна опрема, да препознаваш разлики меѓу уредите и да ја применуваш технологијата поефикасно во секојдневните задачи. Овие знаења не само што ќе те подготвуваат за понатамошно учење во областа на информатиката туку и ќе те охрабрат да станеш свесен и компетентен корисник на современата технологија.

ПРИМЕР 1

Како процесорот ја извршува инструкцијата?

Кога се врши операција на собирање два броја, инструкцијата се чита од меморијата, се интерпретира од контролниот дел на процесорот, а потоа се извршува во аритметичко-логичката единица (ALU) која го пресметува резултатот.

ПРИМЕР 2

Заемна соработка на уреди преку магистрали

Кога корисникот притиснува копче на тастатурата, сигналот оди преку влезен уред, се пренесува преку магистрала до процесорот, каде што се обработува, по што резултатот може да се испрати до излезен уред – на пример, да се прикаже на екран.

ПРИМЕР 3

Зошто постои кеш-меморија?

Процесорот работи побрзо од меморијата и за да не чека, се користи кеш-меморија која ги чува најчесто користените податоци. Таа е побрза, но многу помала и значително ја подобрува ефикасноста на извршувањето.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Опиши ја улогата на CPU во еден компјутерски систем!

2. Што е ALU и која е нејзината функција?

3. Објасни ја разликата помеѓу RAM и кеш-меморија!

4. Кои се трите видови магистрали и за што служат?

5. Напиши чекор по чекор како се извршува една инструкција според Фон Нојмановата архитектура!

6. Истражи што претставува терминот Фон Нојманово „тесно грло“!

7. Зошто е важно податоците и инструкциите да се чуваат во меморија?

8. Направи споредба меѓу влезен и излезен уред со примери од секојдневието!

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи дали постојат алтернативи на Фон Нојмановата архитектура и зошто некои истражувачи предлагаат нови модели! Проучи што претставува „паралелно процесирање“ и зошто се користат повеќе јадра во модерните процесори!

Каква е улогата на графичката картичка. **GPU – Graphics Processing Unit**? Што е матична плоча (Motherboard)? Како добиваат податоци овие уреди?

1.4 Влезни и излезни уреди – класификација и споредба

Во секој компјутерски систем, влезните и излезните уреди играат клучна улога во размената на информации помеѓу корисникот и машината. Преку влезните уреди, податоците се внесуваат во системот, додека излезните уреди ги прикажуваат резултатите од обработката. Овие уреди овозможуваат интеракција со компјутерот и се неопходни за извршување на секојдневни задачи – од работа со текст и слики до комуникација и игри. За правилно разбирање на нивната улога, потребно е тие да се класифицираат според функцијата, типот на податоци што ги обработуваат и нивната примена во различни области.

Влезните уреди служат за внесување податоци во компјутерот. Најчесто користени влезни уреди се тастатурата, глувчето, скенерот, микрофонот и камерата. Тастатурата овозможува внес на алфанумерички податоци, глувчето овозможува покажување и селектирање, додека микрофонот внесува аудиосигнал кој понатаму се обработува. Скенерите овозможуваат внесување слики или документи, а дигиталните камери ги претвораат визуелните информации во податоци што компјутерот може да ги обработи.

Слика 6 *Влезни и излезни уреди*

Излезните уреди се користат за прикажување резултати од обработката. Меѓу најпознатите се мониторите, печатачите, звучниците и проекторите. Мониторите ги прикажуваат податоците визуелно на екран, печатачите ги претвораат дигиталните информации во физички копии, а звучниците овозможуваат аудиоизлез. Секој излезен уред е дизајниран за специфичен вид информации и нивниот квалитет, брзината и прецизноста се приспособуваат според потребите на корисниците и апликациите.

Некои уреди можат истовремено да функционираат како влезни и како излезни. Такви уреди се нарекуваат влезно-излезни уреди. На пример, екранот на допир овозможува истовремено внесување податоци преку допир и прикажување на информациите. Модемите и мрежните картички се користат за размена на податоци преку компјутерски мрежи и исто така се сметаат за I/O уреди.

Слика 7 *Влезно-излезни уреди*

Класификацијата на влезните и излезните уреди може да се направи според различни критериуми: според природата на податоците (текст, слика, звук), според начинот на користење (рачен, автоматски), според брзината на внес/излез и според точноста. Во индустриски и во медицински апликации, од уредите се бара голема прецизност и робусност, додека во домашни услови најважни се едноставноста и удобноста при користењето.

Современите влезни уреди користат напредни технологии за зголемена интерактивност и прецизност. На пример, графичките таблети користат дигитални пенкала чувствителни на притисок за уметничка работа, додека биометриските уреди како скенери за отпечатоци или системи за препознавање лице овозможуваат безбедна идентификација. Во виртуелната реалност, влезните уреди како ракавици и шлемови собираат податоци за движења и гестикулации кои потоа се користат за навигација и интеракција во дигиталниот свет.

Кај излезните уреди, напредокот се забележува во зголемена резолуција, брзина и квалитет на излез. Современите монитори поддржуваат висока дефиниција и ултра широк приказ, додека 3Д-печатачите овозможуваат излез во физичка форма на сложени објекти. Овие технологии се користат не само во индустријата, туку и во медицината, архитектурата и уметноста, каде што визуелната и физичката репрезентација имаат клучна улога.

Во некои системи, посебна категорија претставуваат сензорните уреди кои регистрираат физички појави (светлина, звук, движење, температура) и ги претвораат во дигитални податоци. Овие уреди се користат во паметни домови, безбедносни системи и во индустриска автоматизација. Излезните уреди во овие системи може да бидат звучни аларми, осветлување, или известувања преку мобилна апликација, со што се добива ефикасно управување и оддалечен мониторинг.

Слика 8 *Сензорни уреди*

Покрај хардверските карактеристики, важен е и софтверот што ги управува овие уреди – драјверите. Тие овозможуваат оперативниот систем да ја препознае функцијата на уредот и да комуницира со него. Без соодветен софтвер, хардверот не може да функционира правилно, па затоа постои т.н. слој на апстракција помеѓу уредите и програмите што ги користат.

Влезните и излезните уреди претставуваат клучни компоненти за интеракција со компјутерскиот систем. Тие овозможуваат податоците да се внесуваат, обработуваат и прикажуваат во различни форми. Разбирањето на нивните типови, функционалноста и критериумите за класификација е основа за секое понатамошно учење во информатиката. Со развојот на технологијата, овие уреди стануваат сè понапредни, овозможувајќи нови форми на комуникација, автоматизација и визуализација во секојдневниот и во професионалниот живот.

ЗАКЛУЧОК

Категоризирањето на влезни и излезни уреди помага јасно да се разбере како се разменуваат информации меѓу корисникот и компјутерот.

Влезните уреди служат за внесување податоци и команди, додека **излезните уреди** ги прикажуваат резултатите од обработката.

Влезни уреди (Input Devices)

Уреди што внесуваат податоци во компјутерот:

- **Тастатура** – внесување текст и команди,
- **Глувче** – насочување и избор на опции,
- **Скенер** – претворање хартиени документи во дигитални,
- **Микрофон** – внесување звук,
- **Веб-камера** – внесување видеосигнал,
- **Графичка табла (graphics tablet)** – дигитално цртање,
- **Тачскрин екран** (како влез) – внесување со допир,
- **Џојстик/gamepad** – управување во игри,
- **Баркод читач** – читање кодови,
- **Fingerprint скенер** – биометриска автентикација.

Излезни уреди (Output Devices)

Уреди што прикажуваат или испорачуваат податоци од компјутер:

- **Монитор** – визуелен приказ на слики и текст,
- **Печатач** – печатење документи и фотографии,
- **Звучници** – репродукција на звук,
- **Слушалки** – излез на звук,
- **Проектор** – проекција на слика на ѕид/површина,
- **Плотер** – печатење технички цртежи,
- **Тачскрин екран** (како излез) – прикажување слика,
- **LED индикатори** – светлосни сигнали.

Со ваквата претставеност се добива појасна слика за функционирањето на компјутерските системи и помага свесно да се избере вистинската опрема за одредена задача и да стане поумешен, сигурен и самостоен корисник на дигиталната технологија.

ПРИМЕР 1

Скенирање документ

Кога документ се поставува во скенер, уредот го снима текстот или сликата и ја претвора во дигитална форма која потоа се прикажува на екранот. Овој процес го илустрира функционирањето на влезен уред.

ПРИМЕР 2

Печатење извештај

По извршена анализа на податоци во табеларна пресметка, корисникот го испраќа извештајот на печатач. Излезниот уред го претвора дигиталниот документ во физичка хартиена копија.

ПРИМЕР 3

Користење екран на допир

На банкомат, корисникот внесува податоци преку екран на допир, а истовремено добива визуелни и текстуални повратни информации. Овој уред е истовремено и влезен и излезен.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Наведи три влезни уреди што се користат во секојдневниот живот и објасни ја нивната функција!

2. Кои се најчестите излезни уреди и за што се користат?

3. Објасни како функционира еден уред што е истовремено и влезен и излезен!

4. Направи табела во која ќе споредиш три влезни и три излезни уреди според: тип на податоци, точност и брзина!

5. Истражи што е графичка табла и за кои цели се користи!

6. Објасни зошто се важни драјверите за правилно функционирање на уредите!

7. Дефинирај што е биометриски уред и наведи примери!

8. Напиши како функционираат сензорните уреди во паметен дом и кои информации можат да ги собираат!

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи како се користат влезни и излезни уреди во специјализирани професии – на пример, во медицина, архитектура или гејминг! Проучи кои уреди се користат за работа со виртуелна реалност и на кој начин ја симулираат реалната интеракција со дигиталниот простор.

Слика 9 *Влезни и излезни уреди*

1.5 Современи дигитални технологии: допир, холографија, 3Д-проекции

Современата дигитална технологија постојано се развива, воведувајќи нови начини на интеракција, визуализација и управување со информации. Меѓу највпечатливите иновации се технологиите базирани на допир, холографија и тридимензионална проекција. Овие технологии веќе се користат во уреди за секојдневна употреба, но нивниот потенцијал се проширува и во области како медицината, образованието, архитектурата и забавната индустрија. Тие го менуваат начинот на кој луѓето комуницираат со уредите и со дигиталната содржина, овозможувајќи пореални, подинамични и поприродни кориснички искуства.

Технологијата на допир се заснова на можноста корисникот да комуницира директно со дигитална површина, најчесто екран, преку физички контакт. Најчесто се користат капацитивни и резистивни екрани кои реагираат на допир со прст или пенкало. Денес, допирните екрани се интегрирани во паметни телефони, таблети, банкомати, киосци за информации и интерактивни табли во училиштата. Нивната примена се проширува и во автомобилската индустрија, здравството и индустриските контроли, каде што директната и интуитивна интеракција значително ја подобрува ефикасноста.

Слика 10 *Технологија на допир*

Допирните екрани може да препознаваат повеќе точки на допир истовремено – т.н. „multi-touch“ технологија. Таа овозможува операции како зумирање, ротирање и движење на објекти со два или повеќе прста. Оваа функционалност е особено значајна за кориснички интерфејси каде што визуелните манипулации се клучни, како што се апликациите за дизајн, игри или навигација. Во образованието, интерактивните табли со допир ја зголемуваат ангажираноста на учениците и овозможуваат визуелен пристап до комплексни информации.

Холографијата претставува технологија со која се создава тридимензионална слика што може да се гледа без потреба од специјални очила. Холограмот се формира со интерференција на светлински зраци и содржи информација за длабочината, формата и бојата на објектот од сите агли. Овие слики изгледаат како да „левитираат“ во

воздухот, и овозможуваат уникатно визуелно искуство. Холографските проекции се користат во маркетинг, уметност, едукација и презентации за визуелно пренесување на информации на импресивен начин.

Во медицината, холографијата се применува за визуализација на анатомски структури, овозможувајќи подобро разбирање и планирање на хируршки интервенции. Во музеите, холограмите се користат за реконструкција на артефакти или историски настани, а во музичката индустрија – за виртуелни концерти на починати уметници. Оваа технологија продолжува да се развива и ветува нови начини на интеракција со дигиталните содржини во просторот.

Слика 11 *Германски циркус користи холограм за животни*

3Д-проекцијата овозможува приказ на тридимензионални објекти на рамна површина, со употреба на специјални проектори и понекогаш со потреба од очила. Се користи во киноиндустријата, виртуелните тури, архитектонските презентации и симулации. Во образованието, 3Д-проекциите овозможуваат учениците да ја набљудуваат структурата на атомите, анатомијата на телото или историјата на градби со визуелна длабочина што не може да се постигне со класични средства.

Технологијата „Heliodisplay" претставува еден од најреволуционерните примери на проекција без физичка површина. Сликите се проектираат во воздухот, користејќи магла или пареа како медиум. Корисниците можат да „допрат" и да манипулираат со проекциите, што го прави ова решение блиску до холографските екрани од научната фантастика. Овие уреди се користат во изложби, презентации и луксузни промоции, каде што се бара висок степен на визуелно влијание.

Современите дигитални технологии како допир, холографија и 3Д-проекција нудат нови димензии на интеракција и визуализација. Тие го подобруваат корисничкото искуство, го олеснуваат учењето и овозможуваат импресивна презентација на информации. Разбирањето на нивниот принцип на работа и примена претставува неопходност за младите генерации кои ќе живеат и работат во свет во кој ваквите технологии ќе бидат сè почести и понапредни.

ЗАКЛУЧОК

Современите дигитални технологии брзо се развиваат и создаваат нови начини на комуникација, интеракција и визуализација. Разликувањето на овие технологии помага да се разбере нивната примена и да се следат новитетите кои се појавуваат на пазарот што овозможува разликување различни типови интерфејси, начини на внес и прикажување на информации, како и напредни визуелни технологии кои сè повеќе стануваат дел од секојдневието.

Технологија на допир овозможува управување со уредот преку **едноставен допир**, кај постари модели на паметни телефони, банкомати, интерактивни киосци. **Технологија на повеќекратен допир**, поддржува **повеќе истовремени допири**, што овозможува гестови како зумирање, вртење, лизгање. Вградена е во современи паметни телефони (iPhone, Android), таблети, интерактивни табли. **Технологија без допир (Touchless/Gesture-based technology)** овозможува контрола на уредот **без физички контакт**, користејќи гестови, движења или сензори. Се користи кај Xbox Kinect (контрола со движење), бесконтактни прекинувачи со сензор, автоматски врати со сензори, итн.

3Д технологија на слика која создава слика со **длабочина**, која изгледа реалистично или „излегува“ од екранот. Се применува кај 3Д телевизори и филмови, 3Д очила (анаглифни, поларизирани, бленда, итн.), 3Д проектори и 3Д моделирање во игри.

Холографија (Holography) технологија која создава **тридимензионални слики во просторот**, видливи од повеќе агли, без потреба од очила.
Се применува за холограмски концерти (Тупак, Витни Хјустон), холограмски асистенти во аеродроми/музеи, 3Д холограмски реклами, итн. **Хелиодисплеј (Heliodisplay) технологија е** уред кој создава **слика проектирана во воздух**, така што изгледа како да „лебди“.

Корисникот може да се движи низ менито со гестови, без екран и без допир.
Се користи за изложби, интерактивни презентации, маркетинг инсталации со слики „во воздух“ и сл.

Разликувањето на современите дигитални технологии овозможува да се разбере напредокот на технолошките интеракции – од обичен допир, до интеракција без допир и визуелни 3Д и холографски прикази. Со тоа, се развиваат способности да се следат трендовите, да се процени применливоста на секоја технологија и да се стане подготвен корисник на новите дигитални иновации.

ПРИМЕР 1

Интерактивна училница

Во една училница се користи интерактивна табла со екран на допир која им овозможува на учениците да решаваат задачи директно со прст, да повлекуваат објекти и да пишуваат со дигитално пенкало.

ПРИМЕР 2

Холограм на изложба

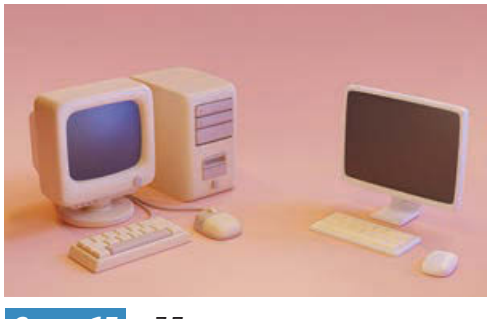
На изложба за технологија, посетителите гледаат холограм на историски објект кој ротира и се прикажува од сите агли. Нема потреба од специјални очила, а ефектот е како објектот да „лебди“ во воздух.

ПРИМЕР 3

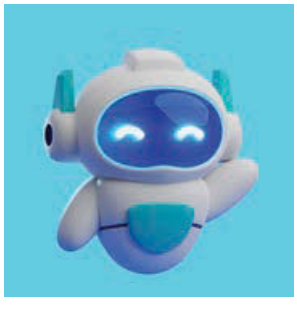
Виртуелна архитектура

Архитект користи 3Д-проектор за да прикаже тридимензионален модел на зграда. Клиентот може да види како изгледа зградата однатре и однадвор, со реална перспектива.

Слика 12 *Хелиодисплеј*



Слика 13 *3Д слики*



Слика 14 *3Д и холограм*

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Објасни како функционира екран на допир и каде најчесто се користи!

2. Што претставува „multi-touch“ и наведи пример каде се користи!

3. Дефинирај го поимот холографија и објасни како се добива холограм!

4. Наведи три области во кои се применува холографијата и објасни зошто!

5. Што е 3Д-проекција и како се разликува од класичен монитор?

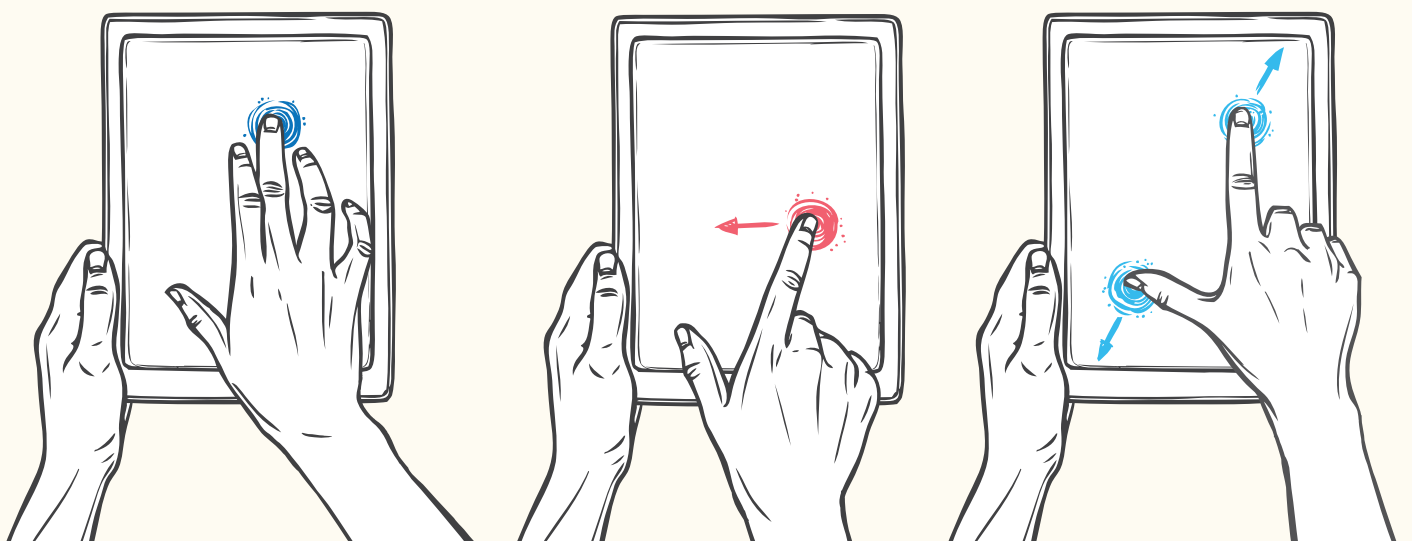
6. Истражи што е „Heliodisplay“ и како функционира!

7. Објасни како современите технологии ги подобруваат учењето и презентациите!

8. Напиши краток текст за иднината на допирните и холографските технологии!

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи како се користат холограми и 3Д-проекции во медицината и во научната визуализација. Проучи нови форми на интерфејси кои не бараат физички контакт – на пример, управување со гестови или гласовна контрола.



1.6 Работната околина на оперативниот систем и на апликативниот софтвер

Работната околина на еден компјутерски систем претставува основа за интеграција и примена на дигиталната технологија. Таа овозможува управување, анализа и автоматизација на процеси преку интеракција со апликации базирани на вештачка интелигенција, алатки за продуктивност и мрежни услуги кои ги зголемуваат ефикасноста, безбедноста и флексибилноста на работниот процес. Интеракциите меѓу човекот, хардверот, софтверот и дигиталните алатки се одвиваат преку интерфејс и неговата инфраструктура, односно работната околина на компјутерскиот систем. Таа е сочинета од:

- оперативен систем (Windows, Linux, macOS),
- графички кориснички интерфејс (GUI),
- алатки и апликации (кориснички програми, бази на податоци, уредувачи, ВИ-платформи),
- мрежни и безбедносни поставки,
- кориснички профили, дозволи и пристапи.

Современите компјутерски системи се базираат на постоење оперативен систем, кој ја управува работата на компјутерот и апликативен софтвер кој овозможува извршување кориснички задачи. Работната околина на оперативниот систем го претставува интерфејсот преку кој корисникот комуницира со хардверот и со програмите. Разбирањето на структурата и функциите на работната околина, како и улогата на апликативниот софтвер е клучно за користење на компјутерот на ефективен и безбеден начин. Овозможува едноставна навигација со оперативниот систем, организирање податоци, стартување програми и работа со различните типови софтвер.

1.6.1 Оперативен систем

Оперативниот систем претставува основен системски софтвер што ги координира сите ресурси на компјутерот – процесорот, меморијата, уредите за влез и излез и апликациите. Најпознати оперативни системи се:

- Windows  ,
- Linux  ,
- macOS  и
- Android 

При вклучување на компјутерот, оперативниот систем се вчитува во меморијата и подготвува работна околина за корисникот. Таа околина вклучува работна површина (desktop), менија, икони и прозорци за програми.

Најавата (login) и одјавата (logout) се основни активности при користење компјутер со повеќе корисници. При најавување, секој корисник добива свој профил со лични поставки, документи и апликации. Работната површина може да содржи икони за брз пристап до апликации, датотеки и системски функции. Преку лентата со задачи (taskbar) се контролираат активните прозорци, се стартуваат програми и се следи статусот на системот. Одредени ставки можеш да ги прилагодиш на сопствените потреби. На пример, јазикот на интерфејсот може да се измени во твојот мајчин јазик, да се додаде јазична поддршка на тастатурата, да се измени кратенка од тастатура за промена на јазикот, автоматско заклучување на екранот по извесно време или пак, компјутерот да бара да се внесе лозинка при секоја најава.

Обиди се да додадеш повеќе корисници на еден компјутер со различни лозинки.

Поврзи го твојот уред на безжична мрежа. Кои параметри треба да ги знаеш?

Апликативниот софтвер овозможува извршување конкретни задачи – пишување текст, пресметки, графички дизајн, сурфање на интернет. Овој софтвер се разликува од системскиот по тоа што не го контролира хардверот, туку се потпира на него. Најчести типови се текстуални процесори (на пример, Microsoft Word, Writer), табеларни пресметки (на пример, Excel, Calc), веб-прелистувачи (на пример, Chrome, Edge), медиумски плеери и образовни апликации.

Датотечниот систем е еден од најважните делови на работната околина. Тој овозможува чување, организирање и пристап до податоци. Датотеките се чуваат во папки (фолдери) кои можат да бидат сместени хиерархиски. Секоја датотека има име и екстензија (на пр. .docx, .jpg, .mp3) која укажува на типот на содржината. Преку графичкиот интерфејс на оперативниот систем, корисникот може лесно да креира, преименува, копира, преместува и да брише датотеки и папки.

Различните оперативни системи имаат слични функционалности, но нивниот изглед и начинот на пристап може да се разликуваат. На пример, Windows користи мени Start, икони и File Explorer, додека Linux околините како GNOME и KDE имаат различни панели и прегледи на датотеки. На мобилните уреди, Android и iOS нудат пристап до датотеки преку апликации и облак, каде што синхронизацијата со други уреди се изведува автоматски.

Апликациите може да се инсталираат од различни извори – онлајн продавници (како Microsoft Store, Google Play), инсталациски датотеки (.exe, .msi) или од мрежа. Важно е корисникот да обрне внимание на изворот и на лиценцата на софтверот. Лиценцата ги одредува правата на користење: дали е апликацијата бесплатна, отворен код, комерцијална или ограничена за одреден период (пробна верзија). Creative Commons лиценците се користат за содржини како документи, слики и образовни материјали, при што авторите дозволуваат различни нивоа на употреба.

Оперативниот систем исто така вклучува алатки за безбедност и одржување – како **заштитни програми** (антивируси), ажурирања на системот, контрола на кориснички пристап и резервни копии. Овие функции се суштински за заштита од злонамерен софтвер и за зачувување на податоците во случај на дефект. Корисниците треба редовно да ги ажурираат своите уреди и да користат сложени лозинки, со цел да се минимизира ризикот од упад или загуба на информации. На компјутерот може да се инсталираат и дополнителни програми за заштита покрај оние што ги содржи оперативниот систем.

За поголема прегледност, поврзаноста на дигиталната технологија со работната околина на компјутерски систем е прикажана во табелата подолу.

Дигитална технологија	Врска со работната околина
Вештачка интелигенција (AI)	Интеграција во оперативни системи за автоматизација (гласовни асистенти, интелигентни препораки, безбедносно скенирање).
Облачни технологии (cloud)	Работната околина се проширува од локален компјутер кон онлајн работна средина (Google Workspace, Microsoft 365).
Алатки за далечинска работа	Работната околина станува мобилна: пристап до ресурси од каде било преку VPN, Remote Desktop, Teams, Zoom.
Кибер безбедност	Интеграција на антивируси, „firewalls“, енкрипција – сето тоа функционира во рамки на работната околина.
Алатки за продуктивност и автоматизација	Microsoft Copilot, Notion AI, Grammarly – дел од софтверската работна околина што го зголемува квалитетот на работа.
Интерактивни интерфејси	Дигитални табли, тач-екрани, гестикулација и VR/AR влезови во работната околина.

Табела 1 *Поврзаност на дигиталната технологија со компјутерски систем*

Дигиталната технологија (ВИ, облак, автоматизација) функционира во рамките на **работната околина на компјутерскиот систем**, овозможувајќи побрза, поефикасна и побезбедна работа.

ЗАКЛУЧОК

Оперативниот систем претставува основен софтвер кој го контролира и управува со целокупното функционирање на компјутерот. Тој ги поврзува хардверските компоненти со корисникот, овозможува извршување програми, управува со датотеки, меморија, уреди и безбедносни процеси. Преку оваа функционалност, оперативниот систем ја создава корисничката околина во која корисниците можат да работат користејќи ги менито, работната површина, поставките, прозорците и другите графички елементи што го олеснуваат користењето на компјутерот.

Разликувањето помеѓу оперативен систем и апликативен софтвер е еден од основните дигитални концепти. Додека оперативниот систем претставува основа без која компјутерот не може да функционира, апликативниот софтвер ги извршува конкретните задачи на корисникот – обработка на текст, цртање, комуникација, игри, интернет-прегледување. Според намената, апликациите може да се поделат на: образовен софтвер, деловни апликации (офис пакети), мултимедијален софтвер, програми за комуникација, игри, графички алатки и софтвер за безбедност. Покрај користење софтвер, важно е и да се заштити системот. Затоа, корисниците треба да ги знаат критериумите за оценување на антивирусни и безбедносни програми, како што се: степен на заштита од вируси и малициозен софтвер, брзина на работа, ажурирања, фајрвол опции, корисничка поддршка, влијание врз перформансите на компјутерот и дополнителни функции (антифишинг, рансомвер заштита и сл.). Применувајќи ги овие критериуми, се обезбедува голема безбедност во работата со дигиталните уреди.

ПРИМЕР 1

Организирање документи во папки

Корисникот креира папка на работната површина со име „Домашни задачи“. Во неа ги зачувува сите Word-документи што ги подготвува за училиштето, поделени по предмети. Со тоа се добива прегледност и лесен пристап до потребните датотеки.

ПРИМЕР 2

Стартување апликација преку лента со задачи

На долниот дел од екранот, ученикот го кликнува иконче за веб-прелистувач (Chrome) на taskbar. Со тоа апликацијата се отвора веднаш, без да се пребарува низ менија.

ПРИМЕР 3

Инсталирање софтвер со лиценца

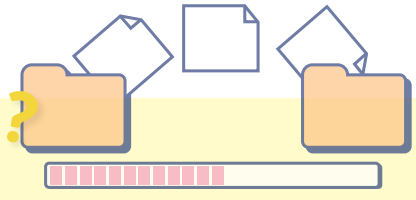
Наставникот инсталира образовна програма за учење математика. При инсталацијата избира бесплатна верзија со Creative Commons лиценца, со што може да ја користи легално и без ограничувања.

ПРИМЕР 4

Директна поврзаност со дигиталната технологија

Наставникот работи во компјутерска работна околина што вклучува:

- LMS (Learning Management System),
- ВИ-алатка што го анализира напредокот на учениците,
- Се врши автоматска синхронизација со облак,
- Има безбедносна заштита преку енкрипција и двофакторска автентикација.



ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Објасни ја улогата на оперативниот систем во компјутерот.

2. Што се подразбира под поимот „работна околина“?

3. Кои се разликите помеѓу оперативен систем и апликативен софтвер?

4. Наведи неколку активности што може да се извршат преку работна површина.

5. Објасни што е лиценца за софтвер и зошто е важна.

6. Напиши кои апликации најчесто ги користиш и за што ги користиш.

7. Истражи која е разликата помеѓу отворен и комерцијален софтвер.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи кои оперативни системи се користат во мобилните уреди, како се разликува нивната работна околина од десктоп компјутерите и кои се нивните предности. Проучи што се виртуелни машини и како се користат за тестирање и симулација на различни оперативни системи!

1.7 Етички правила за користење на дигиталната технологија

Разликувањето и разбирањето на современите дигитални технологии — со допир и повеќекратен допир на екрани, технологии без допир, 3Д прикази, холографија и хелиодисплеј — овозможува да се препознаат нивните можности и ограничувања, но и подготовка за сигурно, креативно и свесно користење на технологијата. Во исто време, важен дел од дигиталната писменост е и познавањето на **етичките правила** за користење дигиталната технологија. Почитувањето на приватноста и личните податоци, избегнувањето навредлива комуникација и кибернасилство, почитувањето авторски права, како и проверување на точноста на информациите пред да се споделат, води кон користење на технологијата на одговорен начин, без злоупотреби, мамење или штети. Етичкото користење на дигиталната технологија претставува основа за безбедно, одговорно и свесно функционирање во современиот дигитален свет. Како што се развива технологијата, така расте и потребата корисниците, особено младите, да научат како правилно да се однесуваат онлајн, како да ги заштитат своите податоци и како да покажат почит кон другите во виртуелната средина.

Заедно со технолошката писменост треба да се развие и силна **дигитална етика**, бидејќи напредната технологија носи и одговорности. Преку конкретни ситуации, треба да се учи како безбедно и правилно да се однесуваме, односно како да се стане одговорен дигитален граѓанин.

Прво и најважно, етиката во користењето технологија започнува со **почитување на приватноста**. Тоа значи дека не смееме да објавуваме туѓи фотографии, видеа или лични податоци без дозвола, дури и кога ни изгледа безбедно или „шега“. Личните информации секогаш треба да бидат заштитени – лозинките мора да бидат силни, не се споделуваат со други лица и не се внесуваат на сомнителни веб-страници или апликации.

Пример: Ученици не снимаат со мобилен (паметен) телефон на час без дозвола од наставникот или надвор од училиште без дозвола од учесниците.

Следен важен сегмент при етичкото користење на технологијата е **Заштитата на личните податоци.** Пример: Не се оставаат свои отпечатоци или биометриски податоци на терминали со допир или без допир што не се доверливи. Следниот важен сегмент е **почитување на авторските права**. Во дигиталниот свет, многу содржини се лесно достапни како што се слики, музика, видеа и текстови. Но, тоа не значи дека може да ги користиме без дозвола. Секогаш треба да се наведе изворот, да се користат

материјали со дозвола или бесплатни ресурси што дозволуваат образовна употреба. Така се развива култура на почит и интегритет. Пример: Кога се прави презентација со 3Д модели, се наведуваат изворите од каде е преземен моделот.

Истовремено, треба да се внимава и на **комуникацијата на интернет**. Онлајн просторот не смее да биде место на омаловажување, исмевање или ширење негативност. Секој збор има последица, дури и кога е напишан зад екран. Треба да знаеме да пишуваме љубезно, да не користиме навредливи коментари и да се воздржуваме од учество во групни напади или шеги кои можат да повредат некого. Поединечните или групни напади подлежат на законска регулатива и се казниви. Во вакви случаи молчењето не помага, пријавувањето е храбар и правилен избор. Исто така, и **кибернасилството** е една од најсериозните закани во дигиталното општество. Етичкото користење на технологијата значи активно избегнување на вакви форми на однесување, но и **препознавање и пријавување** кога ќе се забележат.

Денес, кога информациите се шират со огромна брзина, важен дел од дигиталната писменост е и **проверката на точноста на информациите**. Тоа значи дека пред да споделиме некоја вест, видео или фотографија, треба да провериме дали е изворот веродостоен. На овој начин се спречува ширење лажни вести, манипулации и паника.

Етичкото користење на технологијата значи и **одговорно користење на уредите**.

Технологијата никогаш не треба да се користи за измама, препишување, хакирање, лажно претставување или нанесување штета. Овој принцип важи и во училишната средина — уредите не смее да се злоупотребуваат за фотографирање тестови, препишување или снимање ученици и наставници без дозвола.

Не помалку важни се и **самоконтролата и балансираното користење на технологијата.** Прекумерното користење мобилни телефони, социјални мрежи или игри може да влијае негативно врз здравјето, учењето и меѓучовечките односи. Дел од етичките навики е и препознавање на сопствените граници, поставување умереност и правење свесни избори.

ПРИМЕР 1

Проверка на точноста пред споделување содржина

Пред да сподели видео или вест на Инстаграм, ТикТок или Фејсбук, ученикот проверува дали е изворот веродостоен и го наведува точниот извор. На овој начин се спречува ширење лажни вести и манипулации.

ПРИМЕР 2

Етичко користење на училишни уреди и интернет

При работа во компјутерска лабораторија, ученик забележува дека претходниот ученик останал најавен на Google Classroom. Тој не го прегледува профилот, туку го одјавува и се логира со своја сметка. Ученик кој користи училиштен компјутер или таблет, не се логира на туѓи профили, не менува поставки без дозвола и не презема нелегални содржини. Наместо тоа, ги користи уредите само за училишни активности и ги почитува правилата на училиштето.

ПРИМЕР 3

Етичко снимање и користење дигитални медиуми

Ученик кој снима проект за училиште (на пример: видеопрезентација или интервју) претходно бара дозвола од сите што се појавуваат на снимката и јасно објаснува за што ќе се користи видеото. За проект по историја, групата ученици интервјуира наставник. Пред снимање, тие го замолуваат за дозвола, го информираат дека записот ќе се прикаже само во класот и му ја праќаат финалната верзија за одобрување пред да го објават.

Слика 15 *Паметно користење телефон*

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Што претставува етичко користење на дигиталната технологија? Наведи две правила!

2. Објасни зошто е важно да не се објавува туѓа фотографија без дозвола, дури и ако ти изгледа безопасно!

3. Замисли дека добиваш порака во која некој навредува твој соученик. Што би направил/а според правилата за етичко дигитално однесување?

4. Прочитај го следниов пример и кажи што е етички правилно, а што е неправилно:
Ученик снима дел од часот со свој мобилен телефон и го споделува видеото во приватна група без да го праша наставникот. Образложи ги причините.

5. Процени ја следнава ситуација: дали е етички да се сподели вест што звучи шокантно без да се провери изворот? Објасни ја твојата одлука!

6. Осмисли кратко правило кое би го додал/а во „Училиштен дигитален кодекс“ за да ја подобриш дигиталната култура меѓу учениците!

7. Опиши како би постапил/а доколку некој во групен разговор (чат) сподели лажна информација која може да предизвика страв или паника. Кои чекори би ги презел/а за да ја решиш ситуацијата етички?

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи колку деца во светот биле изложени на закани од интернет и негово неетичко користење. Прочитај ја последната верзија на Актот за дигитални услуги на Европската комисија на следнава адреса:

https://digital-strategy.ec.europa.eu/en/policies/digital-services-act-package.

1.8 Користење папки и датотеки во хиерархиска структура

Во секој компјутерски систем информациите се чуваат во форма на датотеки, а нивната организација се врши преку папки. Овој начин на уредување, познат како хиерархиска структура, овозможува логично групирање, едноставен пристап и ефикасно управување со податоците. Без правилна организација, би се појавиле тешкотии во пронаоѓање, отворање и зачувување на датотеките, особено кога количината на информации е голема. Разбирањето на хиерархијата на папки (фолдери) и именување на датотеки е основа за дигитална писменост и безбедно користење компјутери.

Папките (или директориуми) претставуваат простори во кои се чуваат датотеки или други папки. Хиерархискиот систем значи дека папките може да содржат потпапки, со што се создава структура слична на дрво. На врвот на оваа структура се наоѓа коренската папка (root), а од неа се разгрануваат сите други поддиректориуми и датотеки.

Секоја датотека има свое уникатно име и екстензија која укажува на видот на податокот. На пример, .txt означува текстуална датотека, .jpg слика, .mp3 аудиозапис. Препорачаното именување вклучува описни имиња (на пр. „Домашна_математика_мај.docx“) со избегнување на специјални знаци. Ова ги олеснува идентификацијата и пребарувањето, особено кога бројот на датотеки е голем.

Датотечните системи може да се прикажат преку графички прелистувачи како File Explorer во Windows или Finder во macOS. Корисникот може да креира нова папка, да копира или да премести датотека, да ја преименува или да ја избрише. Достапни се и функции за сортирање според име, тип, големина или датум на последна промена. Во напредни случаи, организирањето се врши и преку тагови, ознаки или категории, што овозможува дополнително филтрирање.

Хиерархиската структура овозможува логичко групирање на датотеки според проекти, предмети, времетраење или намена. На пример, во една главна папка со име „Училиште“, може да се создадат потпапки за секој предмет (на пр. „Математика“, „Хемија“), а во секоја од нив да се чуваат документи, презентации и задачи. Оваа организација им овозможува на учениците и на наставниците брзо и лесно да се снајдат меѓу податоците без потреба од долготрајно пребарување.

Датотеките и папките може да се организираат и според година, месец или недела. На пример, слики од одреден настан може да се зачуваат во структура: „2025 → Март → Екскурзија“. Овој пристап е многу корисен за архивирање и следење на напредокот по периоди, како и за олеснување на споделувањето со други корисници преку мрежа или облак.

Облак услуги како Google Drive, Dropbox и OneDrive нудат ист принцип на хиерархиска организација, но со можност за пристап од кој било уред и споделување во реално време. Корисниците можат да креираат папки, да влечат и да пуштаат документи, да поставуваат дозволи (само за читање, уредување) и да ги следат измените. Ова е особено корисно за тимска работа, училишни проекти и дигитално портфолио.

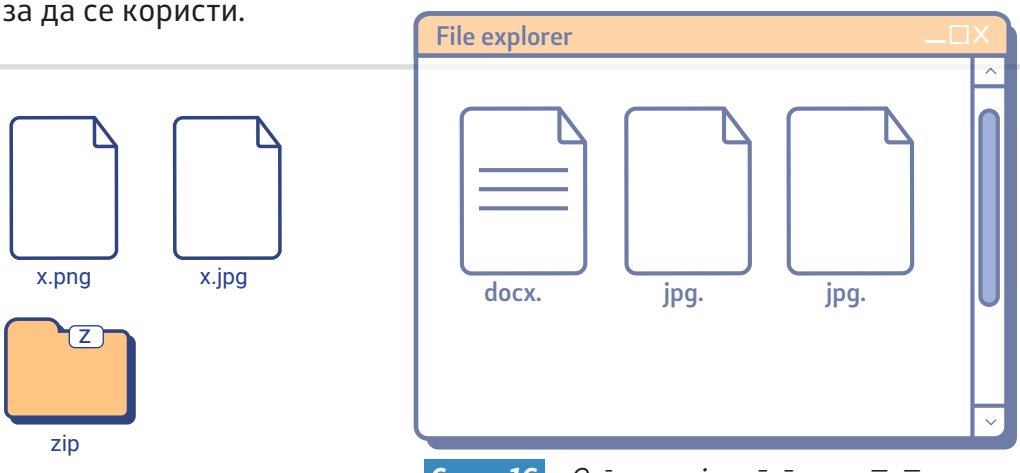
Добрата практика во организирањето вклучува редовно чистење на непотребни датотеки, правење резервни копии и креирање структура уште од почетокот на работата на компјутерот. Неправилната организација води до губење време, дуплирање податоци и потенцијална загуба на важни информации.

Организирањето датотеки и папки во хиерархиска структура претставува основна вештина за секој компјутерски корисник. Соодветната организација ја подобрува ефикасноста, го намалува ризикот од губење податоци и овозможува полесно споделување. Дигиталната писменост започнува токму со овие принципи, кои остануваат применливи и важни независно од уредот, оперативниот систем или софтверот што се користи.

ЗАКЛУЧОК

Постојат различни видови формати на документи. На пример, .docx и .pdf се формати за текстуални документи, .jpg и .png за слики, .mp4 за видеа, .mp3 за аудио, додека .zip и .rar се формати за компресирани датотеки. Со препознавање на овие формати, полесно се организираат податоците и се поврзуваат датотеките со соодветните програми кои ги отвораат – како Word за .docx, PowerPoint за .pptx, Adobe Reader за .pdf, прегледувачи на слики за .jpg/.png и медија-плеери за аудио и видеоформати.

Хиерархиската организација на датотеките е важна за секој корисник. Таа подразбира создавање папки и потпапки по логичен редослед. Ваквата структура ги олеснува наоѓањето, уредувањето и чувањето на податоците и го спречува создавањето неред и загуба на материјали. Покрај препознавање формати и организација, важно е да се знае и разликата помеѓу тип на датотека (format/file type) и програмата што ја отвора. Еден од најважните делови е анализата на компресијата и нејзиното влијание врз големината и квалитетот на датотеките. Кај аудио или видеодатотеки, компресијата може да влијае врз јасноста на звукот и сликата. Кај архивите (.zip, .rar) се намалува големината без да се изгуби оригиналниот квалитет, но датотеката мора да се распакува за да се користи.



Слика 16 *Организација на папки и датотеки*

ПРИМЕР 1

Организирање датотеки за предмети

Учениците креираат главна папка со име „Училиште“, во која организираат три потпапки: „Математика“, „Информатика“ и „Историја“. Во секоја од овие потпапки се зачувани документи во Word-формат со домашни задачи, PDF-презентации и слики од таблата.

ПРИМЕР 2

Структура според датуми

Во папката „Фотографии“, се креираат потпапки по месеци: „2025_Јануари“, „2025_Февруари“ и „2025_Март“, каде што се чуваат слики од секој настан подредени по хронологија.

ПРИМЕР 3

Облак-организација

Еден ученик користи Google Drive за да ги чува сите свои проекти. Тој има папка „Проекти“, со потпапки по предмет. Во секоја папка, документите се именувани според содржина и датум (на пр. „Истражување_биологија_2025_03.docx“).

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Објасни што претставува хиерархиска структура на датотеки!

2. Кои се придобивките од правилно именување на датотеките?

3. Направи шема како би организирал/а документи за сите предмети во едно полугодие!

4. Што е екстензија на датотека? Наведи 5 различни примери и објасни ги!

5. Објасни како се креира нова папка и како се преместува датотека во неа!

6. Кои функции се достапни во File Explorer или во Finder?

7. Што е разликата помеѓу локално и облачно чување на датотеки?

8. Истражи што претставуваат резервни копии и зошто се важни!

ПРАКТИЧНА ЗАДАЧА

Оцени ги програмите за заштита на компјутерите подолу во согласност со критериумите!
Напиши го и името на програмата!

Критериум	Опис	Име на програмата	Оценка (1-5)
Ефикасност во откривање закани	Способноста на програмата да открие вируси, малвер, шпионски софтвер и други закани.		
Реално-временска заштита	Дали обезбедува заштита во реално време при сурфање, преземање и отворање датотеки.		
Брзина и перформанси	Колку влијае програмата врз брзината на системот. Дали работи тивко во заднина.		
Употребливост и интерфејс	Дали е лесна за користење, со интуитивен интерфејс и јасни опции.		
Фреквенција на ажурирања	Колку често се ажурира базата на податоци за нови закани.		
Дополнителни функции	Firewall, родителска контрола, VPN, заштита на веб-камери, итн.		
Поддршка и документација	Достапност на техничка поддршка, упатства, FAQ.		
Цена и лиценцирање	Дали е бесплатна, пробна верзија, или платена. Дали е цената оправдана.		
Компатибилност	Поддршка за различни оперативни системи (Windows, macOS, Linux).		
Рецензии и репутација	Мислења од корисници и експерти, награди и признанија.		

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи што значи поимот „структура на фајл-систем“ и какви типови фајл-системи постојат (на пример: FAT32, NTFS, ext4). Проучи кои се добрите практики за организација на голем број датотеки и кои алатки постојат за автоматизација на именување и сортирање.

1.9 Лиценцирање на софтвер и Creative Commons

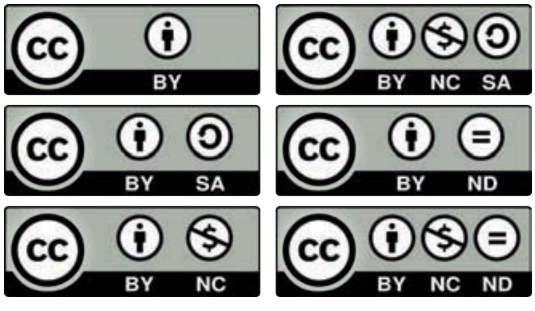
Кога наставниците и учениците користат различни дигитални алатки во наставата, тие треба да проверат **под кои лиценци работат овие алатки** – дали се со затворен код, дали може да се адаптираат и дали е дозволено нивно користење без нарушување на авторските права. Исто така, кога се создава **образовна содржина** со помош на дигиталните алатки, особено со ВИ (на пр., автоматска генерација на текст), потребно е да се внимава дали може да се споделат резултатите или се заштитени со други лиценци.

Во светот на информатиката, софтверот и дигиталните содржини се создаваат од автори кои имаат одредени права врз нивната употреба. За да се регулира начинот на кој другите можат да ги користат тие производи, се користат лиценци. Лиценците го определуваат правото на инсталирање, копирање, модифицирање и дистрибуирање на софтверот или содржината. Creative Commons претставува современ пристап кон лиценцирање кој им овозможува на авторите да споделуваат свои дела под јасно дефинирани услови. Разбирањето на овие концепти е суштинско за етичко користење на софтвер, мултимедијални содржини и образовни материјали.

Софтверските лиценци се правни договори помеѓу авторот или производителот на софтверот и крајниот корисник. Постојат различни видови лиценци, од кои секоја дозволува различни нивоа на користење. Комерцијалниот софтвер обично доаѓа со лиценца што ограничува користење само на една копија, без право на измена или споделување. Такви примери се Microsoft Office, Adobe Photoshop и други професионални пакети. Корисникот мора да ги прифати лиценцните услови пред да го користи софтверот, најчесто во форма на EULA (End User License Agreement).

Од друга страна, постојат и бесплатни лиценци како Freeware, кои дозволуваат користење без паричен надомест, но најчесто без можност за модификација. Shareware дозволува пробен период, по кој е потребна регистрација. Open Source (отворен код) лиценците, пак, му дозволуваат на корисникот да го користи, изучува, менува и дистрибуира софтверот. Најпознати примери се GNU General Public License (GPL), MIT License и Apache License.

Creative Commons (CC) е тип на лиценцирање што најчесто се користи за содржини како слики, текстови, презентации, видеа и музика. Наместо целосно забранување, CC дозволува различни нивоа на употреба, во зависност од изборот на авторот. Постојат шест основни типови CC лиценци кои комбинираат услови како: задолжително наведување на авторство (BY), некомерцијално користење (NC), непромена (ND) и споделување под исти услови (SA).



Слика 12 *Типови CC лиценци*

На пример, лиценцата CC BY дозволува користење, споделување и модификација, сè додека се наведува авторот. CC BY-NC дозволува исто, но не за комерцијални цели. CC BY-ND дозволува споделување, но не и модификација. Овие комбинации овозможуваат флексибилно користење на ресурси со почитување на правата на авторот.

Во образованието, CC лиценците се сè почесто користени за споделување наставни материјали. Многу универзитети, библиотеки и истражувачки организации ги објавуваат своите содржини со овие лиценци, за да ги направат достапни на глобално ниво. Ова придонесува за отворено знаење, подобра соработка и дигитална солидарност.

Важно е секој корисник да внимава на лиценцата на софтверот или на содржината што ја користи. Нелегалното копирање, дистрибуција или измена претставува кршење на авторските права и е казниво со закон. Покрај правните последици, ваквото однесување ја нарушува етиката на дигиталното граѓанство и ја поткопува работата на креаторите на содржини.

Постојат и јавни домени (Public Domain), во кои содржината нема ограничувања за користење. Овие ресурси се слободни за употреба, без лиценци, бидејќи авторските права се истечени или авторот изречно се откажал од нив. Сепак, и при користење на такви извори, се препорачува да се наведе изворот од почит кон оригиналниот креатор.

Лиценцирањето претставува основен механизам со кој се регулира користењето на софтвер и дигитална содржина. Со познавање на различните видови лиценци, корисниците можат одговорно да ги употребуваат алатките и информациите што им се достапни. Лиценците Creative Commons особено придонесуваат кон отворено споделување, но и поставуваат јасни услови за тоа. Преку почитување на овие правила се гради дигитална култура заснована на соработка, почит и правичност.

ЗАКЛУЧОК

Лиценцирањето на софтвер и дигитални содржини има голема улога во тоа како корисниците смеат да ги користат, споделуваат и изменуваат програмите, документите, сликите или видеата. Софтверските лиценци, како комерцијални, бесплатни, слободни или отворен код, одредуваат дали корисникот може да го менува софтверот, да го дистрибуира или само да го користи под одредени услови. Во рамки на дигиталните содржини, голема важност имаат **Creative Commons** (CC) лиценците. Тие се создадени за да му помогнат на авторот да одреди како другите да ја користат неговата содржина. Сите CC лиценци се базираат на комбинација од дозволи и ограничувања, што овозможува авторот да задржи одредени права, а сепак да дозволи слободно користење под одредени услови. Најчестите Creative Commons лиценци се:

- **CC BY (Наведи автор)** – дозволува користење, споделување и изменување на содржината, но мора да се наведе оригиналниот автор.
- **CC BY-SA (Наведи автор – Сподели под исти услови)** – дозволува изменување, но новото дело мора да се сподели под истата лиценца.

- **CC BY-ND (Наведи автор – Не менувај)** – дозволува користење, но не дозволува изменување.
- **CC BY-NC (Наведи автор – Некомерцијално)** – дозволува користење и изменување, но не за комерцијални цели.
- **CC BY-NC-SA / CC BY-NC-ND** – комбинации од некомерцијално, без менување и споделување под исти услови.
- **CC0** – авторот се откажува од сите права, содржината е во јавен домен.

Разликувањето на CC лиценците овозможува правилно да се користат дигитални ресурси, да се цитираат извори и да се однесуваат етички и законски кога се создава или презема содржина. Со познавање на лиценците, може да се одлучи како да се заштити сопствената работа и како да се почитува работата на другите.

ПРИМЕР 1

Бесплатен, но ограничен софтвер

Еден ученик користи текстуален едитор што е Freeware. Иако не плаќа за него, не смее да го менува кодот или да го дистрибуира на други, бидејќи условите го забрануваат тоа.

ПРИМЕР 2

Отворен код со лиценца GPL

Наставник по Информатика користи Linux дистрибуција со GPL лиценца. Тој може да ја менува и да ја споделува со други, сè додека новата верзија се дистрибуира под истите услови.

ПРИМЕР 3

Презентација со Creative Commons

Наставник креира пауерпоинт-презентација со слики и музика преземени од интернет со CC BY-NC лиценца. Тој ги навел авторите и ја користи содржината само за едукативни, некомерцијални цели.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Објасни зошто постојат лиценци за софтвер и дигитални содржини!

2. Кои се разликите помеѓу комерцијален софтвер и отворен код?

3. Наведи 3 типа на Creative Commons лиценци и објасни ги!

4. Наведи ги разликите помеѓу видовите лиценци (CC BY, CC BY-SA, CC BY-ND, CC BY-NC, итн.)!

5. Што значи ако некој софтвер е во јавен домен?

6. Какви се последиците од користење нелиценциран софтвер?

7. Истражи што е MIT лиценца и за што се користи!

8. Дали сите бесплатни програми се и легални за користење? Објасни!

9. Пронајди еден софтвер со CC лиценца и објасни како се користи!

ПРАКТИЧНА ЗАДАЧА

Креирај документ со оригинална содржина која има текст (есеј, блог пост, поезија) и слика/графика. Содржината треба да биде едукативна или креативна. Избери соодветна CC лиценца за нивната содржина. Објасни зошто ја избра токму таа лиценца! Што дозволува, а што ограничува?

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи како се лиценцираат мобилни апликации и кои се условите на користење според продавниците како Google Play или App Store! Проучи што е софтверска пиратерија и кои се нејзините правни и етички импликации!

1.10 Вештачка интелигенција (општа и генеративна)

Вештачката интелигенција (ВИ) претставува една од најзначајните и најбрзорастечки технологии во современиот дигитален свет. Со неа, машините се способни да учат, заклучуваат, комуницираат и решаваат проблеми што традиционално бараат човечка интелигенција. Сè повеќе системи, апликации и уреди се потпираат на некаков облик на ВИ – од паметни телефони и навигатори до медицински дијагностички алатки и автономни возила. Оваа лекција има цел да ги претстави поимот, поделбата и примената на вештачката интелигенција, со акцент на нејзините два современи правци: општа и генеративна ВИ.

Вештачката интелигенција се дефинира како способност на компјутерски систем да извршува задачи што бараат интелигентно однесување – како препознавање говор, решавање проблеми, учење од искуство и донесување одлуки. Во зависност од нивото на интелигенција и способностите, постојат две главни категории: т.н. општа ВИ (AGI – Artificial General Intelligence) и „специјализирана“, за одредена намена или генеративна ВИ.

Општата вештачка интелигенција има цел да создаде систем што може да размислува, да учи и да се снаоѓа во различни ситуации како човек. Оваа форма на ВИ сè уште е во теоретска фаза и не постои во практична примена. Таа би требало да комбинира логика, емоции, контекст и морал во своето функционирање. Истражувањата во оваа насока се фокусирани на создавање универзални алгоритми и модели на учење што би функционирале независно од задачата или околината.

Општата ВИ, иако не е развиена на потребното ниво, продолжува да биде активна тема на академски и технолошки истражувања. Моделите се развиваат за да го симулираат човечкото размислување, со фокус на машинско расудување, разбирање на контекст и адаптација кон нови ситуации. Целта е да се создаде систем кој нема да биде ограничен на една задача, туку ќе може да „размислува“ низ повеќе домени – слично како човек.

Име на ВИ / Компјутер	Креиран за проектот	Што прави?
ХАЛ 9000	2001: Вселенска одисеја	Контролира вселенски брод
Дата (андроид)	Стар трек	Машина со човечко размислување и емоции
Џарвис/ Фрајдеј	Железен човек/Одмаздници	Личен помошник кој има и емоционална интелигенција
Скајнет	Терминатор	Воен ВИ систем што станува самосвесен и опасен

Табела 2 *Примери за општа ВИ*

Наспроти тоа, генеративната вештачка интелигенција претставува практичен тип на ВИ која денес веќе се користи во различни полиња. Оваа ВИ е специјализирана за креирање нови содржини – текст, слики, музика, видео или дури и програмски код. Генеративните модели, како оние базирани на длабоко учење (deep learning), се обучуваат со огромни количини податоци и потоа се користат за создавање нови, оригинални изрази што наликуваат на човечко творештво.

Генеративната ВИ е веќе присутна во секојдневниот живот. На пример, јазични модели како оние што овозможуваат автоматско допишување на реченици во мобилни телефони, преведување текст, или дури пишување цели статии. Визуелни генеративни системи, како DALL·E или Midjourney се користат за создавање уникатни слики врз основа на текстуални описи. Во музичката индустрија, алгоритми создаваат композиции, а во филмската – виртуелни гласови или ликови. Сите овие апликации ги комбинираат податоците и статистиката со креативност што потсетува на човечки начин на размислување.

Карактеристика	Општа ВИ (AGI)	Генеративна ВИ (GenAI)
Област на примена	Универзална (како човек)	Ограничена, специфична
Способност за учење	Самообучување, универзално	Учи од големи податоци
Свесност / разбирање	Потенцијално (теоретски)	Не, само статистички модел е
Пример	Фиктивен компјутер од научна фантастика	ChatGPT, DALL·E, Copilot
Статус	Теорија, сè уште не постои	Реална и користена денес

Табела 3 *Споредба меѓу општа и генеративна ВИ*

Алгоритмите на генеративната ВИ најчесто користат архитектури наречени невронски мрежи – посебно т.н. генеративни спротивставени мрежи (GANs) и трансформери. Овие системи учат со обработка на милиони примери, при што ги извлекуваат основниот образец, структурата и стилот на содржината. Потоа, тие ги користат тие „учени" обрасци за создавање нешто ново, што наликува на оригиналот, но е оригинално. На пример, може да се генерира слика од „замок на Месечина" кој никогаш не постоел, но изгледа реално.

Сепак, и покрај напредокот, постојат бројни предизвици поврзани со генеративната ВИ. Проблеми како неточни информации наведуваат на погрешна претстава за појава или процес, лажни слики (deepfakes), непожелна пристрасност и авторски права за кои е потребно особено внимание. Потребни се етички рамки, закони и алатки за откривање и контрола на содржините што ги генерираат овие системи, особено кога се користат во образованието, во новинарството, или на социјалните мрежи.

Вештачката интелигенција, во своите општи и генеративни форми, се појавува како алатка со огромен потенцијал за подобрување на различни аспекти од животот. Додека општата ВИ сè уште се развива како визија за иднината, генеративната ВИ веќе се користи во бројни полиња. Познавањето на овие технологии е неопходно за разбирање на модерниот дигитален свет, но и за одговорно користење. Со соодветна примена и регулирање, ВИ може да се претвори во моќен сојузник на образованието, науката, уметноста и секојдневието.

Дополнително, вештачката интелигенција се применува и во персонализирани препораки на содржини, како што се филмови, музика или производи на платформи како Јутјуб (YouTube), Нетфликс (Netflix) и Амазон (Amazon). Алгоритмите следат навики на корисниците, собираат податоци и создаваат предлози што го олеснуваат изборот и го зголемуваат задоволството од користење. Овие системи стануваат сè порафинирани и не само што учат од индивидуалниот корисник, туку прават споредби со илјадници други корисници за да понудат најрелевантна содржина.

Во здравството, генеративната ВИ се користи за анализа на медицински податоци и генерирање извештаи, моделирање молекули за нови лекови, па дури и симулација на ефекти од одредени терапии. На овој начин се забрзува истражувањето и се подобрува дијагностиката. Некои ВИ-системи веќе покажуваат способност да препознаваат симптоми на болести со точност слична или повисока од човечки експерти.

При користењето на ВИ треба да се почитуваат голем број препораки на глобално ниво, законски регулативи на ЕУ, но и национални. Моќта што ја има во себе, не треба да се злоупотребува. При користење ВИ треба да се почитуваат следниве препораки:

1. Приватност и заштита на лични податоци

- ВИ не смее да прибира или да обработува лични информации без согласност.
- Податоците мора да бидат заштитени од злоупотреба, кражба или неовластен пристап.
- Секој корисник треба да знае кои податоци се собираат и зошто.

2. Транспарентност (јасност и објаснивост)

- ВИ-системите треба да бидат јасни во своето функционирање.
- Корисникот треба да знае дека комуницира со ВИ, а не со човек.
- Важно е ВИ да може да објасни како донела одредена одлука (на пример: зошто му доделила одредена оценка на ученик/чка).

3. Фер однесување и отсуство на дискриминација

- Податоците со кои се тренира ВИ треба да бидат разновидни и избалансирани.
- ВИ не смее да биде пристрасна по раса, пол, возраст, религија или други основи.

Човечка контрола и одговорност

- Корисниците треба секогаш да имаат крајна одговорност за одлуките на ВИ.
- ВИ не треба да донесува одлуки без можност за човечка интервенција (особено во чувствителни области, како здравство или образование).

Безбедност и сигурност

- ВИ-системите треба да бидат заштитени од кибернапади, злоупотреби или грешки.
- Треба редовно да се тестираат и да се ажурираат за да функционираат безбедно.

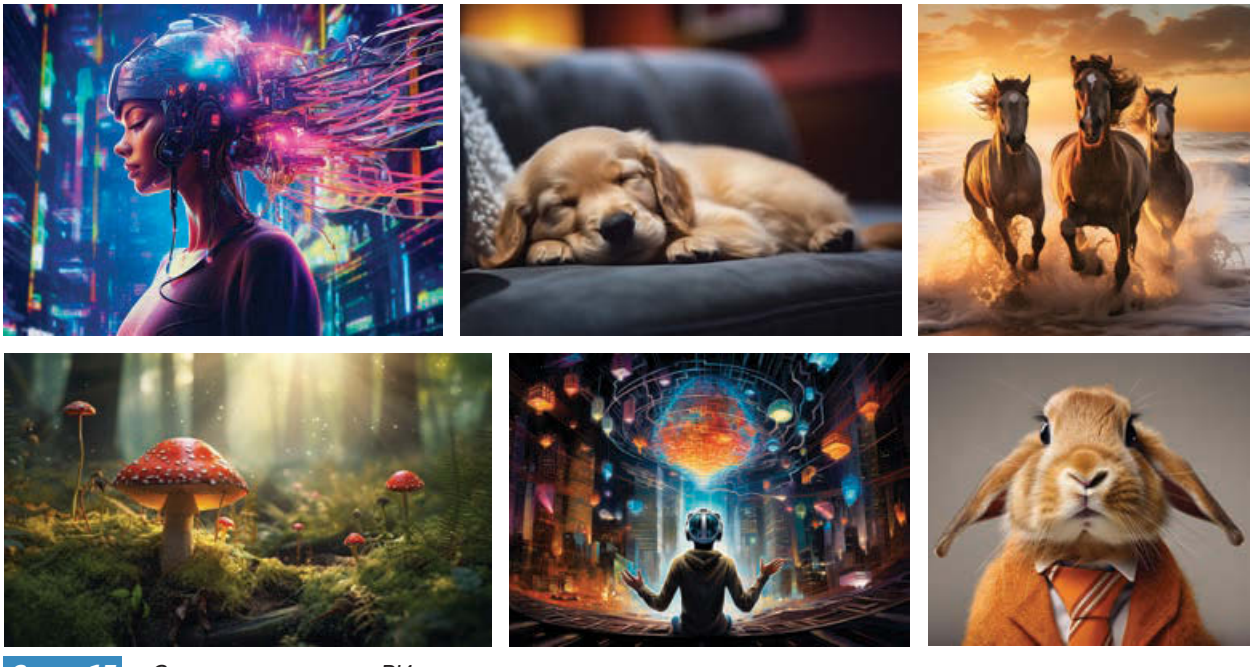
Етичка употреба според целта

- ВИ треба да се користи за доброто на луѓето – во образованието, здравството, заштита на животната средина итн.
- Не смее да се користи за лажни вести, манипулации, надзор без дозвола или воени цели против цивили.

Почитување на човековото достоинство и права

- ВИ мора да ги почитува основните човекови права, како слобода на изразување, приватност, слобода на мислење и сл.
- ВИ не смее да ги заменува луѓето на начин што ги деградира или им ја одзема вредноста.

Етичката употреба на ВИ значи одговорност, транспарентност, човечност, фер однесување и безбедност. Технологијата треба да им служи на луѓето, а не обратно.



Слика 13 *Слики ѓенерирани од ВИ*

ЗАКЛУЧОК

Вештачката интелигенција (ВИ) претставува способност на компјутерските системи да извршуваат задачи кои обично бараат човечка интелигенција — учење, препознавање слики и говор, донесување одлуки и решавање проблеми. Во секојдневието, ВИ се користи во голем број ситуации: препораки на видеа и музика, навигација, паметни асистенти како Сири и асистентот на Гугл, филтри против спам, системи за препознавање лица, автоматски преведувачи и чат-ботови. **Општата** ВИ е замислена како вид ВИ што би можела да мисли и да учи слично како човек и да се снаоѓа во многу различни ситуации. Таа сè уште не постои во реалноста, туку е само теоретска идеја.

Истражувачите се обидуваат да создадат системи што ќе разбираат контекст, ќе се прилагодуваат на нови услови и ќе решаваат различни задачи, а не само една конкретна работа. Целта е да се развие интелигенција што би можела да функционира во повеќе области – како човековиот ум.

Генеративната ВИ, пак, е способна да создава нов текст, слики, музика, видео или код врз основа на податоците со кои е обучена. Таа може да пишува есеј, да реши задача, да генерира фотографија или да создаде нови идеи и содржини.

- **Примери за општа ВИ:** автопилот во автомобили, системи за предвидување временска прогноза, медицинска анализа, спам филтри, препораки на филмови.
- **Примери за генеративна ВИ:** ЧатЏПТ, ДАЛЛ·Е, Мидџорнеи, Бард и други системи за создавање музика или 3Д модели.

Како што се зголемува примената на ВИ, големо значење имаат и етичките правила за нејзино користење. Потребно е контролирано споделување лични податоци со системите како и проверка на информациите произведени од ВИ бидејќи може да бидат погрешни. Авторството сè повеќе е значајно, копирањето туѓи идеи без дозвола е строго регулирано. ВИ треба да се користи за учење, а не за препишување.

Етичките дилеми поврзани со општа и генеративна ВИ сè повеќе добиваат на важност. Овде спаѓаат прашањата за **приватноста**, бидејќи некои системи собираат многу податоци. Дилемата за **точност** затоа што ВИ може да создава грешки или лажни информации. Ризикот од **зависност** од ВИ наместо самостојно размислување, како и прашањето дали е фер да се користат ВИ-алатки во училишни задачи ако не е дозволено. Други дилеми се: како да се спречи злоупотреба на генеративната ВИ за создавање дезинформации, лажни фотографии или аудио и дали е етички машините да го заменат човекот на некое работно место.

ПРИМЕР 1

Генерирање текст

Онлајн платформа користи генеративна ВИ за да создаде автоматски резиме на долги статии. Корисникот внесува текст, а ВИ го обработува и прикажува кратка, концизна верзија.

ПРИМЕР 2

ВИ во медицината

Систем за анализа на рендген-снимки користи невронска мрежа за да открие потенцијални заболувања на белите дробови. Дијагнозата потоа ја потврдува лекар, кој ја користи ВИ како поддршка.

ПРИМЕР 3

Виртуелен уметник

Ученик користи генеративен модел за да создаде дигитален постер базиран на опис: „фантастичен пејзаж со виолетови планини и небо со две сонца“. ВИ креира слика за неколку секунди.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Објасни со свои зборови што е вештачка интелигенција!

2. Кои се разликите помеѓу општата и генеративната ВИ?

3. Наведи три реални примери каде што се користи генеративна ВИ!

4. Што претставува поимот „невронска мрежа“ и која е нејзината функција?

5. Истражи кои генеративни апликации постојат денес и што овозможуваат!

6. Кои се етичките проблеми поврзани со генеративната ВИ?

7. Што се „deepfake“ видеа и зошто можат да бидат опасни?

8. Замисли како ќе се користи ВИ во училница во иднина. Опиши еден пример!

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи како се обучуваат генеративни јазични модели како GPT и какви податоци се користат за нивно тренирање! Проучи какво значење имаат термините етичка ВИ и ВИ транспарентност и зошто се важни за идниот развој на технологијата!

ПРАКТИЧНА ЗАДАЧА

Замисли дека твоето училиште сака да воведе ВИ-систем за:

- автоматско препознавање лица за влез во училиштето,
- следење на учениците на одмор (со камери) заради безбедност,
- автоматско оценување писмени задачи со анализа на текст.

Направи **етичка процена.**

Одговори на следниве прашања:

1. Кои етички проблеми може да се појават кај овој ВИ-систем? (пр. приватност, пристрасност, лажни резултати...)

2. Кој би можел да биде оштетен или дискриминиран и зошто?

3. Кое етичко правило е засегнато?

4. Со кои дилеми ќе се соочиш?

5. Како може училиштето да обезбеди ВИ да се користи одговорно?

6. Што мислиш – дали е добро или не да се воведе овој систем во вашето училиште? Зошто?

7. Што би направил/а за да се реши проблемот?

Користи ја табелата на сликата подолу за да дојдеш до правилно решение.

Бр.	Опис на ситуацијата / дилемата	Засегнати етички правила	Кој е проблемот? (објаснување на судирот)	Можни решенија (според етичките правила)
1				

Табела 4 *Разрешување дилема*

ПОВТОРУВАЊЕ ЗА ТЕМА 1

1. Наведи три примери за дигитална технологија што ја користиш во секојдневниот живот!

2. Објасни што претставуваат папките и датотеките и зошто се важни за организирање на податоците на компјутер!

3. Креирај нова папка на работна површина со име 'Мои фотографии' и додај една фотографија во неа! Потоа премести ја папката во твоите документи и преименувај ја по твој избор!

4. Спореди две современи дигитални технологии (на пример: паметен телефон и паметен часовник) и објасни ги разликите во нивната намена!

5. Процени кои апликации се најсоодветни за: пишување текст, правење табела и создавање презентација! Објасни зошто.

6. Осмисли кратки кориснички упатства за прилагодување на основните поставки на оперативниот систем (ажурирања — проверка и инсталација на најновите системски исправки, кориснички сметки и лозинки, постави силна лозинка на локален корисник со 2FA, поврзување на Wi-Fi мрежа, вклучување фајрвол, креирање дозволи само за неопходни апликации да имаат пристап до камера/микрофон/локација, иницијализација на антивирус и проверка на системот, оптимизација на екран за учење, јасност, контраст, дотерување на режим на мирување/исклучување на екранот и отстранување на непотребни програми и апликации).

7. Наведи две разлики меѓу општата (General AI) и генеративната вештачка интелигенција!

8. Објасни како влијае дигиталната технологија врз образованието, комуникацијата или работата во современото општество!

9. Анализирај како се разликуваат различните оперативни системи (Windows, Android, Linux) во изгледот, во работната околина и можностите!

10. Замисли дека треба да имплементираш генеративна ВИ во училиштето. Предложи три начини како би можела да се користи. Објасни како функционира и кои се можните придобивки и ризици!

ТЕМА 2: Онлајн-живеење

Денес е многу лесно да се допишуваш со пријател што живее во друга држава, да гледаш филм онлајн, да закажеш лекарски преглед преку апликација, или да играш виртуелна игра со луѓе од целиот свет — сето тоа се случува благодарение на компјутерските мрежи и интернетот.

Во денешно време, светот е поврзан како никогаш досега. Без разлика дали станува збор за мобилен телефон, лаптоп, паметен часовник или индустриски робот — сите овие уреди „разговараат" меѓусебно преку мрежи. Токму затоа, познавањето на компјутерските мрежи и интернетот не е само корисна вештина, туку е и еден од најважните чекори кон разбирање на светот околу нас.

Нови поими

- компјутерска мрежа, сервер, клиент, LAN, WAN, интернет, веб, „сурфање" на интернет, интернет-сервиси, www, ftp, машини за пребарување (search engines), електронска пошта, видеоповици, е-трговија, електронско банкарство, интерактивно комуницирање, дигитален отпечаток, насилство, омраза, промотивен материјал за користење наркотици или самоповредување, дезинформации, лажни вести, дотерувања за безбедност и приватност, правен аспект, механизми за пријавување, добросостојба.

Веќе знаеш да...

- разликуваш веб-пребарувач и веб-прелистувач;
- пристапуваш до интернет-ресурси и да пребаруваш потребни информации;
- препознаваш валидни извори на информации на веб;
- ги препознаваш позитивните и негативните страни на „дигиталниот отпечаток";
- комуницираш преку интернет.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

Патувачка авантура

Стоиш на аеродром, багажот ти е спремен, а во рака имаш магичен билет што те води до кое било место на светот. Каде ќе одбереш да заминеш? Одбери една земја или град што отсекогаш си сакал/а да го посетиш. Која е нејзината култура – јазик, традиции, храна? Како можеш да стигнеш таму (летови, цени, пат)? Колку би те чинел престој од 3 дена (најди реални примери)? Кои три работи треба да ги видиш или да ги направиш таму?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- ги објаснуваш поимите компјутерска мрежа и интернет;
- го опишуваш интернетот како средство за добивање и споделување информации;
- разликуваш различни видови интернет-сервиси (услуги);
- користиш валидни извори на информации на веб;
- ги анализираш позитивните и негативните страни на „дигиталниот отпечаток“ кој го оставаш;
- создаваш дигитална содржина од пронајдени податоци и информации.

2.1 Поим за компјутерска мрежа

Компјутерска мрежа подразбира постоење на најмалку два компјутера кои меѓусебно се поврзани за да можат да комуницираат и да разменуваат ресурси. Под ресурси се подразбираат хардверски компоненти (компјутер, сервер, скенер, печатач и други) и софтверски компоненти (игри, бази на податоци, апликации....). Компјутерите се поврзуваат во мрежа за да се зголеми ефикасноста и да се намалат трошоците (штедат време и пари). Овие уреди може да бидат поврзани преку физички кабли или преку безжична комуникација, зависно од инфраструктурата и од потребите на корисникот.

2.1.1 Намена на компјутерските мрежи

Преку компјутерските мрежи корисниците споделуваат податоци, софтвер, хардвер и комуницираат едни со други.

Споделување податоци и информации. Во мрежна околина, можно е да пристапиш до податоци од други компјутери во мрежата. Така, на пример, во една компанија, преку компјутерската мрежа, сите вработени можат да пристапат до заеднички календар со состаноци или до база на податоци со информации за клиентите. Во компјутерска мрежа поставена во домашни услови, еден член на семејството може да сподели филм или документ на својот компјутер, а другите членови – користејќи таблет или друг компјутер – да пристапат до тие содржини непречено, односно не треба да ги копираат на својот уред.

Споделување софтвер. Во компјутерската мрежа можно е да се користи софтвер што се извршува на други компјутери. На пример, во училиште, на еден компјутер (сервер) може да се инсталира програма за графички дизајн. Наместо секој ученик да ја има инсталирано програмата на својот компјутер, тие можат да пристапат до него преку училишната мрежа и да работат на свои проекти.

Споделување хардвер. Во мрежна околина можно е неколку корисници да споделуваат хардверски ресурси (на пример, сервер, печатач, скенер). Честопати во компаниите или во домовите има еден печатач со подобар квалитет кој е поврзан со неколку компјутери преку мрежа и од секој од овие компјутери е можно да се печати на тој печатач.

Комуникација. Компјутерските мрежи го променија начинот на кој луѓето комуницираат – денес луѓето најчесто комуницираат преку е-пошта, социјални (друштвени) мрежи, услуги за разговор, интернет-телефони итн. Овие мрежи овозможуваат комуникацијата да биде побрза, економски исплатлива и достапна во реално време, без разлика на физичката оддалеченост. Благодарение на мрежите, можеш да разговараш со пријатели во странство, да учествуваш во онлајн настава, да работиш на заеднички проекти со луѓе од целиот свет и да споделуваш фотографии и видеа за само неколку секунди.

2.1.2 Интернет

Интернет е глобална компјутерска мрежа која обезбедува различни информациски и комуникациски средства, составена од меѓусебно поврзани мрежи што користат стандардизирани комуникациски протоколи. Интернетот користи **клиент-сервер** архитектура.

Компјутер-сервер е специјално конфигуриран компјутер кој прима барања од клиенти и обезбедува услуги, ресурси или информации на други компјутери преку мрежа. Тој е постојано активен. Серверите се силни машини, со голема процесорска моќ, голем простор за складирање, дизајниран за сигурност и постојано работење, често во центри за податоци. Може да бидат виртуелни машини, облачни услуги или софтверски компоненти.

Примери на услуги што може да ги нуди серверот се:

- чување и споделување датотеки (фајл сервер),
- прикажување веб-страници (веб-сервер),
- испраќање и примање е-пошта (мејл-сервер),
- управување со податоци (сервер за бази на податоци).

Компјутер-клиент е уред (лаптоп, телефон, таблет, пребарувач и др.) кој иницира комуникација со серверот со цел да добие некоја информација или услуга. Пример: Кога еден корисник користи интернет-прелистувач за да посети веб-страница, неговиот компјутер е клиент, а компјутерот што ја хостира веб-страницата е сервер. Во компјутерските мрежи секогаш има двонасочна комуникација меѓу клиентот и серверот.

Клиентот (на пример, веб-прелистувачот) праќа барање до серверот, кој ја пронаоѓа и ја враќа бараната содржина. Оваа комуникација се одвива преку протоколи како HTTP, HTTPS, TCP/IP. Секој уред на интернет има IP адреса – уникатен број кој служи за идентификација и насочување на податоците.

Дополнително, овие мрежи се конфигурираат со различни поставки во зависност од потребите – на пример, броење активни уреди, контрола на пристап, приоритизација на сообраќајот и мониторинг на активностите. Во образовни институции, компјутерските мрежи овозможуваат централизирана администрација, контрола на пристап до интернет, споделување ресурси и соработка меѓу учениците и наставниците. На глобално ниво, интернетот е составен од стотици илјади меѓусебно поврзани мрежи кои користат заеднички јазик – TCP/IP протоколот – за да се разберат едни со други.

DNS (Domain Name System) ги претвора човечките имиња на веб-страници (на пр. www.wikipedia.org) во машински читливи IP адреси. Ова го прави интернетот попристапен и полесен за користење. Пакетите на податоци се испраќаат низ мрежата преку различни рути, а секој чекор се контролира од рутери и прекинувачи кои овозможуваат ефикасен пренос.

Рутерите играат централна улога во насочување на податоците преку интернет. Тие се одговорни за избор на најдобра рута за секој пакет што се испраќа, со цел брзо и сигурно пристигнување до дестинацијата. Прекумерното оптоварување или прекин на одредени рути може да предизвика доцнење или загуба на податоци. Затоа, протоколите вклучуваат механизми за проверка, повторно испраќање и корекција на грешки.

2.1.3 Видови компјутерски мрежи

За да се разбере структурата и примената на компјутерските мрежи, тие се класифицираат според нивната големина и опфатот – најчесто во две главни категории:

- LAN (Local Area Network) – локална мрежа,
- WAN (Wide Area Network) – мрежа распространета на географска област (различни населби-ист град, различни градови, и сл.)

Локална мрежа – LAN

LAN претставува локална компјутерска мрежа која овозможува меѓусебно поврзување на различни уреди – компјутери и периферни уреди во рамките на ограничена географска област, како што се домови, училници, училишта, канцеларии, или цела зграда. Може да биде едноставна (составена од два компјутера) или сложена (составена од стотина компјутери и периферни уреди во една голема компанија).

WAN компјутерска мрежа

Компјутерска мрежа распространета на географска област. Се применува за меѓусебно поврзување на оддалечени компјутери. Компјутерската мрежа обединува и поврзува помали мрежи за да можат корисниците на една мрежа да комуницираат со корисниците на друга мрежа или да користат ресурси на оддалечена локација.

Карактеристика	LAN	WAN
Покриеност	Мала област	Големи растојанија
Брзина	Брза (Gbps)	Побавна од LAN (доволно брза)
Цена	Мала	Голема (изнајмување линии, посебна инфраструктура)
Сопственост	Приватна (организација, дома, и сл.)	Јавна или приватна (инфраструктура од интернет-провајдер)
Опрема	Свичеви, рутери	Оптички линии, сателити, телеком мрежи
Управување	Едноставно	Сложено, со поддршка од провајдери
Пример	Домашна мрежа	Компанија со повеќе оддалечени сектори, интернет-мрежата

Табела 1 *Разлики помеѓу локална и глобална LAN и WAN мрежа*

ЗАБЕЛЕШКА:

Компјутерските мрежи не треба да се поистоветуваат со интернетот. На пример, се случува во училиштен кабинет компјутерите да се вмрежени и меѓусебно да разменуваат податоци, но оваа мрежа да не е поврзана со интернет.

ЗАКЛУЧОК

Компјутерските мрежи овозможуваат поврзување на повеќе компјутери и уреди со цел споделување информации, ресурси и услуги. Серверот обезбедува услуги, податоци или ресурси, додека клиентот ги користи тие услуги преку испраќање барања. Разликата меѓу локалните и глобалните мрежи покажува колку широко можат да бидат поврзани уредите — од мала училница до целиот свет.

ПРИМЕР 1

LAN мрежа во училиште

Во едно училиште, сите училници се поврзани преку локална мрежа (LAN) која овозможува пристап до интернет, централен печатач и заедничка база на податоци со наставни материјали.

ПРИМЕР 2

DNS и пребарување веб-страница

Кога ученик ќе внесе „www.wikipedia.org“ во веб-прелистувач, DNS серверот ја претвора оваа адреса во IP број и ја пренасочува до соодветниот веб-сервер каде што се наоѓа содржината.

ПРИМЕР 3

Користење Wi-Fi во домот

Во домашни услови, уредите се поврзуваат безжично преку Wi-Fi рутер. Тој го насочува интернет-сообраќајот меѓу мобилниот телефон, лаптопот и надворешната интернет-конекција.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Објасни ги поимите компјутерска мрежа, компјутер сервер и клиент!

2. Што е разликата меѓу LAN и WAN мрежа?

3. Наведи два примера за сервери што се користат во секојдневниот живот!

4. Користиш интернет-прелистувач за да отвориш веб-страница. Кој уред е клиент, а кој е сервер во таа ситуација?

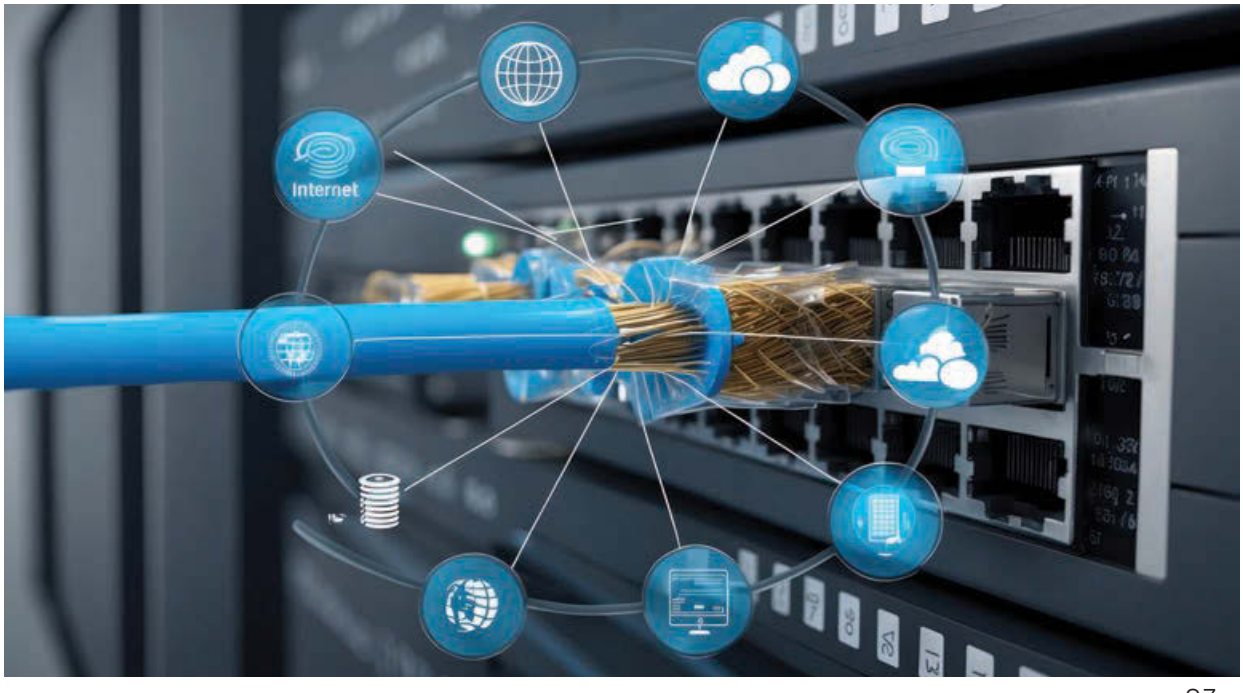
5. Зошто е важно серверите да бидат постојано активни, а клиентите **не**?

6. Дали секој компјутер може да стане сервер? Образложи го твоето мислење!

7. Осмисли едноставна мрежа за едно училиште! Опиши како би ги организирал/а серверите и клиентите!

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи што се случува во заднина кога ќе внесеш веб-адреса во прелистувач и притиснеш Enter. Обиди се да направиш мапа на процесот на пренос на податоци од твојот уред до серверот и назад. Дополнително, разгледај какви кариери може да се градат во областите поврзани со мрежи – како мрежен администратор, инженер за кибербезбедност или ИТ поддршка.



2.2 Интернет-сервиси

Интернетот овозможува пристап до огромен спектар на услуги што го олеснуваат секојдневниот живот. Овие интернет-сервиси се користат за комуникација, информирање, образование, купување, банкарство и многу повеќе. Разбирањето на нивната структура, намена и функционирање е клучно за правилно, безбедно и ефикасно користење. Во оваа лекција ќе се објаснат најважните интернет-сервиси, како функционираат и кои се нивните предности и ризици.

WWW

Најчесто користен интернет-сервис е World Wide Web (WWW). Тој овозможува прегледување на веб-страници со текст, слики, видео и интерактивни содржини. До веб-страниците се пристапува преку веб-прелистувачи (на пример, Chrome, Firefox, Edge), а содржините се пренесуваат со користење на HTTP и HTTPS протоколи.

Веб-страниците се наоѓаат на сервери (веб-сервери) и претставуваат група поврзани веб-страници. Секоја локација на веб има единствена адреса таканаречена URL (Uniform Resource Locator) адреса. WWW е корисен за учење, истражување, забава и комуникација преку форуми, портали и социјални мрежи. Овие интернет-сервиси се развиваат со голема брзина, при што постојано се додаваат нови функционалности за да се подобри корисничкото искуство. Корисниците треба постојано да се информираат за новите можности и начини на безбедна употреба, особено во случаи кога се бара внесување на лични или финансиски податоци.

Важно: WWW не е исто што и интернет – тоа е услуга што работи преку интернет. Додека интернетот претставува мрежа од поврзани компјутери, WWW е збир од апликации или дигитални содржини до кои пристапуваме преку таа мрежа.

Машини за пребарување (search engines)

Денес на интернет постојат милиони веб-страници, што го прави нивното пронаоѓање понекогаш сложено и долготрајно. Интернетот стана огромен, непрегледен и практично неисцрпен извор на информации. За полесно пребарување на содржините, можеш да користиш веб-директориуми (кои ги групираат страниците според тематски области) или, што е почеста практика, да пребаруваш со помош на пребарувачки машини (search engines).

Со внесување на барање во веб-пребарувач (Google, Edge, Yahoo, Bing) составено од клучни зборови, се активира алгоритам за пребарување на интернет. Како одговор на тоа барање, пребарувачот прикажува список со веб-страници кои ги содржат клучните зборови од барањето. Секоја од наведените веб-страници, чии адреси (и кратки описи) се прикажани како резултат на пребарувањето, може да ги посетиш со клик на името на таа страница во резултатите прикажани од пребарувачот.

Електронска пошта

Електронската пошта е еден од најстарите и најкористени сервиси на интернет, кој овозможува размена на текстуални пораки и различни видови датотеки помеѓу корисници преку компјутерски мрежи. Претставува дигитален облик на традиционалната пошта, но со многу поголема брзина, удобност и достапност.

За да може да се користи електронска пошта, секој корисник треба да има своја електронска адреса. E-mail адресата ја избира корисникот, а му ја одобрува e-mail серверот.

За секоја електронска адреса се доделува **електронско сандаче** во кое се чуваат електронските пораки на корисникот. E-mail сервер е компјутер поврзан на интернет кој ги прима и ги испраќа електронските пораки на корисниците на електронската пошта.

E-mail адресата се состои од два дела: корисничко име и име на e-mail серверот. Овие два дела се одделени со знакот @ ("at" кој се чита ет, го нарекуваат и мајмунче). Корисничкото име го избира корисникот при креирање на e-mail адресата ако е слободно (не е веќе избрано од некој друг корисник на тој mail - сервер). Името по знакот @ е името на e-mail серверот преку кој се одвива електронската пошта на корисникот. На пример, во e-mail адресата maja_matevska@t-home.mk корисничкото име е maja_matevska, а името на серверот е t-home.mk.

При работа со електронска пошта e-mail адресите со мали и големи букви се исти.

На пример: biljana@gmail.com=BILJANA@GMAIL.COM.

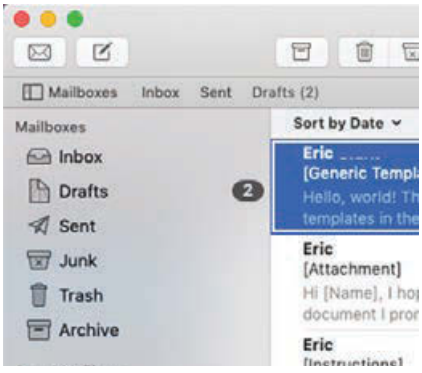
Структура на електронска порака:

- адреса на испраќачот (**From**): автоматски се пополнува,
- адреса на примачот (**To**): се пишува во адресниот дел,
- **Carbon Copy** (**cc**): се внесува меил-адресата на друго лице кое ќе прими копија од пораката, а примачот гледа дека пораката му е испратена и на друг,
- **Blind carbon copy** (**bcc**): примачот на пораката не знае дека копија од пораката се испраќа и на лица чија адреса е наведена во bcc полето,
- наслов на пораката – тема (**Subject**): краток опис на содржината. Примачот на пораката, освен името (адресата) на испраќачот и датумот, го гледа и предметот на пораката. Врз основа на тие податоци примачот на пораката решава дали и кога ќе ја прочита пораката,
- текст на пораката: се пишува во делот за текст порака. Текстот на пораката зависи од тоа на кого му е наменет прилогот (**Attachment**): фајл кој се додава на пораката. Може да биде документ, слика, презентација, програма...
- потпис (**Signature**).

Слика 1 *Електронска пошта*

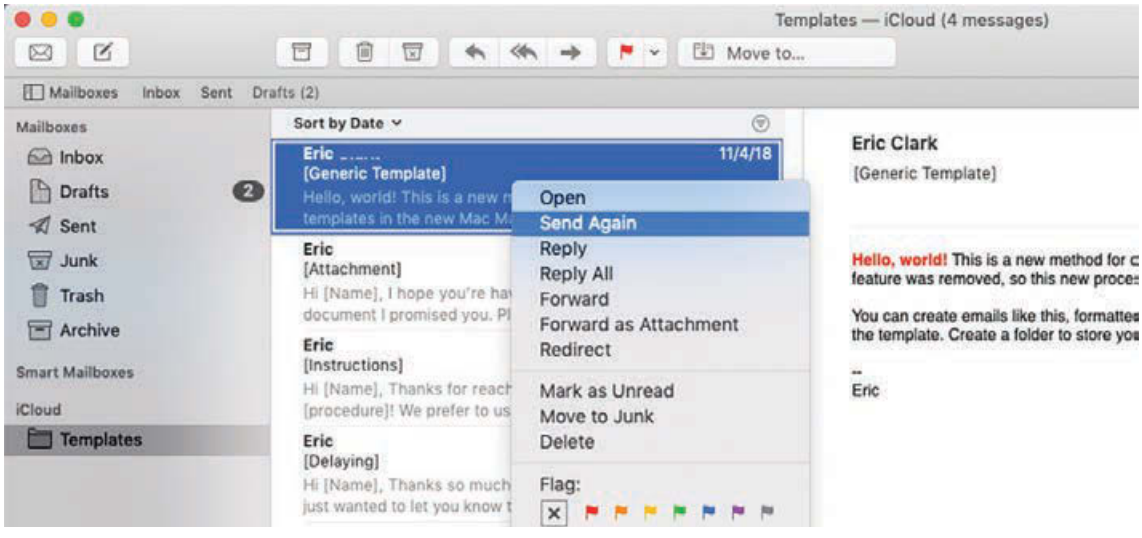
Најчесто сандачињата за електронска пошта содржат стандарден број на директориуми (папки) како што се:

- **Inbox** (папка со примени пораки, наречена поштенско сандаче),
- **Sent** (папка со испратени пораки),
- **Trash** (или **Deleted**, папка со избришани пораки),
- **Drafts** (папка за неиспратените пораки во правец на уредување).



Слика 2 *Структура на електронско сандаче*

Сите програми за работа со е-пошта овозможуваат организирање на е-поштата. Корисниците можат да креираат нови папки, да преместуваат и копираат пораки од една папка во друга и да дефинираат правила врз основа на кои пораките ќе бидат сортирани во одредени папки веднаш по приемот.



Слика 3 *Целосна структура на електронско сандаче*

Програмите за електронска пошта ги имаат следниве функции:

- Креирање електронска порака – Нова порака (New);
- Испраќање порака – Испрати (Send);
- Одговарање на примена порака – Одговор (Reply);
- Препраќање на примена порака кон друга личност – Препрати (Forward);
- Бришење на пораката – Избриши (Delete);
- Печатење на пораката – Печати (Print);
- Вклучување на други документи во пораката – Закачи (Attach);
- Користење адресар.

ftp

FTP е кратенка од File Transfer Protocol, што значи протокол за пренос на датотеки. Тоа е еден од најстарите интернет-сервиси кој овозможува испраќање и преземање датотеки меѓу два компјутера преку интернет – најчесто од клиент (твојот компјутер) до сервер и обратно.

Видеоповици

Видеоповици се начин на комуникација преку интернет каде што можеме да се гледаме и слушаме во живо со некој друг. За таа цел ти е потребно:

- уред – компјутер, лаптоп, таблет или мобилен телефон;
- камера (вградена или надворешна) – за слика;
- микрофон и звучници (или слушалки) – за звук;
- интернет-конекција – подобра конекција значи поквалитетен повик;
- апликација или платформа за видеоповици.

Видеоповиците се дел од нашето секојдневие. Можеш да ги користиш за онлајн настава, деловни состаноци, семејни разговори, онлајн интервјуа и обуки и слично. Популарни сервиси за видеоповици се: Zoom, Microsoft Teams, Google Meet, Viber, WhatsApp.

Други сервиси кои сè повеќе се користат се облачни услуги (Cloud Services) кои овозможуваат снимање податоци на интернет-сервери и пристапување од каде било, како што се: Google Drive, OneDrive, Dropbox, потоа стриминг услуги (Streaming Services) за гледање видеа, слушање музика или пренос во живо преку интернет. Најпознати се YouTube, Netflix, Spotify и Twitch.

Електронска трговија

Во денешниот дигитален свет, благодарение на развојот на интернетот и модерните технологии, сè повеќе луѓе избираат да купуваат производи и услуги онлајн, со само неколку кликнувања на своите уреди. Оваа форма на трговија се нарекува електронска трговија (или е-трговија).

Електронската трговија овозможува комуникација и размена на производи помеѓу купувачи и продавачи преку интернет, без разлика на физичката оддалеченост. Безбедните веб-страници, онлајн продавниците, мобилните апликации и електронските плаќања се само дел од алатките што ја прават оваа трговија брза, практична и достапна. Овој вид трговија вклучува:

- продажба од страна на компаниите на стоки и услуги на индивидуални клиенти – B2C (business to customer),
- продажба од страна на компаниите на стоки и услуги на други компании – B2B (business to business),
- продажба од страна на продавачите кои поединечно ги продаваат своите стоки и услуги на индивидуални клиенти – C2C (customer to customer).

Денес, многу страници овозможуваат купување и продажба на разни производи. Купувањето се реализира така што купувачот ги плаќа стоките и услугите со кредитна картичка, готовина при испорака или со стандардна уплатница. Продавачот ја доставува стоката по пошта или преку курирска служба. Повеќето светски даватели на услуги овозможуваат електронска трговија. На пример, денешното патување може целосно да се организира преку интернет. Авионски, автобуски или билети за воз најлесно се купуваат преку веб-страниците на транспортните компании. Корисникот добива електронски билет кој може да го користи и во електронска и во печатена форма (не се препорачува) и да патува со авион, автобус или воз со својот пасош. Хотелите и хостелите може да се резервираат и преку нивните веб-страници или уште побезбедно, преку добро позната веб-страница специјализирана за вакви услуги.

При користење услуги за е-трговија, особено внимание треба да се посвети на безбедноста. Со оглед на тоа што станува збор за трошење вистински пари, невнимателноста може да има многу непријатни последици.

Електронско банкарство

Денес, банките нудат услуги за електронско банкарство. Електронско банкарство (или е-банкарство) претставува користење на интернет и дигитални уреди за да се пристапи до банкарски услуги. Тоа овозможува корисниците да управуваат со своите финансии преку компјутер, мобилен телефон или банкомат, без потреба да одат физички во експозитура.

Откако ќе се најавиш во системот, можеш да ја провериш состојбата на твојата сметка, да вршиш уплати, исплати и да префрлуваш средства од сметка на сметка. Услугата за електронско плаќање сметки е многу корисна бидејќи можеш сè да платиш од дома и не треба да чекаш во пошта или во банка. Безбедноста е клучна, па банките користат заштитни системи како шифрирање, двофакторска автентикација и мобилни токени за заштита на податоците на корисниците.

Интерактивно комуницирање

Интернет-сервисите овозможуваат интерактивно комуницирање. Освен разговорите во живо (Chat), видеоповици и видеоконференции, интернет-комуникацијата вклучува и форуми и онлајн дискусии како и социјални (друштвени) мрежи.

Форумите се веб-страници што им овозможуваат на корисниците да дискутираат (обично само на теми утврдени со правилата на форумот). Дискусијата е организирана во теми – некој започнува нишка на одредена тема со објавување на првата порака, а понатамошната дискусија на таа тема продолжува со одговарање на пораките од таа нишка.

Блоговите (weB LOG – блог) се еден вид лични дневници во кои корисниците ги објавуваат своите мисли за одредена тема и ги споделуваат со јавноста. Корисниците читаат статии објавени од автори на блогови и можат јавно да коментираат и да дискутираат за нив.

Социјалните (друштвените) мрежи се создадени во средината на првата деценија од 21 век и многу брзо станаа еден вид социолошки феномен. За помалку од една деценија, бројот на активни корисници порасна на неколку милијарди. Социјалните мрежи им овозможуваат на корисниците да се поврзат и да воспостават контакт со сметките на нивните лични пријатели и познаници или со сметките на личности од јавната сфера. Поединечните социјални мрежи имаат свои специфики:

Facebook – најголема социјална мрежа што овозможува споделување статуси, фотографии, видеа и поврзување со пријатели, групи и страници.	**Instagram** – платформа за споделување фотографии и кратки видеа, популарна меѓу младите. Се користи и од инфлуенсери и бизниси.	**TikTok** – мрежа за кратки, креативни видеа со музика, ефекти и предизвици.
Twitter (cera X) – платформа за кратки текстуални објави („твитови“), идеална за вести, мислења и брза комуникација.	**Snapchat** – апликација за испраќање фотографии и видеа кои исчезнуваат по гледањето. Популарна кај тинејџерите.	**LinkedIn** – професионална мрежа за вмрежување, барање работа и деловна комуникација.

YouTube – иако е видеоплатформа, има и функции на социјална мрежа преку коментари, објави и заедници.	**Pinterest** – мрежа за споделување инспиративни слики и идеи, најчесто поврзани со мода, уметност, храна, дизајн итн.	**Reddit** – форумска платформа каде што луѓето дискутираат за различни теми, поставуваат прашања и споделуваат мислења.

Важен аспект на социјалните мрежи е приватноста на корисниците, а при поставување статуси (на пример, фотографии од вашите најблиски), треба да се има предвид дека тие стануваат видливи за широк круг луѓе. Ако не сакаш твоите информации да бидат јавно видливи, подобро е да ги поставиш така што нема да може да ги види никој освен твоите пријатели. Бројот на корисници на социјалните мрежи докажува дека тие се исклучително лесни за користење.

Интернетот е одлично место за учење, истражување, дружење и забава. Но, како и во реалниот свет и таму постојат ризици. Затоа е важно да ги почитуваш правилата за безбедна комуникација, за да се заштитиш себеси и другите. Тоа вклучува никогаш да не споделуваш лични податоци со непознати лица, да избегнуваш кликање на сомнителни линкови и да се однесуваш учтиво и одговорно.

При креирањето на онлајн сметка (на пр. за е-пошта, училиштен портал или апликација) треба да си внимателен и да водиш сметка за својата безбедност. При регистрација, многу веб-страници бараат корисничко име и лозинка. Во овој чекор, најважно е да избереш безбедна лозинка – таа треба да биде доволно долга (најмалку 12 знаци) и да содржи комбинација од мали и големи букви, броеви и специјални знаци (на пр. @, #, $, %). Не е препорачливо да користиш информации кои лесно може да бидат откриени, како датум на раѓање или име на домашно милениче. Исто така, важно е да не ја споделуваш лозинката со други лица. Препорачливо е лозинката редовно да ја менуваш и да користиш различни лозинки за различни профили. На овој начин се намалува ризикот од злоупотреба на податоците.

Безбедната комуникација е темел на одговорно користење на интернетот и учење во безбедно онлајн опкружување.

ЗАКЛУЧОК

Интернет-сервисите претставуваат основа на современото дигитално општество, овозможувајќи брз и едноставен пристап до информации, комуникација и различни услуги. Тие го трансформираат нашето секојдневие, подобрувајќи ги начините на работа, учење, забава и социјална интеракција. Со постојаниот развој на технологиите, интернет-сервисите стануваат сè повеќе интегрирани и незаменливи во современиот живот, овозможувајќи глобална поврзаност и иновации кои ја обликуваат иднината.

ПРИМЕР 1

Користење е-пошта за училишни цели

Еден ученик/чка добива домашна задача преку е-пошта од својот наставник. Ја отвора пораката, го презема прикачениот документ, ја решава задачата и ја испраќа назад со нов прилог. Овој процес се одвива брзо и ефикасно, без физичка средба.

ПРИМЕР 2

Онлајн купување преку е-трговија

Корисник/чка сака да купи слушалки преку интернет. Го посетува веб-сајтот на продавницата, го избира производот, ги внесува податоците за испорака и плаќа преку интернет банкарство. Наскоро добива потврда и број за следење на пратката.

ПРИМЕР 3

Користење на облак услуга

Наставничка ги чува сите свои наставни материјали на Google Drive. Таа креира заедничка папка со учениците, кои можат да ги прегледуваат документите, да коментираат и да уредуваат проекти во реално време од кој било уред.

ПРАШАЊА НА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Што се интернет-сервиси и која е нивната главна улога?

2. Кои се најчестите видови интернет-сервиси што ги користиш во секојдневниот живот?

3. Како влијаат интернет-сервисите врз комуникацијата меѓу луѓето?

4. На кои начини интернет-сервисите го олеснуваат учењето и образованието?

5. Кои се најголемите предности на користењето интернет-сервиси?

6. Какви безбедносни ризици може да носат интернет-сервисите?

7. Како влијае технологијата врз развојот на интернет-сервисите?

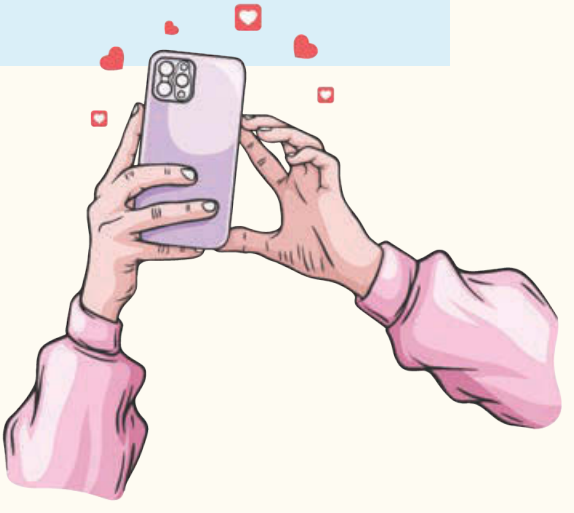
8. Зошто е важно да се има пристап до интернет-сервисите во современото општество?

9. Кои примери за интернет-сервиси можеш да ги набројиш?

10. Како придонесуваат интернет-сервисите за глобалната поврзаност?

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи како функционираат системите за стриминг (како YouTube и Netflix) – од складирање на податоци до пренос на видеосодржини до твојот уред. Дополнително, разгледај како се користат интернет-сервисите во едукација, на пример, преку Moodle, Google Classroom или Microsoft Teams, и кои се нивните предности за учење на далечина.



2.3 Развој на веб-технологии

Интернетот постојано се развива, а секоја негова фаза претставува нова генерација на технологии и начини на користење на веб-технологиите. Кога зборуваме за еволуцијата и иновациите на интернетот, најчесто се користат поимите Web 1.0, Web 2.0, Web 3.0, Web 4.0 и Web 5.0 – термини што го опишуваат патот низ кој се менувал интернетот и продолжува да се менува. Овие фази не се одделени со јасни граници, туку се надоврзуваат една на друга.

Web 1.0 е првата генерација на веб, која доминира во 1990-тите и во почетокот на 2000-тите. Во оваа фаза, веб-страниците се статички, што значи дека содржината е фиксна и не може да се менува од страна на корисниците. Во оваа фаза има малку или речиси никаква интеракција со корисниците, кои се пасивни набљудувачи. Во оваа фаза веб-страницата е еднонасочна – од креаторот кон корисникот. Примери од оваа фаза се енциклопедиите како Britannica.com кои содржеле текст без можност за коментари или уредување, како и првите верзии на Yahoo! – само листа од линкови и информации, без интеракција.

Со доаѓањето на Web 2.0, интернетот почнува да се движи кон двонасочна комуникација – од корисник кон веб, но и обратно. Сега интернетот е динамичен, интерактивен и со можност за соработка. Корисниците веќе не се само читатели, туку и создавачи на содржини. Примери од оваа фаза на развојот на веб се: социјалните мрежи (платформи каде што секој може да објавува, коментира, споделува и комуницира), Wikipedia (енциклопедија што секој може да ја уредува), Blogspot и WordPress (платформи за блогирање каде што луѓето споделуваат мислења, приказни и совети). Оваа фаза донесе голем напредок, но и нови предизвици – како злоупотреба на податоци и дезинформации.

Web 3.0 ја претставува следната голема промена во начинот на кој се комуницира, се споделуваат информации и се користат дигитални услуги. Оваа нова фаза се разликува од претходните генерации по тоа што овозможува децентрализација, поголема приватност и автономија на корисниците. Наместо податоците да бидат складирани на централизирани сервери, тие се распределени преку блокчејн технологии (едноставно кажано – систем во кој сите имаат копија од податоците и никој не може да ги промени без сите да забележат). Тоа значи дека наместо да се чуваат податоците на еден централен сервер (како банка или компанија), тие се распределени меѓу многу компјутери низ светот. Оваа фаза вклучува и вештачка интелигенција и е обид вебот да се направи „паметен“. Во оваа фаза се вклучени гласовните асистенти како Siri, Alexa и Google Assistant кои, иако се централизирани, бидејќи нивните податоци и функционалности се контролирани од големи компании (Apple, Amazon, Google), сепак, користат вештачка интелигенција (ВИ) и машинско учење за подобро разбирање на корисничките барања. Иднината на Web 3.0 може да доведе до подемократски ВИ асистенти кои нема да зависат од големи компании. Пример за оваа фаза е – Brave Browser, веб-прелистувач кој не ги следи корисниците и ги наградува. Web 3.0 е „семантички веб“ – не само податоци, туку податоци со значење.

Web 4.0 претставува следна еволуција во која интернетот не е само паметен, туку и предвидлив и автономен. Оваа генерација се базира на интернетот на нештата (IoT), каде што уредите комуницираат меѓусебно и самостојно донесуваат одлуки, често без човекова интервенција. Web 4.0 вклучува интернет кој „размислува" (анализира податоци и предлага најрелевантни информации уште пред да биде прашан), природна комуникација (комуникација со гласовни асистенти и виртуелни помошници како со вистински човек) и уреди (паметни фрижидери, автомобили, часовници) кои ќе комуницираат меѓусебно и ќе помагаат во секојдневниот живот. На пример, твојот часовник може да му каже на автомобилот да се загрее пред да излезеш. Како примери од оваа фаза може да се посочат паметните домови – системи кои автоматски ги регулираат осветлувањето, температурата или безбедноста, како и автомобилите тесла кои учат и возат самостојно со помош на вештачка интелигенција.

Во претходните фази на интернетот, корисниците главно пребаруваа информации (Web 1.0), комуницираа и создаваа содржина (Web 2.0), добија попаметни системи и децентрализирани платформи (Web 3.0 и Web 4.0). Но, Web 5.0 оди чекор понатаму — тој ги разбира човечките емоции и овозможува целосна контрола врз нашиот дигитален идентитет. Иако сè уште е во рана фаза на развој, Web 5.0 претставува визија за интернет што не само што „мисли", туку и „чувствува". Станува збор за фаза во која вебот не е само корисен туку е и емпатичен. Пример од оваа фаза е виртуелен асистент кој препознава кога си тажен и нуди совети и помош, дури и соодветна музика за подобро расположение.

ЗАКЛУЧОК

Интернет-сервисите претставуваат основа на современото дигитално општество, овозможувајќи брз и едноставен пристап до информации, комуникација и различни услуги. Тие го трансформираат нашето секојдневие, подобрувајќи ги начините на работа, учење, забава и социјална интеракција. Со постојаниот развој на технологиите, интернет-сервисите стануваат сè повеќе интегрирани и незаменливи во современиот живот, овозможувајќи глобална поврзаност и иновации кои ја обликуваат иднината.

ПРИМЕР 1

Статична веб-страница од Web 1.0

Една училишна веб-страница од 1999 година прикажува само информација за распоред на часовите, без можност за интеракција или коментари. Содржината е статична и се менува ретко.

ПРИМЕР 2

Социјална мрежа во Web 2.0

Учениците користат Инстаграм за споделување фотографии и комуникација. Тие создаваат содржини, коментираат, следат други и добиваат повратни информации од заедницата.

ПРИМЕР 3

Гласовен асистент во Web 4.0

Ученик користи Google Assistant за да постави аларм, пребарува дефиниција или сака да закаже состанок. Асистентот го разбира гласот и дава персонализиран одговор врз основа на навиките на корисникот.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Кои се главните карактеристики на секоја од фазите на развој на веб-технологиите?

2. Наведи примери за секоја од фазите на развој на веб-технологиите!

3. Што е „корисничко генерирана содржина“ и во која веб-генерација се појавува?

4. Што значи „семантички веб“ и со која генерација се поврзува?

5. Дали можеш да наведеш уред од секојдневниот живот што е дел од Web 4.0?

6. Дали постои јасна граница меѓу Web 2.0, Web 3.0 и следните генерации? Зошто?

7. Како се менува улогата на корисникот низ Web 1.0 до Web 5.0?

8. За која од веб-генерациите мислиш дека најмногу го промени начинот на живеење? Објасни зошто!

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи како се користи блокчеин (blockchain) технологијата во Web 3.0 за обезбедување транспарентност и безбедност. Дополнително, разгледај етички аспекти на употребата на вештачка интелигенција во Web 4.0 и Web 5.0 – како што се дигитална добросостојба, пристап до технологии и заштита на менталното здравје.

2.4 Веб како извор на информации

Интернетот е огромен извор на информации кои доаѓаат од различни извори како, на пример, владини институции, универзитети, новинарски куќи, блогови, форумски дискусии, социјални мрежи. Дел од овие извори се научни и проверени, а други се неофицијални и можеби неточни. Како ќе знаеш кои извори се кредибилни и веродостојни, односно како ќе знаеш дали информациите што ги читаш се точни и сигурни?

За да оцениш дали една веб-страница е кредибилна, постави си ги следниве прашања:

- Кој ја објавил информацијата? Дали информацијата доаѓа од познат и сигурен извор, како училиште, универзитет, државна институција или познат медиум?
- Дали има автор? Дали е наведено името на авторот и што работи (дали е новинар, научник или експерт)?
- Кога е објавен текстот? Дали е напишан неодамна или е многу стар? Дали информациите се сè уште важечки и точни?
- Од каде се преземени фактите во текстот? Дали се споменати извори или линкови до други страници што го потврдуваат тоа што е напишано?
- Како е напишан текстот? Дали текстот има пристрасен тон – се обидува да те убеди во нешто со силни емоции или пристрасност, или е објективен и фактографски?

Кога ќе утврдиш дека некоја веб-страница изгледа безбедна и сигурна, следниот чекор е да провериш дали и самата информација што ја читаш е точна. За да го направиш тоа, треба да:

- ја споредиш со други извори – проверуваш дали и други страници ја потврдуваат информацијата.
- бараш факти и докази – дали има објаснувања, бројки или научни податоци што ја поткрепуваат информацијата.
- бидеш исклучително внимателен на измами – внимаваш на лажни вести, дезинформации и наслови што сакаат само да те натераат да кликнеш (т.н. „clickbait“).

За успешно пребарување на интернет, покрај техничките аспекти на пребарување, многу е важно да се поттикне медиумската писменост кај корисниците – односно способноста да се разликуваат факти од мислења, вистинити од лажни информации, и објективни извештаи од пристрасни интерпретации. Критичко вреднување на извори подразбира не само процена на технички параметри, туку и контекстуална анализа: кој го објавува текстот, со каква намера и дали постојат алтернативни верзии на истата информација. Информациски манипулации, дезинформации и „лажни вести“ често се ширени преку социјални мрежи и сомнителни веб-страници. Затоа, се препорачува употреба на проверени медиуми, проверка на факти преку платформи како Snopes, FactCheck.org или локални извори за верификација, како и консултација на повеќе извори при анализирање на одреден настан. Учениците треба да развијат навика да не ја прифаќаат првата информација што ќе ја пронајдат, туку да ја споредат со други извори и да проценат дали има логичка последователност, поддршка од реномирани автори или институции, и дали изнесените податоци се проверливи.

Веб-пребарувањето претставува основна вештина во дигиталната ера, но самото пребарување не е доволно без соодветна процена на изворите. Потребно е да се применува критичко размислување при анализирање на содржината, нејзиниот автор, структурата, и контекстот. Со примена на напредни техники за пребарување и процена на кредибилитетот, се добиваат посигурни информации кои може да се користат за учење, истражување и информирање. Само со ваков пристап може да се избегнат манипулации, дезинформации и погрешни заклучоци што може да влијаат врз донесувањето одлуки.

ЗАКЛУЧОК

Интернетот е моќна алатка за учење, но само ако знаеш како да ги препознаеш вистинските и сигурни информации. Со внимателна процена на изворите и споредување на информациите, можеш да се заштитиш од лаги и измами и да донесуваш правилни заклучоци. Користи го вебот не само како пребарувач, туку како критичен мислител.

ПРИМЕР 1

Академско пребарување

Ученик/чка подготвува презентација за темата „Климатски промени“. Наместо да ги користи првите резултати, пребарува со помош на наводници („климатски промени во Македонија“) и додава site:.gov.mk за да добие официјални извори.

ПРИМЕР 2

Проверка на веродостојноста

Корисник наидува на вест на социјална мрежа која изгледа сензационалистички. Тој ја проверува содржината на независни медиуми и користи платформа за проверка на факти за да открие дали е веста точна.

ПРИМЕР 3

Филтрирање резултати

Наставник пребарува PDF материјали за дигитална писменост. Тој користи Google со операторите: filetype:pdf intitle: „дигитална писменост“ site:.edu.mk, за да добие точни и релевантни документи.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Што значи интернетот да се користи како извор на информации?

2. Што е кредибилен (сигурен) извор? Дај пример!

3. Како можеш да препознаеш дали е сигурен еден веб-сајт?

4. Зошто е важно да знаеш кој ја напишал информацијата?

5. Што значи информацијата да биде неутрална? Како ќе препознаеш пристрасен текст?

6. Што е „clickbait“ и зошто треба да внимаваш?

7. Кои се трите најважни прашања што треба да си ги поставиш кога проверуваш веб-информации?

8. Што треба да направиш ако најдеш различни информации на две веб-страници?

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи како функционираат пребарувачите од технички аспект – што е web crawler, како се индексираат веб-страници и кои алгоритми се користат за рангирање. Дополнително, разгледај го концептот на „информациски меур“ (filter bubble) и како влијае врз резултатите што ги добиваме кога пребаруваме на интернет.

2.5 Дигитален отпечаток

Секогаш кога користиш интернет – кога пребаруваш нешто на Google, објавуваш фотографија на Инстаграм, гледаш видео на Јутјуб или дури само кога кликнуваш на некој линк – оставаш трага зад себе. Таа трага се нарекува дигитален отпечаток. Твојот дигитален отпечаток е твојата дигитална историја – и може да каже многу за тебе.

Дигиталниот отпечаток може подетално да се објасни како збир од сите информации што ги остава една личност на интернет – било свесно (како што се профили на социјални мрежи, објави, фотографии), било несвесно, кога системот автоматски собира информации без ние да внесеме нешто конкретно (како податоци за локација, пребарувања, колачиња, или активност на апликации). Тоа значи дека ти не само што активно можеш да се прикажеш себеси во дигиталниот простор, туку и пасивно оставаш податоци зад себе – како што се т.н. колачиња (cookies) што ги следат твоите навики на интернет.

Дигиталниот отпечаток има свои позитивни и негативни страни. Позитивни страни на дигиталниот отпечаток:

- Помага при пристап до услуги, како што се онлајн купување и банкарство.
- Може да помогне во градењето позитивна слика за тебе – позитивниот отпечаток може да ти помогне да добиеш стипендија, пракса или работа затоа што многу универзитети, организации и компании проверуваат што има за некого на интернет.
- Интернетот ги памети твоите интереси и ти нуди релевантни содржини.
- Овозможува комуникација и градење мрежи со луѓе со слични интереси со кои можеш да соработуваш и од кои можеш да учиш.

Негативни страни на дигиталниот отпечаток:

- Твоите податоци може да бидат злоупотребени или следени без твоја согласност.
- Сè што објавуваш на интернет може да остане достапно засекогаш. Фотографии, коментари, видеа или објави што некогаш ти изгледале забавно, подоцна може да те прикажат во лошо светло.
- Непромислени објави можат да влијаат на сликата за тебе во иднина. Непримерна содржина, навредливи изјави или непочитување на други може да влијаат негативно врз твојот углед.

Одговорното користење на интернетот и внимателното одбирање на тоа што ќе споделиш, може да ти помогне да создадеш позитивен и силен дигитален отпечаток кој ќе биде корисен и за твоето образование и за твојата идна кариера. Затоа:

- Пред да споделиш нешто, прашај се: „Би сакал ли моите родители, наставници или идниот работодавец да го видат ова?„ Дали ова може да ми наштети во иднина?“
- Провери ги поставките за приватност на твоите апликации и профили. Не треба целиот свет да знае сè за тебе.
- Не споделувај лични податоци (како адреса, телефон, лозинки, податоци за картички) – ниту свои, ниту на другите.
- Внимавај кого прифаќаш за пријател. Не секој што изгледа пријателски онлајн, има добри намери.
- Изгради добар дигитален идентитет. Објавувај содржини што ги покажуваат твоите интереси, проекти, идеи.

Покрај свесниот и несвесниот начин на создавање дигитален отпечаток, важно е да се разбере како се користат овие информации. Компаниите ги користат податоците за таргетиран маркетинг – тоа значи дека на корисникот му се прикажуваат реклами според неговите интереси, претходни пребарувања или локација. На пример, доколку корисник пребарува обувки, веќе следниот ден може да добива реклами за чевли на повеќе веб-страници. Ова е резултат на алгоритми кои ја анализираат активноста и ги доставуваат релевантните содржини. Сепак, иако персонализацијата може да биде корисна, таа отвора прашања за надзор и дигитална слобода. Во многу случаи, корисниците не се информирани како, од кого и до кој степен се следени. Најчесто, овие процеси се наведени во долги политики за приватност кои ретко се читаат. Со текот на времето, се создава комплексен дигитален профил кој може да се продаде на трети страни или да се користи за политичко таргетирање, манипулација со мислење или комерцијални активности надвор од контрола на самиот корисник. Овие ризици ја нагласуваат потребата од едукација, свесност и користење алатки за заштита на приватноста, како блокатори за следење, VPN, енкриптирана комуникација и чести проверки на дозволи на апликации.

Уште една значајна компонента на личната приватност е начинот на кој се чуваат и се обработуваат податоците. Платформите собираат огромни количини информации – вклучувајќи лични документи, фотографии, дописи, локации, уреди што се користат и обрасци на однесување. Овие податоци се складираат на сервери што може да се наоѓаат во различни држави, при што правната регулација може да биде различна и недоволна. Корисниците често не знаат кои компании имаат пристап до нивните податоци, под кои услови и за колкав период. Дури и кога се избришани профилите, податоците може да останат во резервни копии или да бидат продадени на други организации. Овие прашања се регулираат со закони како Општата регулатива за заштита на податоци (GDPR) во Европската Унија, но потребно е глобално ускладување за да се обезбеди ефикасна заштита. На индивидуално ниво, корисниците треба да бидат внимателни при споделување информации, да користат силни лозинки, да не ги прифаќаат сите дозволи без размислување и редовно да ги прегледуваат своите дигитални навики и поставки.

Дигиталниот отпечаток претставува моќен показател за навиките, интересите и идентитетот на секој корисник на интернет. Иако понекогаш е корисен за персонализација и подобрување на услугите, тој носи сериозни ризици кога се злоупотребува. Одговорноста за заштита на личната приватност не е само на институциите, туку и на самите корисниците кои треба свесно и внимателно да се однесуваат онлајн. Со користење на безбедносни алатки, критичко размислување и дигитална писменост, секој поединец може да го контролира својот дигитален отпечаток и да ја заштити својата приватност во дигиталниот свет.

ЗАКЛУЧОК

Дигиталниот отпечаток е моќна алатка, но треба да знаеш како да ја управуваш за да ги избегнеш ризиците. Со правилно користење на интернетот, можеш да ги заштитиш твоите податоци и да изградиш позитивна онлајн репутација.

ПРИМЕР 1

Несвесно создавање отпечаток

Ученик/чка се најавува на неколку апликации со иста лозинка и дозволува пристап до локација и контакти. Податоците се складираат и се проследуваат кон маркетинг–мрежи без негова/нејзина експлицитна дозвола.

ПРИМЕР 2

Селфи и изложување лични податоци

Корисник објавува селфи слика на социјална мрежа во реално време со локација и тагирани пријатели. Ваквото објавување остава дигитална трага која може да се анализира и да се злоупотреби.

ПРИМЕР 3

Употреба на VPN

Наставник користи VPN кога патува за да ги заштити личните податоци од јавни Wi-Fi мрежи. Со тоа се енкриптира комуникацијата и се ограничува создавањето на пасивен дигитален отпечаток.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Што значи поимот „дигитален отпечаток“?

2. Кои видови информации се дел од твојот дигитален отпечаток?

3. Зошто е важно да бидеш внимателен со тоа што споделуваш на интернет?

4. Како можеш да создадеш позитивен дигитален отпечаток?

5. Состави своја листа на правила за одговорно користење на интернетот и создавање позитивен дигитален отпечаток!

6. Дали избришаната фотографија е засекогаш избришана? Објасни каде се чуваат податоците на интернет!

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи како функционираат алгоритмите кои следат однесување на интернет – на пример cookies, tracking pixels, fingerprinting и data mining. Дополнително, дознај како компаниите собираат податоци преку мобилни апликации и што претставува поимот „податоци како валута“. Разгледај алатки за заштита на приватноста како DuckDuckGo, Tor, Signal и други платформи за безбедна комуникација.

2.6 Опасности на интернет

Во денешниот свет, интернетот е неодминлив дел од нашето секојдневие. Тој овозможува брз пристап до информации, комуникација без граници и бескрајни можности за учење и забава. Благодарение на интернетот, можеш да се поврзеш со луѓе од целиот свет, да истражуваш нови теми и да го подобриш своето знаење.

Покрај сите придобивки, тој крие и бројни опасности кои може да имаат сериозни последици врз твојот личен живот и врз општеството.

Една од најчестите опасности е дигиталното насилство – намерата да се понижи некое лице, да се повреди, да му се нанесе штета преку употреба на дигитални технологии, т. е. мобилни телефони и интернет. Дигиталното насилство може да се манифестира на многу начини. Некои од најраспространетите форми на дигитално насилство се уцени, закани, вознемирување, сексуална злоупотреба преку интернет, кибермалтретирање, креирање и користење лажни профили, како и злоупотреба на фотографии од други луѓе.

Друг ризик се содржини со омраза, кои поттикнуваат дискриминација и нетрпеливост кон одредени групи луѓе поради нивната раса, религија, пол или други карактеристики. Овие содржини негативно влијаат врз соживотот и разбирањето меѓу луѓето и може да доведат до сериозни конфликти.

На интернет може да се сретнат содржини што отворено или прикриено промовираат штетни и опасни однесувања, како што се употребата на наркотици, злоупотреба на алкохол, нарушувања во исхраната (анорексија, булимија) или дури и самоповредување и самоубиство. Овие содржини може да се во форма на видеа, блогови, постови на социјални мрежи или музички текстови и често се претставуваат како „начин за справување со болката“ или како нешто „нормално“. Пораките што ги пренесуваат, може да изгледаат привлечно, бунтовнички или дури како израз на личен идентитет, но всушност носат сериозни ризици за менталното и за физичкото здравје. Овие содржини може да создадат искривена слика за реалноста, да поттикнат самоуништувачко однесување или да го намалат чувството на вредност и самопочит кај младите.

Многу опасни се и дезинформациите и лажните вести, кои намерно се шират за да ги збунат луѓето или да им наштетат. За дезинформација можеме да кажеме дека е

Слика 4 *Кражба на податоци*

намерно лажна или погрешна информација што се шири со цел да се измамат луѓето, да се манипулира јавното мислење или да се нанесе штета некому. За разлика од неа, лажната вест е измислена или погрешно претставена вест, која изгледа како да е вистинска. Може да биде направена со или без намера да се наштети. Овие содржини често изгледаат како вистински вести – имаат наслов што привлекува внимание, фотографии и форматирани текстови што изгледаат убедливо. Но, ако не се проверени, тие можат да предизвикаат сериозни последици. На пример, лажни информации за болести или лекови можат да ја загрозат здравствената безбедност, особено ако луѓето престанат да ги следат препораките на медицинските лица. Во време на кризи, како што се природни катастрофи или пандемии, дезинформациите можат да создадат паника, недоверба и хаос.

На крајот, да не ги заборавиме и штетните компјутерски програми, вируси, кои можат да го оштетат твојот компјутер или да ги украдат твоите лични податоци. Тие често се кријат на сомнителни веб-страници, линкови во е-пошта или во непроверени апликации. Некои од нив можат да ги заклучат твоите податоци, други можат тивко да го следат твоето движење на интернет и да собираат чувствителни информации, како што се лозинки или банкарски податоци.

Зависноста од интернет и дигитални уреди е уште една опасност што сè почесто се среќава, особено кај млади корисници. Долготрајната изложеност на екрани, игри, социјални мрежи или платформи за видеосодржини може да доведе до нарушување на спиењето, недостаток на концентрација, социјална изолација и намалена физичка активност. Игровната зависност е посебна категорија која веќе е призната од Светската здравствена организација како ментално нарушување. Карактеристично е дека лицето чувствува силен импулс да игра, дури и кога тоа му ги нарушува здравјето, училишните обврски или социјалните односи. Слични ефекти има и претераното користење на социјални мрежи, кое може да доведе до споредување со другите, анксиозност или незадоволство од себе. Промовирањето на дигитална рамнотежа, воведувањето време без уреди и активности надвор од екран, се едни од најважните стратегии за спречување зависност.

„Ризичните програми“ и „ризичните содржини“ кои можат негативно да влијаат на твојата безбедност, менталното здравје, приватноста или однесувањето вклучуваат:

- непроверени апликации за преземање кои може да содржат вируси или шпионски софтвер,
- игри со неприфатливи сцени на насилство или говор на омраза,
- видеа, текстови или коментари што навредуваат луѓе поради вера, раса, пол, сексуална ориентација и сл.,
- платформи што го претставуваат користењето на наркотици како мода или самоповредувањето како начин за справување со емоции,
- бесплатни „хакерски“ алатки со кои се „отклучуваат“ одредени програми кои не се бесплатни,
- онлајн предизвици кои можеби изгледаат забавно, но се опасни и деструктивни и др.

Затоа, кога користиш интернет, размисли:

- Дали се бара од тебе споделување лични податоци?
- Дали содржината те наведува да направиш нешто опасно или незаконско?
- Дали влијае негативно врз твоето расположение или начинот на кој се гледаш себеси и другите?
- Дали е сомнителен изворот или анонимен?
- Како би постапил/а ако доживееш негативно искуство на интернет?

Секогаш користи го интернетот со фокус на содржините, критичко размислување и чувство на одговорност – кон себе и кон другите.

ЗАКЛУЧОК

Интернетот е одлично место за учење, дружење и забава – но истовремено може да биде и простор каде што се кријат сериозни ризици. Затоа е важно да бидеш внимателен и одговорен корисник.

ПРИМЕР 1

Фишинг порака по е-пошта

Ученик добива е-пошта во која пишува дека добил награда и треба да кликне на линк за потврда. По кликнувањето, внесува лични податоци кои потоа се злоупотребуваат.

ПРИМЕР 2

Зависност од игри

Средношколец поминува по 6 часа дневно играјќи онлајн игри, поради што ги запоставува училишните обврски, се успива и избегнува дружење со пријателите.

ПРИМЕР 3

Ширење лажна вест

Корисник споделува сензационалистички напис за „откритие“ што подоцна се докажува дека е целосно лажна информација. Тоа предизвикува паника кај некои од неговите пријатели.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Наведи три примери на злонамерни програми и интернетски содржини!

2. Како можат да влијаат дезинформациите врз луѓето и врз општеството?

3. Објасни ја разликата помеѓу дезинформација и лажна вест!

4. На кој начин насилната содржина исполнета со говор на омраза на интернет може да влијае врз тинејџерите?

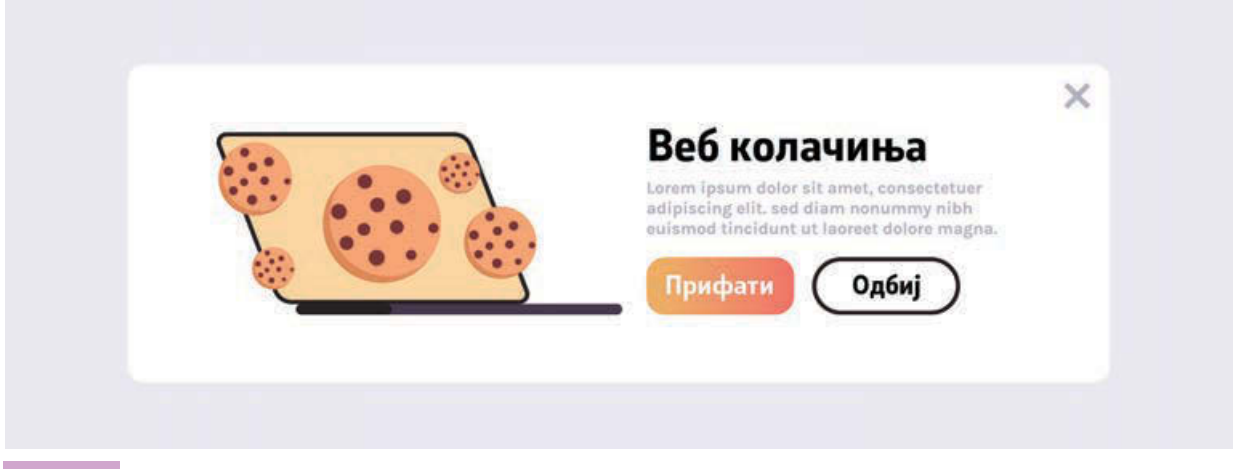
5. Зошто е важно да препознаеме промотивен материјал поврзан со наркотици или самоповредување?

6. Што можеш да направиш ако откриеш содржина што те вознемирува или те загрижува?

7. Зошто е важно критички да размислуваме кога користиме интернет?

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи што се „dark patterns“ – техники кои веб-страниците ги користат за да ги натераат корисниците да направат нешто што можеби не би го направиле доброволно (на пример: прифаќање на сите колачиња, купување производ, пријавување на билтен). Дополнително, анализирај случаи кога дезинформации довеле до реални последици во општеството, како протести, паника или финансиски измами.



Слика 5 *Услови за користење веб-страница*

2.7 Безбедно користење интернет

Безбедноста на интернет опфаќа многу аспекти, вклучувајќи разбирање на современите технологии, заштита на личните податоци и паметно и правилно однесување на интернет. Да се биде безбеден онлајн, значи да го користиш интернетот на начин што е корисен за тебе, а истовремено да бидеш заштитен од различни ризици и закани.

Најважна е заштитата на личните податоци. Споделувањето информации на интернет треба да се врши свесно, со разбирање дека податоците што се објавуваат – како што се адреси, слики, навики и мислења – може да бидат искористени надвор од контекст или злоупотребени. Секој корисник има право на приватност, но и одговорност да го почитува тоа право кај другите. Неетичко е да се снима, споделува или користи лична содржина на други лица без нивна согласност. Истовремено, важно е да се препознаат измамнички и манипулативни обрасци на интернет – од лажни огласи до сомнителни пораки – и да не се споделуваат ниту да се поддржуваат такви содржини. На тој начин ќе придонесеш кон создавање посигурна и подоверлива дигитална средина.

Следниве совети ќе ти помогнат да го заштитиш твојот онлајн идентитет и да избегнеш ризици:

- Користи силни и уникатни лозинки – важно е да имаш различни и комплексни лозинки за секоја твоја сметка. Користи најмалку 12 знаци, комбинирај големи и мали букви, броеви и специјални знаци, избегнувај очигледни зборови, или лични податоци како име или датум на раѓање.
- Внимавајте на фишинг измами и злонамерни линкови – никогаш не кликнувај на сомнителни линкови во е-пошта или пораки од непознати извори. Фишингот е обид за кражба на твоите лични податоци преку лажни веб-страници или пораки, затоа секогаш проверувај ја доверливоста на изворот.
- Активирај двофакторска автентикација (2FA) – ова е дополнителен безбедносен слој кој бара, покрај лозинката, и дополнителен код (на пример, од мобилен телефон). Со 2FA, дури и ако некој ја дознае твојата лозинка, нема да може лесно да пристапи до твојот профил.
- Заштити ги твоите уреди – редовно ажурирај ги софтверот и оперативниот систем, бидејќи тие ажурирања често содржат безбедносни поправки. Користи антивирусни програми кои ќе ги откријат и ќе ги блокираат потенцијалните закани.
- Проверувај дали веб-страницата започнува со “https://”, што укажува на безбедна конекција. Избегнувај користење јавни Wi-Fi мрежи за сензитивни активности, како банкарски операции или купување преку интернет.
- Преземај апликации, филмови и музика од проверени и безбедни веб-страници и соодветни на твојата возраст.
- Постави соодветни поставки за приватност на уредите и платформите кои ги користиш.
- Повеќето платформи и веб-страници имаат можност за пријавување несоодветни содржини и нивно обележување или злоупотреба од злонамерни корисници.

- Приспособи ги опциите за споделување информации за да го ограничиш пристапот на непознати лица. Биди свесен какви податоци споделуваш и со кого.
- Доколку се соочиш со непријатна ситуација како што е **кибернасилство** или онлајн предатор, разговарај со доверлива личност или со твоите родители, кои ќе ти помогнат во нејзиното разрешување.

Постојат различни програми кои ќе ти помогнат да го заштитиш компјутерот од опасности како вируси, малициозен софтвер и хакерски напади. Следните примери се користат за различни функционалности:

- **Антивирусни програми:** Касперски, Аваст, Битдифендер, (Kaspersky, Avast, Bitdefender) – ги откриваат и ги отстрануваат вирусите.
- **Фајрвол (заштитен ѕид)** кој го контролира интернет сообраќајот и спречува неовластен пристап.
- **Антималвер програми,** Малвербајтс (Malwarebytes) – заштита од шпионски софтвер и опасни апликации.
- **Филтри за содржина** – родителски контроли и филтри кои блокираат непожелни веб-страници.
- **Програми за ажурирања на оперативниот систем**, – со редовни надградби ја зголемуваат безбедноста на системот.

Овие програми помагаат компјутерот да работи безбедно и стабилно.

Безбедното работење на компјутер треба да го користиш правилно за да не ти влијае на физичкото и менталното здравје. Подолу се наведени неколку активности кои ја одржуваат добросостојбата.

1. Одмори ги очите од работа пред екран, прави пауза на секои 30 – 40 минути. Прошетај, истегни се или одмори ги очите.

2. Постави здрави навики, односно ограничи го времето поминато на социјалните мрежи. Исто така, не користи уреди пред спиење (минимум еден час).

3. Разговор со родители/наставници. Ако се соочиш со непријатност или онлајн вознемирување, разговарај со возрасен.

4. Дигитална добрина. Биди љубезен/зна во онлајн комуникацијата, пријави несоодветна или опасна содржина.

5. Внимавај на менталното здравје. Не се споредувај со луѓе на интернет – многу содржини се нереални. Користи интернет за учење, хоби и развој, не само за забави.

ЗАБЕЛЕШКА:

Во РС Македонија:

- Во согласност со Законот за заштита на лични податоци, секој корисник има право на заштита на своите лични податоци. При споделување или обработка на податоци, треба да се почитува приватноста, а личните податоци да се користат само со согласност или според законски овластувања.
- Кривичниот законик предвидува санкции за кибер-кривични дела, како што се кибернасилство, ширење вируси и злонамерен софтвер.

- Законот за електронски комуникации во Република Северна Македонија игра важна улога во обезбедувањето безбедно користење на интернетот преку создавање рамка која помага да се обезбеди заштитена, сигурна и одговорна употреба на интернетот во земјата.

ЗАБЕЛЕШКА:

Глобално, Денот на безбеден интернет се слави во февруари секоја година за да се промовира безбедното и позитивно користење на дигиталната технологија за децата и младите.

ЗАКЛУЧОК

Со паметно и правилно користење на интернетот, можеш да ги искористиш сите негови предности, а истовремено да се заштитиш од потенцијалните ризици. Корисникот треба да биде добро информиран, внимателен и одговорен, со цел интернетот да остане безбеден простор за секого.

ПРИМЕР 1

Почитување на личната приватност

Ученик/чка прави групна фотографија на училиштен настан, но пред да ја објави на социјална мрежа, бара дозвола од сите присутни и поставува ограничувања на пристап.

ПРИМЕР 2

Пријавување навредливи содржини

Корисничка забележува дискриминаторски коментар во јавна група и одлучува да го пријави на платформата, наместо да одговори со иста агресивност.

ПРИМЕР 3

Вклучување во едукативна дискусија

Ученик/чка се вклучува во онлајн училишна дебата и изразува спротивно мислење, но тоа го прави на учтив и аргументиран начин, поттикнувајќи позитивен дијалог.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Кои три програми или алатки се користат за одржување безбедност при користење компјутер?

2. Објасни зошто е важно да се почитуваат правилата за безбедно користење на интернет според законската регулатива!

3. Примени едно правило за безбедност и објасни како би постапил/а ако добиеш сомнителна порака или линк!

4. Анализирај две ситуации и утврди во која има поголем ризик за безбедноста на корисникот:

а) споделување лозинка со пријател или

б) преземање програма од непозната веб-страница. Објасни зошто!

5. Процени дали користењето социјални мрежи повеќе од пет часа дневно може да влијае на општата добросостојба! Аргументирај го твојот став!

6. Осмисли пет правила за безбедно користење интернет кои би им ги препорачал/а на учениците од твојата паралелка!

ПОВТОРУВАЊЕ ЗА ТЕМА 2

1. Во што се разликуваат компјутерска мрежа и интернетот?
2. Наведи три интернет-сервиси (услуги)!
3. Што значи поимот „дигитален отпечаток“?

4. Објасни со свои зборови како интернетот овозможува добивање и споделување информации!
5. Зошто е важно да се користат валидни и проверени извори на веб?
6. Објасни една позитивна и една негативна последица од дигиталниот отпечаток!

7. Пронајди една веб-страница за која сметаш дека е валиден извор на информации и наведи две причини зошто е доверлива!
8. Дај пример како би користел/а електронска пошта или облак услуга за споделување училиштен проект.
9. Опиши како би се заштитил/а од една конкретна опасност на интернет (на пример: вирус, лажен профил, фишинг порака)!

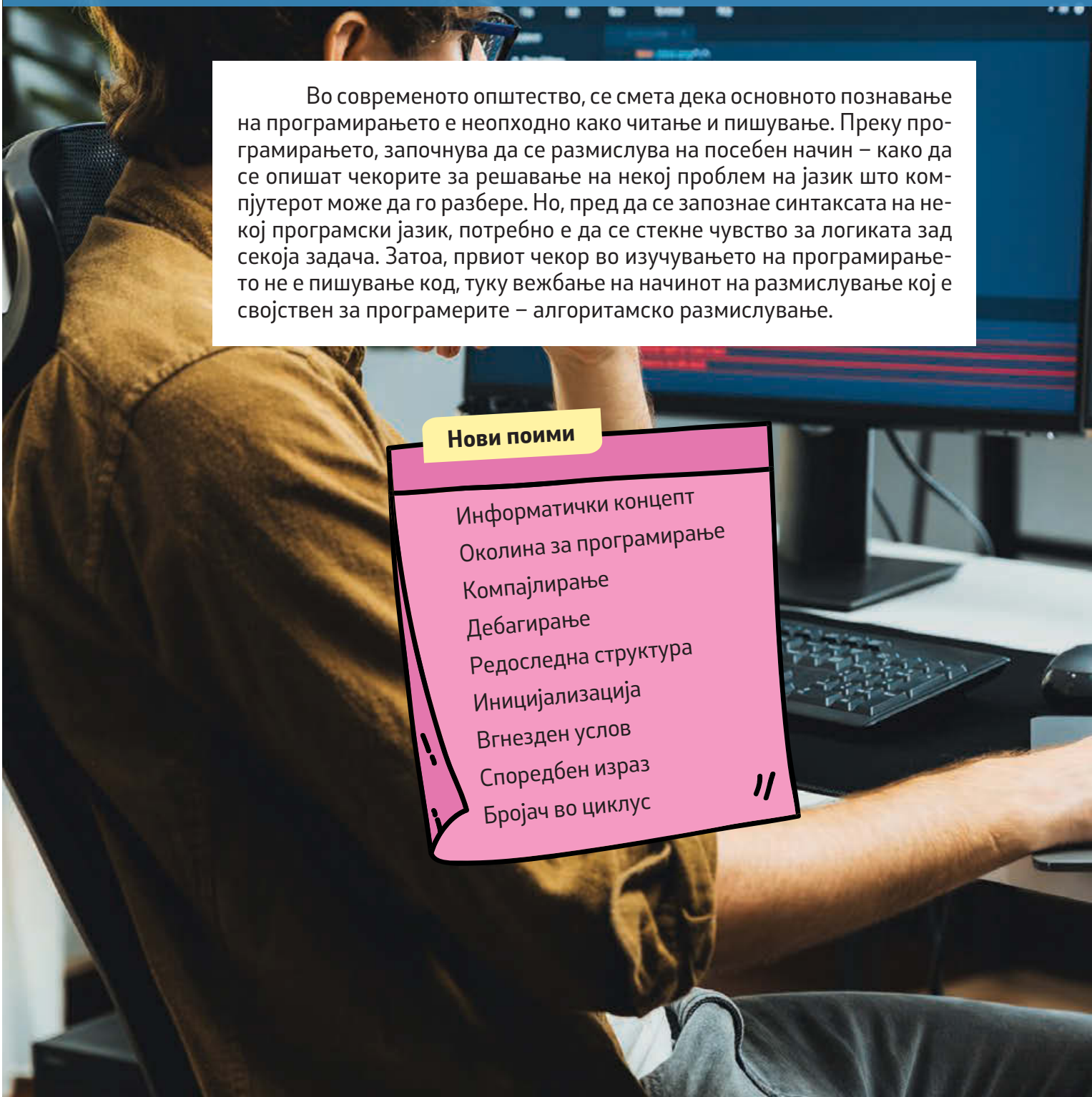
10. Спореди два вида интернет-услуги: на пример, електронска пошта и социјални мрежи! Во што се слични, а во што се разликуваат?
11. Анализирај зошто некои веб-страници се небезбедни (препознај три знаци)!
12. Објасни како може да влијае дигиталниот отпечаток врз идните можности на една личност!
14. Процени дали следниот извор е валиден ако ги има следните елементи: веб-страница со многу реклами, без автор и без датуми! Објасни ја твојата процена!
15. Дали е етички прифатливо да се сподели слика на другар на социјални мрежи без негово знаење? Објасни зошто!
16. Процени дали е безбедно да се прифати пријателска покана од непозната личност на некоја социјална мрежа!

17. Создај кратка интерактивна презентација или инфографик за тоа како безбедно да се користи интернет со минимум три правила.

- Направи миниводич за ученици од твоја возраст! Нека содржи: дефиниција за интернет, два безбедносни совети и еден пример за добар и еден за лош дигитален отпечаток.

18. Креирај дигитална содржина (постер, краток текст, слика или минивидео) користејќи информации што ги пронајде од валидни веб-извори!

ТЕМА 3: Програмирање во C++



Во современото општество, се смета дека основното познавање на програмирањето е неопходно како читање и пишување. Преку програмирањето, започнува да се размислува на посебен начин – како да се опишат чекорите за решавање на некој проблем на јазик што компјутерот може да го разбере. Но, пред да се запознае синтаксата на некој програмски јазик, потребно е да се стекне чувство за логиката зад секоја задача. Затоа, првиот чекор во изучувањето на програмирањето не е пишување код, туку вежбање на начинот на размислување кој е својствен за програмерите – алгоритамско размислување.

Нови поими

- Информатички концепт
- Околина за програмирање
- Компајлирање
- Дебагирање
- Редоследна структура
- Иницијализација
- Вгнезден услов
- Споредбен израз
- Бројач во циклус

3.1 Логички и алгоритамски задачи

Веќе знаеш да...

- препознаеш едноставни логички ситуации од секојдневниот живот,
- решаваш проблеми користејќи здрав разум и логичко размислување,
- следиш упатства чекор по чекор,
- забележиш обрасци и правила во игри и задачи,
- правиш разлика меѓу точно и неточно тврдење.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

1. Што е тоа Scratch?

2. Што е тоа натпреварувачки тип задачи?

3. Што се тоа променливи?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- разликуваш логичка задача од математичка пресметка,
- го анализираш начинот на решавање алгоритамски проблем,
- го следиш текот на логичко размислување чекор по чекор,
- ги користиш основните поими: услов, наредба, резултат,
- ги идентификуваш чекорите на едноставен алгоритам,
- разбираш зошто е важно алгоритамското размислување.

Програмирањето како нов јазик на размислување

Програмирањето претставува дисциплина која ги поврзува математиката, логиката, и решавањето проблеми. Се работи за процес кој започнува со анализа на проблем, продолжува со планирање решение преку чекори и завршува со реализација. Во оваа прва лекција, нема да се користи ниту еден ред код. Ќе се навлезе во суштината на логичкото и алгоритамското размислување преку интересни и едноставни примери. Тие ќе помогнат да се разбере како се поставува еден проблем, како се анализира и како се пристапува кон негово решавање.

Со логичка задача се означува секоја ситуација каде што еден проблем може да се реши само доколку се размислува логички, чекор по чекор, без потреба од сложени пресметки. Прво, се поставува целта: што треба да се постигне? Потоа, се согледуваат сите информации што се дадени. Се анализира нивната поврзаност и се идентификува кои од нив се клучни. Конечно, се планираат можни решенија.

ПРИМЕР 1

Место на предметот

На една маса има три кутии: црвена, сина и жолта. Едната од нив содржи предмет. Над секоја кутија има по едно тврдење:

- **Црвената:** „Предметот е во оваа кутија.“
- **Сината:** „Предметот не е во црвената кутија.“
- **Жолтата:** „Предметот не е во оваа кутија.“

Само едно од тврдењата е точно. Во која кутија е предметот?

Анализата започнува со претпоставка. Ако е точно тврдењето на црвената кутија, тогаш предметот е таму, но и тврдењето на жолтата (дека не е во неа) би било точно, што значи две точни – што не е дозволено. Ако е точно тврдењето на сината – дека не е во црвената, тогаш предметот е во жолтата, но и жолтата би имала точно тврдење. Единствената опција според која само едно тврдење е точно, е ако предметот е во сината кутија. Ова е пример кога се користи логика без пресметка.

Втората важна способност што се развива при решавање логички задачи е препознавање обрасци (шаблони) и поврзување услови. Тоа значи дека при даден број податоци, се бара каков однос имаат тие меѓусебно и што може да се заклучи од нив. На овој начин се развива способноста да се планираат чекори што водат до целта.

ПРИМЕР 2

Движење по патека

Робот се движи по табла од 5 реда и 5 колони. Почнува од горниот лев агол. Во секој потег може да оди само надолу или десно. Колку различни патишта постојат до долниот десен агол?

Ова е класична задача од типот „броење патишта“, каде што секој потег е ограничен со одредени правила. Иако уште не се пресметува ништо, може да се замислат сите можни комбинации на движење: колку пати се оди десно и колку пати надолу. Ова го развива чувството за структурирање на решението и за бројноста на можни опции – основа за алгоритамско размислување.

Следниот важен аспект на логичките задачи е идентификација на услови и последици. Во многу задачи, постои збир од правила или ограничувања што мора да се почитуваат и врз основа на нив, се заклучува што може или не може да се случи. Тоа се нарекува анализа на услови.

ПРИМЕР 3

Загатка со исклучување

Пет ученици седат во ред. Се знае дека:

- Петар не седи крај Марина.
- Сашко седи десно од Ана.
- Ана не е крај Петар. Кој каде седи?

Преку логичка елиминација и изведување, може да се најде точниот распоред. Ваквите задачи бараат внимателно читање и постепено исклучување на невозможни опции. Не се користат броеви, туку врски меѓу учесници и ограничувања. Ова е одличен тренинг за развој на систематичност и логичка строгост.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Алгоритмите често се користат во секојдневниот живот: следење рецепти, навигација, дури и решавање судоку. Натпреварот „Дабар“ е меѓународен настан во кој учениците решаваат логички задачи слични на овие. Ако ти се допаднаа овие примери, можеш да се обидеш со други задачи на официјалната страница на натпреварот или да создадеш свои логички загатки и да ги решаваш со пријателите.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што значи логичко размислување во контекст на алгоритамски задачи?

2. Зошто се важни чекорите при решавање задача?

3. Кога се вели дека некое тврдење е логички невозможно?

4. Зошто задачите без броеви можат да бидат корисни при учење програмирање?

5. Пронајди ја логичната лажна реченица ако две од следниве се точни:

- Синиот живее лево од Жолтиот.
- Зелениот живее десно од Синиот.
- Жолтиот не е крај Синиот.

6. Тројца пријатели имаат по една книга: роман, енциклопедија и бајка. Се знае дека Јана не чита енциклопедија, а Бобан не ја чита бајката. Кој што чита?

7. Осмисли своја логичка задача со најмногу три услови и најмалку две можни решенија!

Запомни што научи!

- Логичките задачи бараат внимателна анализа на податоци и тврдења.
- Алгоритамското размислување започнува со јасно дефинирани цели и чекори.
- Не се потребни броеви или код за да се вежба решавање проблеми.
- Секој проблем може да се реши систематски, со идентификација на можностите и исклучување на невозможните опции.

3.2 Информатички концепти

Веќе знаеш да...

- размислиш логички за редоследот на настаните,
- го препознаеш значењето на редослед и правила во секојдневни задачи,
- употребуваш стратегии за донесување одлуки,
- се организираш при решавање проблеми,
- препознаеш кога нешто може да се направи поефикасно.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Што претставува анализа на услови?

2. Замисли патека во лавиринт со ограничени насоки (само десно или долу). Колку чекори се потребни до целта?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што претставуваат податочните структури,
- препознаеш стратегии за оптимизација на задачи,
- разликуваш пристапи како „раздели па владеј“ и „алчно оптимизирање“,
- разбереш зошто распределувањето на ресурси е важно,
- ги примениш концептите во секојдневни проблеми.

Во светот на информатиката, концептите се темелите врз кои се гради способноста за решавање сложени задачи. Без основно разбирање на начините на кои се организираат, управуваат и оптимизираат податоците, не може да се развие солидна основа за програмирање. Секој проблем со кој се соочува компјутерот, а и човекот, има структура, ограничувања и цел што треба да се постигне. Затоа е потребно да се изучат техники што помагаат за рационално донесување одлуки и ефикасно искористување на ресурсите.

Покрај тоа, информатичките концепти овозможуваат решенијата да бидат не само точни, туку и оптимални. Начини на организирање податоци, разбирање на моделите за решавање, како и избор на вистинската стратегија за конкретен проблем, се

дел од вештините што ќе се градат преку оваа лекција. Тоа се вештини што се користат и во реалниот свет, кога треба да се избере најдобар пат, да се распоредат задачи или да се организира време.

Податочни структури

Податочните структури претставуваат начин на кој се организираат и се чуваат информациите, за полесно да може да се користат. На пример, списокот на ученици во дневник или табела со распоред на часовите се реални примери на структури што служат за складирање податоци. Во информатиката, таквите структури можат да бидат редици, множества, листи, дрвја или графови – во зависност од тоа како и каде се применуваат. Секој од тие начини има свои предности и се користи кога се бара одредена ефикасност или начин на пристап до информациите.

ПРИМЕР 1

Замисли библиотека во која сите книги се поставени по случаен редослед. Ќе биде тешко и бавно да се најде потребната книга. Но, ако се тие подредени по жанр, автор или наслов, станува многу полесно. Тоа важи и за компјутерите – кога се податоците организирани во структури, тие побрзо се пребаруваат и користат.

Распределување на ресурси

Распределувањето е концепт кој се однесува на тоа како ограничените ресурси (време, простор, енергија) се доделуваат на задачи или процеси. Во информатиката, ова може да значи како ќе се подели процесорското време меѓу повеќе програми, или како ќе се искористи меморијата за различни задачи. Добро распределување може да доведе до многу побрзо и поефикасно извршување на задачите, додека лошо распределување води до застои и губење ресурси.

ПРИМЕР 2

Замисли дека треба да се подели распоред за учење меѓу пет предмети во една недела. Ако се потроши премногу време на еден предмет, другите ќе останат необработени. Добар план значи рамномерна распределба според важноста или тежината на секој предмет – концепт што го користат и компјутерите при управување со задачи.

Алчно оптимизирање

Алчниот (greedy) пристап е стратегија каде што на секој чекор се прави избор што изгледа најдобар во моментот, без да се размислува за идните последици. Иако ова секогаш не води до најдобро можно решение, често доведува до прифатлив резултат со многу помалку ресурси. Таков пристап се користи кога е поважно брзо да се дојде до одговор, отколку тој да биде совршен.

ПРИМЕР 3

Замисли дека имаш 100 денари и сакаш да купиш што е можно повеќе грицки. Ако на секој чекор го избираш најевтиниот производ што го гледаш, може да купиш повеќе, но можеби не најквалитетно. Ова е алчен пристап, сличен на некои алгоритми што веднаш го избираат најдоброто локално решение.

Раздели па владеј

„Раздели па владеј" (divide and conquer) е стратегија каде што голем проблем се дели на помали, полесни за решавање потпроблеми. Потоа, тие се решаваат одделно и резултатите се комбинираат во едно финално решение. Овој метод е често користен во информатиката, бидејќи ја намалува сложеноста и ја зголемува ефикасноста на решението. Наместо да се напаѓа проблемот како целина, тој се решава во делови.

ПРИМЕР 4

Кога се подготвува презентација, не се пишува целиот текст одеднаш. Прво се прави наслов, потоа содржина по делови, слики, и на крај – обединување. Овој пристап може да се примени и во компјутерски решенија.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што претставуваат податочните структури?

2. Зошто е важно распределувањето на ресурсите?

3. Дали секогаш алчниот пристап дава најдобро решение?

4. Наведи пример од секојдневниот живот што одговара на податочна структура!

5. Осмисли план за учење каде што се користи распределување на времето!

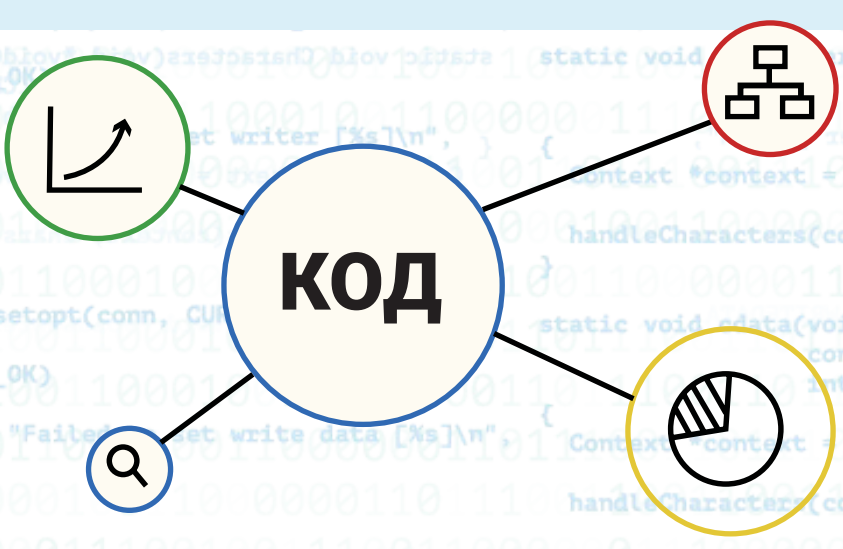
6. Опиши како би организирал/а библиотека користејќи некоја структура!

Запомни што научи!

- Податочните структури се основа за ефикасно управување со информации.
- Распределувањето ресурси овозможува подобро користење време и простор.
- Алчниот пристап избира најдобро локално решение, но секогаш не води до глобално најдобро.
- „Раздели па владеј“ се користи за поефикасно решавање сложени задачи преку разложување.
- Информатичките концепти се применливи и надвор од програмирањето – во секојдневниот живот.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истите концепти што се користат во програмирањето, како оптимизација, распределување и моделирање, се изучуваат и во економијата, менаџментот и математиката. На пример, при оптимизација на патишта во транспорт или планирање на работни смени. Податочните структури како графови се користат во социјалните мрежи за анализа на поврзаноста меѓу луѓе. Алчните алгоритми се вградуваат во мобилни апликации за брзо донесување одлуки (на пример: избор на најкратка патека). Истражете како се користат во реални системи и размислете каде би ги примениле.



3.3 Бинарни броеви: претставување и примена

Веќе знаеш да...

- разликуваш броеви и начини на нивно претставување,
- користиш децимален број како вредност во секојдневни ситуации,
- ја препознаеш важноста на точноста при работа со броеви,
- користиш разни единици мерки и да ги пресметуваш основните аритметички операции,
- објасниш каде и како се среќаваат броеви во технологијата околу нас.

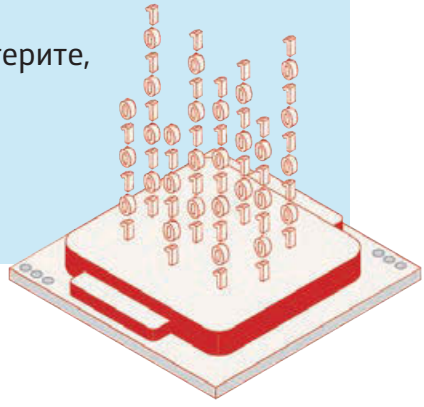
ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Што значи стратегијата „раздели па владеј“?

2. Напиши сценарио во кое разделувањето на задача води до полесно решавање!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш што е бинарен систем и како функционира,
- претставуваш цели броеви во бинарен запис,
- разликуваш помеѓу бинарен и децимален систем,
- поврзеш бинарни броеви со основната работа на компјутерите,
- претставиш информации како битови и бајти,
- применуваш бинарни броеви во практични ситуации.



Во дигиталната ера, речиси секоја технологија е поддржана со процеси во кои се користат броеви. Броевите не се само дел од математиката, туку се сржта на комуникацијата меѓу човекот и машината. Секој компјутер, телефон, дигитален часовник или микропроцесор комуницира и обработува информации користејќи броеви. Но, овие броеви не се истите какви што секојдневно се користат. Компјутерите користат посебен систем кој овозможува прецизна, брза и доверлива обработка: **бинарниот систем.**

Сè што се гледа на екранот – слики, звуци, текстови – во основа се трансформира во серија на бинарни вредности. Ова значи дека секоја информација што ја внесува или ја добива еден компјутер, на крајот се сведува на вредности составени само од две можности – 0 и 1. Со други зборови, компјутерите не „размислуваат" со децимални броеви (0 – 9), туку со бинарни, кои се основа на дигиталната логика.

Претставувањето на броевите во компјутерската наука се базира на **бинарниот систем**, кој користи само две цифри: 0 и 1. Овој систем е избран затоа што е многу поедноставно и поефикасно електронските уреди да препознаваат две состојби – присуство или отсуство на струја, односно висока или ниска вредност на напон. Со оваа

ПРИМЕР

Да се претстави бројот 5 во бинарен систем.

Се користи метода на делење со 2:

5 / 2 = 2 остаток 1

2 / 2 = 1 остаток 0

1 / 2 = 0 остаток 1

Се чита одоздола нагоре: 101

Значи, 5 во бинарен систем е 101.

Претставувањето информации со нули и единици се нарекува битови. Еден бит е најмалата количина на податок и може да биде или 0 или 1. Осум бита формираат бајт, што е најчесто користеното мерно количество во складирањето и трансферот на податоци.

Во дигиталните уреди, како што се фотоапаратите, се користи бинарниот систем за да се кодифицира секој пиксел со одредена вредност. Колку повеќе битови се користат за секој пиксел, толку е поголема прецизноста на боите и деталите. Во секој уред, па и во најмалите компоненти, бинарниот код ја има главната улога во процесирање и меморирање податоци.

ПРИМЕР

Како се кодира буквата **А** во бинарен код?
Во **ASCII стандардот**, буквата **А** има број 65. Претставено во бинарен систем, тоа е:
65 / 2 = 32 остаток 1
32 / 2 = 16 остаток 0
16 / 2 = 8 остаток 0
8 / 2 = 4 остаток 0
4 / 2 = 2 остаток 0
2 / 2 = 1 остаток 0
1 / 2 = 0 остаток 1
Се чита одоздола нагоре: **1000001**
Ова значи дека буквата А се чува како **1000001** во меморијата на компјутерот.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што е бинарен број?
2. Зошто компјутерите користат бинарен систем?
3. Претвори го бројот 9 во бинарен систем!
4. Што претставува бајт и колку бита содржи?
5. Претвори ги броевите 10, 12 и 15 во бинарен систем!
6. Со колку бита може да се претстават точно 256 различни вредности?

Запомни што научи!

- Бинарниот систем користи само две цифри: 0 и 1.
- Компјутерите работат со бинарни броеви поради електронската природа на нивните компоненти.
- Секој број, буква или податок во компјутерот е претставен во бинарна форма.
- Бит и бајт се основни единици за мерење податоци.
- Претворањето од децимален во бинарен систем се врши преку делење со 2.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Испитај како изгледа бинарна аритметика и како се врши собирање на бинарни броеви.
- Истражи како компјутерот користи двоичен систем за претставување слики и звуци!
- Прочитај за IEEE форматите на претставување децимални броеви (floating point)!
- Истражи како се претставуваат негативни броеви во компјутерот (на пример, со „дополнување до два“ – two's complement)!

3.4 Основи на кодирање и енкрипција

Веќе знаеш да...

- препознаваш бинарни броеви и да ги поврзуваш со дигитални претставувања,
- разликуваш логички и алгоритамски чекори,
- сфатиш што е информација и како се претставува со симболи,
- ги следиш основните чекори во анализата на проблем.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Колку вредности може да се претстават со 6 бита?

2. Напиши како би се претставил бројот 100 во бинарна форма!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што претставува кодирање и каде се применува,
- разликуваш кодирање од енкрипција,
- препознаеш примери на кодирање во секојдневниот живот,
- разбереш зошто се користат шифри и како помагаат за заштита на информацијата,
- разликуваш меѓу јавни и тајни пораки,
- размислуваш како би се претставиле податоци на различни начини.

Во светот на дигиталната комуникација постои постојана потреба информацијата да се претставува, пренесува и чува на начин што ќе биде сигурен, точен и разбирлив. Поради тоа, од најраните денови на човечката цивилизација, различни методи биле развивани за да се трансформираат пораките од еден облик во друг – процес познат како кодирање. Преку кодирањето, можело да се пренесуваат значења дури и без употреба на пишан јазик, а енкрипцијата, како посебна гранка, овозможила тие пораки да останат тајни. За да се разбере програмирањето и дигиталната обработка на податоци, потребно е прво да се совладаат овие основни концепти, бидејќи токму тие се основата врз која се градат безбедноста и размената на информации во современиот свет.

Разликата меѓу кодирање и енкрипција често се губи кога се користат во секојдневниот говор, но во информатиката тие имаат многу прецизни значења. Кодирањето подразбира претворање на информацијата од еден облик во друг заради полесно читање, обработка или складирање. На пример, користењето на бинарен код за претставување букви и симболи во компјутерот е форма на кодирање. Од друга страна, енкрипцијата претставува метод со кој се сокрива вистинската содржина на пораката, така што само овластените лица можат да ја дешифрираат. Разбирањето на овие поими претставува неопходна основа за идно учење за безбедност на податоци, дигитални сертификати и компјутерски мрежи.

Кодирање

Во историјата на човештвото, кодирањето било користено како начин за пренос на пораки меѓу луѓето, особено во ситуации кога информациите требало да останат разбирливи само за одредени примачи. Еден од најпознатите историски примери за ваков облик на кодирање е Морзеовата азбука, која се користела во телеграфската комуникација. Во овој систем, секоја буква од азбуката се претставувала со комбинација на кратки и долги сигнали (точки и цртички). Иако денес изгледа примитивно, Морзеовиот код се сметал за револуционерен начин на кодирање, бидејќи овозможувал комуникација преку едноставни електрични импулси. Во информатичка смисла, овој начин на кодирање претставува предвесник на дигиталната претстава на информацијата, каде што комбинации од симболи (или сигнали) се користат за точно претставување на значења.

ПРИМЕР 1

Морзеов код

Буквата „А“ во Морзеов код се претставува како „.-“, што значи една кратка точка следена од една долга цртичка. На истиот начин, „Б“ се претставува како „-...“. Целата англиска азбука, како и бројките и некои специјални знаци, имаат свој код во оваа азбука. Примената на Морзеовата азбука се состоела од испраќање електрични сигнали или светлосни импулси кои биле разбирани од примачите што ја познавале шифрата. Со тоа се овозможувало пренесување на информациите дури и во отежнати услови кога гласовната комуникација не била возможна.

Во информатичките системи, кодирањето не се применува само за прикривање на податоци, туку и за нивно претставување на начин што машините можат да го обработат. Најосновниот и најважен вид кодирање во компјутерскиот свет е **бинарното кодирање**. Целиот дигитален свет се базира на претставување на информации со само две состојби – 0 и 1. Овие бинарни вредности се користат за претставување на секој тип податок: броеви, текст, слики, звуци и видео. Секој знак од текстот, на пример, се претставува преку стандарден код како што е ASCII (American Standard Code for Information Interchange), каде што секоја буква, број или специјален знак има своја уникатна бинарна вредност.

ПРИМЕР 2

ASCII кодирање

Буквата „А“ според ASCII стандардот има вредност 65. Во бинарен запис тоа е 01000001. Слично, буквата „а“ има вредност 97, што во бинарен облик изгледа како 01100001. Со помош на вакви кодови, текстот може да се претвори во низи од бинарни вредности што компјутерот лесно ги зачувува и обработува. Ова покажува дека кодирањето не мора секогаш да има тајна функција, туку е клучно за комуникацијата меѓу човекот и машината.

Енкрипција

Енкрипцијата, како процес на прикривање на значењето на пораката, отсекогаш претставувала важен дел од безбедната комуникација. Еден од првите и најпознатите примери е Цезаровата шифра, именувана по Јулиј Цезар, кој ја користел за шифрирање воени пораки. Принципот на оваа енкрипција се базира на едноставно преместување на буквите во пораката за одреден број места во азбуката. Така, секоја буква од оригиналната порака се заменува со друга, која се наоѓа неколку места подалеку во азбуката. Иако денес оваа техника се смета за слаба и лесна за разбивање, таа претставува одличен вовед во концептот на симетрична енкрипција, каде што истиот клуч се користи и за шифрирање и за дешифрирање.

ПРИМЕР 3

Цезарова шифра

Ако во Цезаровата шифра се користи поместување за 3 позиции, тогаш буквата „А“ ќе стане „Д“, „Б“ ќе стане „Е“ и така натаму. На пример, зборот „ИНФО“ ќе се претвори во „ЛСКР“. За да се дешифрира пораката, примачот треба да знае дека е користено поместување од три позиции наназад. Овој едноставен метод покажува како може да се прикрие информацијата, но и колку е важно преносот на клучот да биде безбеден – ако некој го открие клучот, може лесно да ја дешифрира пораката.

Современото дигитално општество се потпира на напредни методи за енкрипција за да се гарантира безбедноста на комуникациите, трансакциите и складирањето податоци. За разлика од едноставните техники од минатото, како Цезаровата шифра, денешните методи користат сложени математички алгоритми и криптографски протоколи. Два од најчесто користените модели се симетричната и асиметричната енкрипција. Кај симетричната енкрипција, истиот клуч се користи за шифрирање и дешифрирање на податоците, што значи дека двете страни во комуникацијата мора да го поседуваат истиот клуч и да го чуваат во тајност. Од друга страна, кај асиметричната енкрипција, како што е RSA алгоритмот, се користат два клуча – јавен и таен – што овозможува безбедна комуникација и автентикација без претходна размена на тајни.

ПРИМЕР 4

Асиметрична енкрипција со RSA

Кога едно лице сака да испрати порака до друго лице преку интернет, може да ја енкриптира пораката со јавниот клуч на примачот. Откако ќе пристигне пораката, примачот ја дешифрира со својот приватен (таен) клуч. На овој начин, дури и ако некој ја пресретне пораката, нема да може да ја прочита без приватниот клуч. RSA и слични алгоритми се основа за денешните безбедни комуникации преку HTTPS, онлајн банкарство и дигитален потпис.

Криптографијата претставува наука за заштита на информации преку нивно претворање во форма која не може да се разбере без соодветен клуч. Таа била користена уште од антички времиња, но во дигиталната ера станала незаменлив дел од секојдневното функционирање на интернетот, банкарските системи, здравствените услуги и владините институции. За разлика од обичното кодирање, криптографијата има цел не само да претвори податоци, туку и да ги заштити од неовластен пристап.

Криптографските системи се базираат на математички принципи и алгоритми, и тие овозможуваат автентикација, доверливост, интегритет и непроменливост на податоците. Секој пат кога некое лице се најавува на веб-страница, прави онлајн трансакција или испраќа електронска пошта, криптографијата е таа што обезбедува дека информацијата е безбедна.

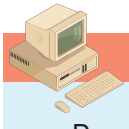
Современата криптографија се потпира на принципот на конверзија на читливи податоци (т.н. „обичен текст“) во шифрирани податоци („шифротекст“), со користење на криптографски алгоритми и клучеви. Притоа, податокот станува бесмислен за секој што ќе го пресретне, освен ако не го поседува вистинскиот клуч за дешифрирање. Овој процес може да биде симетричен, каде што истиот клуч се користи и за шифрирање и за дешифрирање, или асиметричен, каде што се користат два различни клуча – јавен и приватен. Симетричните системи се побрзи, но бараат безбеден начин за споделување на клучевите, додека асиметричните нудат поголема флексибилност и се користат во ситуации каде што безбедното пренесување на клуч не е можно. Примена на криптографија се среќава кај електронската пошта, банкарските трансакции, дигиталните потписи, сертификатите и енкрипцијата на складирани податоци на компјутер или мобилен уред. Без овие технологии, довербата во дигиталниот свет би била речиси невозможна.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што претставува поимот „кодирање“ и по што се разликува од енкрипцијата?
2. Објасни ја разликата меѓу симетрична и асиметрична енкрипција!
3. Кои се основните елементи на Цезаровата шифра и како функционира?
4. Каква улога играат јавниот и приватниот клуч во RSA алгоритмот?
5. Претвори ја следнава порака во Морзеов код: „HELLO“!
6. Шифрирај го зборот „DATA“ користејќи Цезарова шифра со клуч 3!
7. Објасни како би се користел RSA алгоритмот при испраќање електронска пошта!
8. Замисли дека имаш таен клуч за симетрична енкрипција! Опиши како би го споделил/а со соработник без некој друг да го пресретне!

Запомни што научи!

- Кодирањето е процес на претворање на информацијата од еден облик во друг, со цел полесна обработка, складирање или пренос.
- Енкрипцијата се користи за заштита на податоци, со тоа што информацијата се шифрира така што без клучот не може да се прочита.
- Пример за кодирање е Морзеовиот код, каде што буквите се претставени со точки и црти.
- Цезаровата шифра е еден од најстарите облици на енкрипција, базирана на поместување на буквите во азбуката.
- RSA е асиметричен криптографски алгоритам кој користи јавен и приватен клуч за безбедна комуникација.
- Во симетричната енкрипција, истиот клуч се користи и за шифрирање и за дешифрирање.



ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Во современата криптографија, освен RSA, се користат и други алгоритми како AES (Advanced Encryption Standard) и ECC (Elliptic Curve Cryptography).
- Дигиталните сертификати и SSL протоколот користат енкрипција за да обезбедат безбедност при интернет-комуникацијата.
- Хаш функциите (на пример SHA-256) се користат за проверка на интегритетот на податоците.
- Квантната криптографија е напредна област која користи квантни феномени за создавање непробивни шифри.
- Принципите на енкрипција и кодирање се основа на кибербезбедноста и без нив, модерната дигитална комуникација би била лесна мета за злоупотреба.

3.5 Што е програмирање?

Веќе знаеш да...

- објасниш што е компјутер и кои се неговите главни компоненти,
- разликуваш хардвер и софтвер,
- користиш дигитални уреди за секојдневни задачи.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Зошто Морзеовиот код се смета за форма на кодирање, но не и за енкрипција?

2. Напиши упатство како би се енкриптирала и дешифрирала порака користејќи симетричен клуч!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што претставува програмирањето,
- опишеш и да објасниш постапка за одредено решавање проблем.

Во секојдневниот живот, компјутерите се користат за голем број задачи: пресметување, обработка на текстови, цртање, комуникација и многу повеќе. Но, сам по себе, компјутерот не знае како да ги извршува овие задачи. Тој мора да добие јасни и прецизни инструкции за секој чекор што треба да го направи. Токму тоа е суштината на програмирањето. Програмирањето претставува процес на создавање низа инструкции, напишани на специјален јазик разбирлив за компјутерот – програмски јазик. Овие инструкции се групираат во т.н. програма, која го упатува компјутерот што точно да направи, во кој редослед и под кои услови. Целта на оваа лекција е да се запознаат учениците со основната суштина на програмирањето, неговата примена и зошто е важна оваа вештина во дигиталниот свет денес.

Програмирањето е активност во која луѓето – наречени програмери – создаваат упатства за компјутерот. Овие упатства, или инструкции, се нарекуваат код и се пишуваат на програмски јазик. Програмскиот јазик е средство преку кое човекот може да комуницира со машината. Секој јазик има свои правила, граматика и структура, слично како природните јазици. На пример, како што реченицата „Запали ја светилката“ има

јасна смисла за човек, така и инструкцијата turnOn(light) може да има смисла за компјутер, ако е напишана во разбирлив програмски јазик. Целта е да се опише што сакаме да направи компјутерот – чекор по чекор.

ПРИМЕР 1

Замисли дека треба да му дадеш инструкции на робот да изврши домашна задача. Ако сакаш да се измијат садовите, инструкциите би биле: „оди до мијалникот", „вклучи вода", „стави детергент", „измиј го садот", „исплакни", „стави го на сушење". Слично, програмерот му дава на компјутерот јасни чекори – секој чекор мора да биде точен, без нејаснотии.

Кога еднаш ќе се напише програмата, компјутерот може да ја извршува истата задача секогаш на ист начин и без грешка, сè додека кодот е правилен. Ова ја прави програмата моќна алатка за автоматизација – активности што би одзеле време и би биле подложни на човечки грешки, се претвораат во задачи што ги врши машината брзо и точно. Програмите не се користат само за големи и комплицирани проекти – тие се наоѓаат и во секојдневни работи: дигитрон, мобилен телефон, банкомат, фрижидер или паметен часовник – сите работат по некоја програма напишана од човек.

ПРИМЕР 2

Во калкулаторот, кога ќе се напише „5 + 3 =", во заднина се извршува програма која ги чита броевите, ја препознава операцијата „+", ја пресметува сумата и го прикажува резултатот. Сите овие чекори се претходно опишани во програмски код.

Важно е да се разбере дека програмерот не само што пишува код – тој размислува логички, ги дели задачите на помали делови, проверува дали сите можни ситуации се земени предвид и го тестира кодот за да се увери дека работи правилно. Кога некоја програма не се однесува очекувано, се прави т.н. „дебагирање" – откривање и исправање на грешките. Програмирањето е комбинација од логичко размислување, креативност и дисциплина. Не е нужно потребно големо математичко знаење, туку способност за решавање проблеми.

ПРИМЕР 3

За да се направи апликација што пресметува колку време е потребно за патување, треба да се внесе растојание и брзина, а програмата да пресмета време = растојание / брзина. Но, програмерот мора да предвиди што ќе се случи ако некој внесе брзина = 0. Тоа би довело до грешка – затоа, добриот програмер ќе внесе дополнителна проверка: „ако брзина = 0, прикажи порака: Внеси валидна вредност".

ЗАКЛУЧОК

Програмирањето не е само пишување код – тоа е уметност на прецизно и логично изразување на чекори кои компјутерот треба да ги изврши. Преку програмирање, компјутерите се користат за автоматизирање, забрзување и поедноставување на процеси во секојдневниот живот. Учењето програмирање значи стекнување способност да се решаваат проблеми, да се гради нешто ново и да се разбере како функционираат дигиталните алатки што секојдневно ги користиме.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што претставува програмски јазик?
2. Каква е разликата меѓу хардвер и програма?
3. Дали програмирањето секогаш бара математика?
4. Кои се основните активности на еден програмер?
5. Што е дебагирање и зошто е важно?

Запомни што научи!

- Програмирањето претставува пишување инструкции за компјутерот.
- Инструкциите се пишуваат на програмски јазици.
- Програмите се користат за автоматизација на задачи.
- Програмерот размислува логички и ја тестира програмата.
- Грешките се отстрануваат преку дебагирање.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Истражи кои се најпопуларните програмски јазици денес!
- Обиди се да решиш логичка загатка каде што секој чекор мора да биде јасен и точен!
- Посети некоја бесплатна онлајн платформа за учење програмирање како Scratch или Code.org!

3.6 Програмски јазик и преведувач

Веќе знаеш да...

- разликуваш хардвер и софтвер,
- објасниш што претставува програмирање,
- опишеш логичка постапка преку секвенца од инструкции.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Објасни со свои зборови што е програмирање!

2. Опиши една задача од животот што може да се автоматизира со програма!

3. Наведи неколку уреди што користат програми!

4. Што прави програмерот кога настанува грешка во програмата?

5. Зошто е важно секоја инструкција да биде јасна и точна?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што е преведувач,
- објасниш што е компајлер,
- направиш разлика меѓу различни начини на работа на преведувачот и компајлерот.

Кога луѓето сакаат да комуницираат со компјутерите, се користат специфични јазици кои машината може да ги разбере. Овие јазици, наречени програмски јазици, претставуваат начин на формално и логички организирано изразување кој овозможува пишување инструкции што компјутерот може да ги изврши. За разлика од природните јазици што луѓето ги користат за секојдневна комуникација, програмските јазици мора да бидат прецизни, структурирани и без двосмисленост. Без таков јазик, програмирањето не би било можно, бидејќи компјутерот не разбира човечки говор, туку само машински код – низа бинарни команди.

Програмскиот јазик овозможува изразување на логиката и чекорите што треба да се извршат за решавање одредена задача. Секој јазик поседува сопствена синтакса – правила според кои се пишува кодот. Постојат многу програмски јазици како C++, Python, Java, JavaScript и други, и секој од нив има свој стил на пишување и области на примена. Изборот на јазик зависи од задачата што се решава – некои јазици се подобри за мобилни апликации, некои за научни пресметки, други, пак, за веб-развој. Иако различни по изглед, сите тие имаат заедничка цел: да ги претворат човечките идеи во форма што компјутерот може да ја изврши.

ПРИМЕР 1

Во C++ се пишува инструкција cout << “Zdravo”; за да се прикаже текст. Во Python истата намера се изразува како print(“Zdravo”). Двата примера постигнуваат ист резултат – приказ на текст – но на различен начин, во согласност со синтаксата на секој јазик.

Програмскиот јазик не се разбира директно од компјутерот. Напишаниот код прво мора да биде „преведен“ во машински јазик. Оваа улога ја имаат **преведувачите**, кои може да бидат од два типа: компајлери и интерпретери. Компајлерот го зема целиот изворен код, го анализира и го претвора во извршна програма која може да се стартува самостојно. Интерпретерот, пак, го чита кодот линија по линија и го извршува директно, без создавање посебен фајл.

ПРИМЕР 2

Кога се пишува програма во C++, компајлерот прво го преведува целиот код во извршен .exe фајл. Потоа, тој фајл може да се стартува на компјутер. Во Python, интерпретерот директно чита што е напишано и веднаш го извршува. Разликата е слична на тоа дали една претстава ќе биде снимена на филм (компајлирана) или ќе се игра во живо пред публика (интерпретирана).

Преведувачот не само што го претвора кодот во машински јазик туку и проверува дали е правилна синтаксата. Ако има грешки во кодот – недостасува запирка, заграда или има несоодветно име на променлива – преведувачот ќе сигнализира и нема да го продолжи процесот. Оваа проверка е исклучително важна, затоа што спречува извршување погрешни или небезбедни инструкции. Така, програмскиот јазик и преведувачот заедно создаваат безбеден и ефикасен процес на развој на софтвер.

ПРИМЕР 3

Ако програмер напише во C++ cout << “Zdravo” без точка-запирка на крајот, компајлерот ќе врати грешка: „expected ‘;’ before end of statement“. Така, програмерот веднаш знае каде треба да интервенира и да ја исправи грешката, пред да се изврши програмата.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Истражи како работи компајлерот за C++ (g++) или Python интерпретерот!
- Обиди се да напишеш краток код во Python и види што ќе се случи ако направиш синтаксичка грешка!
- Посети ја страницата https://www.learnpython.org/ и обиди се со едноставни примери!

ЗАКЛУЧОК

Програмските јазици и преведувачите се клучни алатки во процесот на програмирање. Без нив, не би можело да се напишат, преведат и извршат инструкции за компјутерите. Тие овозможуваат прецизна, сигурна и ефикасна комуникација меѓу човекот и машината. Разбирањето на начинот на кој функционираат поставува темел за сите идни знаења поврзани со развој на софтвер и дигитални решенија.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Споредете го начинот на работа на компајлер и интерпретер!

2. Зошто е важна синтаксата во програмскиот јазик?

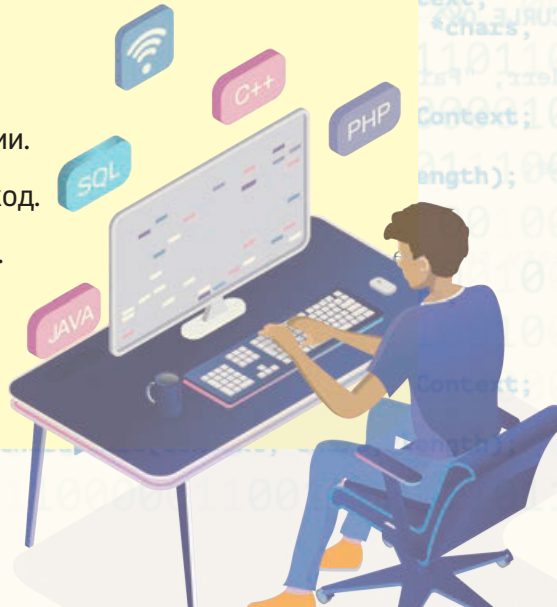
3. Дали еден програмски јазик може да се користи за сите задачи?

4. Која е улогата на преведувачот во процесот на извршување на програмата?

5. Што ќе се случи ако има грешка во кодот?

Запомни што научи!

- Програмските јазици се користат за пишување инструкции.
- Компајлерот ја претвора целата програма во машински код.
- Интерпретерот ја извршува програмата линија по линија.
- Синтаксата е збир на правила кои мора да се почитуваат.
- Преведувачот предупредува ако има грешки во кодот.



3.7 Разлика меѓу програмски јазици: Scratch, C++, Java, Python, PHP, Lisp

Веќе знаеш да...

- објасниш што е програмски јазик,
- правиш разлика меѓу компајлер и интерпретер,
- препознаеш грешки во синтаксата.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

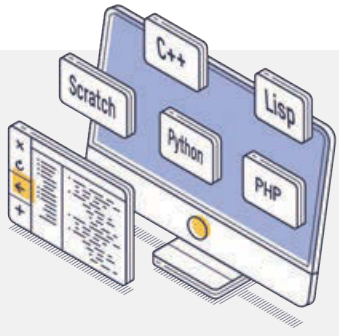
1. Што претставува програмски јазик?

2. Кои видови преведувачи постојат?

3. Кој е принципот на работа на компајлерот?

4. Кој е принципот на работа на интерпретерот?

5. Што претставува синтаксичка грешка?



ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш која е разликата помеѓу различни програмски јазици,
- разликуваш начин на работа на различни програмски јазици.

Во светот на програмирањето постојат десетици, па дури и стотици програмски јазици, од кои секој има свои предности, правила, примена и целна публика. Секој јазик е создаден со определена цел: некои се погодни за образовна употреба, други за развој на системски софтвер, трети за веб-апликации, а четврти за вештачка интелигенција. Иако сите служат за да се напишат инструкции што компјутерот ќе ги изврши, начинот на кој се пишуваат, се читаат и се извршуваат тие инструкции, може драстично да се разликува. Затоа е важно учениците да се запознаат со неколку различни јазици, со цел да стекнат пошироко разбирање за можностите на програмирањето.

Првиот јазик со кој најчесто се запознаваат почетниците е Scratch. Како што веќе е познато, станува збор за визуелен програмски јазик развиен од MIT, каде што наместо пишување код, се користат графички блокови кои се поврзуваат како сложувалка. Овој пристап ги елиминира синтаксичките грешки и им овозможува на учениците да се фокусираат на логиката, редоследот на операции и условите во програмата. Scratch се користи за создавање игри, анимации и едноставни симулации, со што се добива интерактивен начин на учење без заплашувачкиот изглед на „вистинскиот“ код.

ПРИМЕР 1

Во Scratch, ако сакате ликот да се движи напред 10 чекори, едноставно се избира блокот „помести се 10 чекори“ и се поврзува со останатите блокови. Нема потреба од пишување текстуален код или грижа за точка-запирки.

Следниот важен јазик е **C++**, еден од најстарите и најмоќните програмски јазици кој сè уште се користи денес. Тој е компајлиран јазик, што значи дека целата програма прво се преведува во машински код, па дури потоа се извршува. C++ е познат по својата брзина, контрола над меморијата и можноста за ниско-ниво операции. Тој се користи во развој на оперативни системи, вградени системи, па дури и за видеоигри. Сепак, поради сложената синтакса, не се препорачува како прв јазик за апсолутни почетници, но е одличен за напредни ученици. C++ е основа за многу социјални мрежи како и за криптовалути. Неговата брзина и стабилноста го прават идеален програмски јазик за многу типови софтвер.

ПРИМЕР 2

За да се прикаже текст во C++, се користи кодот cout << “zdravo”;, но претходно мора да се вклучи и заглавен фајлот #include <iostream> како заглавје и да се напише функција main() – што претставува поголем влезен праг од Scratch.

Java и **Python** се два модерни јазици со широка примена. Java е објектно-ориентиран и многу структуриран, се користи за мобилни апликации (особено Android), десктоп програми и веб-апликации. Python, пак, е скриптен јазик, многу популарен поради својата едноставна синтакса и читливост. Тој се користи во машинско учење, обработка на податоци, наука и автоматизација. Голем број почетници започнуваат токму со Python поради неговата едноставност.

ПРИМЕР 3

Во Python, за приказ на текст доволно е да се напише print (“Zdravo”), додека во Java истиот резултат бара повеќе редови код и повикување на објекти. Тоа го прави Python поинтуитивен за првиот контакт со програмирање.

PHP и **Lisp** претставуваат специјализирани јазици. PHP е најчесто користен за серверска веб-програма, особено кај динамички веб-страници. Тој се вметнува во HTML-кодот и овозможува интеракција со бази на податоци и обработка на форми. Lisp, пак, е еден од најстарите јазици, познат по својата применливост во вештачката интелигенција и при функционално програмирање. Иако ретко се користи кај почетниците, Lisp нуди сосема поинаков начин на размислување, бидејќи користи рекурзија и симболичка обработка на податоци.

ПРИМЕР 4

Во PHP, ако сакате да прикажете текст, се пишува: <?php echo “Zdravo!”; ?>, што покажува како се вметнува јазикот во HTML. Во Lisp, изразот (+ 1 2) значи собирање – операторот е на почеток, што е спротивно од повеќето јазици.

ЗАКЛУЧОК

Разбирањето на разликите меѓу програмските јазици им овозможува на учениците да направат информиран избор за тоа со кој јазик да продолжат. Секој од овие јазици има свои предности, слабости и подрачја на примена. Од Scratch за визуелна интуиција, до C++ за системска контрола, Java и Python за апликации, до PHP и Lisp за веб и вештачка интелигенција – спектарот е широк и вреден за истражување.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

Прашања:

1. Споредете ги Scratch и Python според начинот на пишување код!

2. Зошто PHP често се користи во веб-развој?

3. Кои сличности имаат Java и C++?

4. Кој јазик е подобар за апликации со вештачка интелигенција?

5. Опишете разлика меѓу компајлиран и интерпретиран јазик со пример!

Запомни што научи!

- Scratch е визуелен јазик, погоден за почетници.
- C++ е компајлиран и моќен за напредни задачи.
- Python има едноставна синтакса и широк спектар на примена.
- Java е објектно-ориентиран и стабилен.
- PHP се користи за веб, Lisp за вештачка интелигенција.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Инсталирај Scratch и обиди се да направиш анимација!
- Напиши кратка програма во Python и спореди ја со истата задача во Java!
- Истражи како се користи PHP на веб-страниците што ги посетуваш!
- Испробај Lisp во онлајн компајлер и анализирај ја неговата синтакса!

3.8 Интегрирана околина за програмирање

Веќе знаеш да...

- објасниш што е IDE,
- разликуваш компајлирање од интерпретирање,
- препознаеш грешка во синтаксата.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

1. Што е визуелен програмски јазик?

2. За што најчесто се користи C++?

3. Кој јазик е најпогоден за почетници?

4. Кои се предностите на Python?

5. Што е особено за Lisp?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што е IDE,
- препознаеш различна развојна околина.

Во современото програмирање, процесот на пишување, тестирање и извршување код не се одвива во празна конзола или рачно во командна линија. Наместо тоа, се користат специјализирани алатки кои го олеснуваат целиот процес на развој. Овие алатки се познати под името интегрирана развојна околина (IDE). Една IDE претставува целосен софтверски пакет што ги содржи сите потребни компоненти за ефикасно програмирање: уредувач на код, компајлер, дебагер и конзола за извршување. Оваа целосна околина се користи и од почетници и од професионални програмери, затоа што нуди структуриран начин на работа и визуелна контрола над процесот на развој.

Интегрираната развојна околина претставува „дом“ на програмерот – место каде што се пишуваат, се проверуваат и се извршуваат програмите. За разлика од ед-

ноставниот уредувач на текст, IDE нуди функции како автоматско довршување на код, обележување на синтаксички грешки, проверка на типови и брз пристап до алатки за компајлирање. Овие карактеристики не само што го забрзуваат развојот туку го прават и попродуктивен. Преку IDE, ученикот може веднаш да види дали некој дел од кодот е напишан правилно, како и да го тестира со притиснување на едно копче.

ПРИМЕР 1

Во Visual Studio Code, додека се пишува cout, IDE автоматски го нуди остатокот од изразот, додавајќи << и предложувајќи endl. Ова го намалува ризикот од грешки и ја зголемува брзината на пишување.

Во една IDE, постои и можност за директно компајлирање и извршување на кодот во рамките на програмата. Нема потреба ученикот рачно да отвора командна линија, да бара фајл и да го стартува преку посебна команда. Со еден клик, целата програма се преведува и се извршува. Дополнително, IDE овозможува уредување на повеќе фајлови истовремено, следење на структурата на проектот и уредување на зависности меѓу деловите од кодот.

ПРИМЕР 2

Во Code::Blocks, откако ќе се напише програмата, се притиска копчето „Build and Run", по што се појавува резултатот веднаш во долниот прозорец. Нема потреба од надворешно пребарување на грешки.

Уште една важна компонента на интегрираната околина е **дебагерот** – алатка која овозможува следење на извршувањето на кодот чекор по чекор. Со оваа функција, ученикот може да постави т.н. „breakpoint" – место каде што програмата ќе застане – и да види кои вредности имаат променливите во тој момент. Ова е од непроценлива вредност за разбирање на логиката на програмата и за решавање на т.н. логички грешки.

ПРИМЕР 3

Ако во една програма променливата x треба да има вредност 5, но IDE покажува дека таа добива вредност 3, преку дебагерот се лоцира редот во кој се случува грешката и се предлага корекција.

Модерните IDE нудат и дополнителни модули како симулатори за хардверски уреди, вградени терминали, или интеграција со системи за контрола на верзии како Git. Во образовен контекст, некои IDE се прилагодени за почетници, со поедноставен изглед и ограничени функции – како што се Thonny за Python или Arduino IDE за микроконтролери. Овие варијанти обезбедуваат почетна рамнотежа меѓу функционалност и едноставност, со што се овозможува полесна транзиција од визуелни алатки кон реален код.

ЗАКЛУЧОК

Интегрираната развојна околина претставува централно место за пишување и тестирање програми. Без разлика дали се работи за почетнички проект или професионална апликација, користењето на IDE значително ја олеснува работата, ја зголемува ефикасноста и ја намалува можноста за грешки. Познавањето на нејзините компоненти и функции е прв чекор кон вистинско програмирање.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

1. Што претставува IDE?
2. Кои се трите главни компоненти на една IDE?
3. За што се користи дебагерот?
4. Дали е можно да се изврши програма без IDE?
5. Кои IDE се користат за Python?

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Опиши ја разликата меѓу IDE и обичен уредувач на текст!
2. Која е улогата на автоматското довршување во IDE?
3. Како помага дебагерот при откривање грешки?
4. Што се breakpoints и кога се користат?
5. Која е најсоодветната IDE за почетници и зошто?

Запомни што научи!

- IDE претставува интегрирана средина за пишување, проверка и извршување код.
- Вклучува уредувач, компајлер и дебагер.
- Помага во откривање и отстранување грешки.
- Овозможува брз и визуелен преглед на резултатите.
- Идеално за почетници и за професионалци.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Инсталирај Visual Studio Code и напиши програма во Python!
- Испробај ја функцијата дебагирање и користи breakpoints!
- Истражи како работи Git интеграцијата во IDE!
- Спореди две IDE и запиши ги разликите!

3.9 Пишување, извршување и дебагирање програма

Веќе знаеш да...

- објасниш што е интегрирана развојна околина,
- разликуваш основни програмски јазици,
- наведеш примери за IDE.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

1. Зошто служи IDE?

2. Која околина за C++ ја знаеш?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што е дебагирање,
- компајлираш.

Процесот на создавање програма не завршува со напишаниот код. Напротив, пишувањето е само првиот чекор во поширокиот процес на развој на софтвер. Програмата треба да биде компајлирана, извршена и проверена дали се однесува како што се очекува. Вообичаено, првата верзија на кодот содржи грешки – синтаксички, логички или грешки при извршувањето. Затоа, важна фаза во програмирањето претставува **дебагирањето** – процесот на откривање и отстранување грешки во кодот. Разбирањето на оваа целина е клучно за секој што учи да програмира, бидејќи го води од теорија кон практична примена.

Пишувањето програма започнува со употреба на **уредувач на текст** кој е дел од IDE. Се внесуваат инструкции на избраниот програмски јазик, притоа внимавајќи на синтаксата, распоредот на елементите и правилната логика. Кодот треба да биде читлив, уреден со празни редови, со коментари кои објаснуваат делови од логиката. Со правилно пишување се намалува веројатноста за грешки и се олеснува подоцнежната анализа.

ПРИМЕР 1

При пишување на C++ програма, ако заборавиме да ставиме ; на крајот од линија, ќе добиеме синтаксичка грешка при компајлирање. IDE како Code::Blocks автоматски ќе ја истакне погрешната линија и ќе прикаже порака: "expected ';' before 'return'".

По пишувањето, кодот мора да се **компајлира** – тоа значи да се преведе од изворен код во машински разбирлив код. Овој процес проверува дали има синтаксички грешки, како погрешно напишани клучни зборови, заборавени загради или неправилни изрази. Ако постојат грешки, компајлерот прикажува пораки кои укажуваат на редот и видот на грешката. Учениците учат да ги читаат и да ги толкуваат овие пораки, што е дел од секојдневието на секој програмер.

ПРИМЕР 2

Ако се напише наредба каде во позиција за цел број ќе сместиме текст, компајлерот ќе пријави грешка бидејќи текст не може да се додели на целобројна променлива. IDE ќе истакне: "invalid conversion from 'const char*' to 'int'".

Откако е успешно компајлирањето, следува **извршување на програмата**. Во оваа фаза, кодот се стартува и го покажува резултатот на екранот. Ако програмата не се однесува како што се очекува, можно е да постои **логичка грешка** – програмата се извршува без проблеми, но не го дава точниот резултат. Овие грешки се најтешки за откривање, бидејќи компајлерот не ги детектира.

ПРИМЕР 3

Ако програмата треба да пресмета производ, но во кодот се користи + наместо *, резултатот ќе биде погрешен, без предупредување. Програмата ќе работи, но нема да го прави тоа што се бара.

Дебагирањето е процес со кој се откриваат и анализираат ваквите логички или скриени грешки. Се користат алатки како **breakpoints** – точки во кодот каде извршувањето паузира – и **step-by-step** функција за да се следи текот на променливите. IDE овозможуваат визуелно прикажување на вредностите и лесно движење низ редовите за проверка. Оваа техника им помага на учениците да разберат што точно се случува во нивниот код.

ЗАКЛУЧОК

Пишувањето програма не завршува со кодирањето. Секоја програма мора да биде внимателно компајлирана, тестирана и – по потреба – дебагирана. Грешките не треба да се доживуваат како неуспех, туку како природен дел од процесот. Со секоја исправена грешка се учи нешто ново. Совладувањето на дебагирањето значи совладување на една од најважните вештини во програмирањето.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Наведи пример за синтаксичка грешка!

2. Што претставува порака за грешка од компајлерот?

3. Опиши како се врши дебагирање чекор по чекор!

4. Кои IDE алатки се користат за дебагирање?

5. Зошто некои грешки не се откриваат при компајлирање?

Запомни што научи!

- Пишувањето код бара внимание на синтакса и логика.
- Грешките се нормален дел од процесот.
- Компајлерот проверува синтакса.
- Извршувањето покажува дали резултатот е точен.
- Дебагирањето е процес на откривање и решавање грешки.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Напиши кратка програма со намерна логичка грешка и обиди се да ја пронајдеш!
- Истражи како функционира дебагирање во Visual Studio Code!
- Научи како се поставува breakpoint!
- Преведи пораки за грешка од англиски на македонски и објасни што значат!

3.10 Исказ во програмирањето

Веќе знаеш...

- да препознаеш основни елементи на програмска околина,
- да разликуваш уредувач, компајлер и прозорец за извршување,
- што значи да се добие резултат на екран.

ЗАДАЧИ И ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

1. Објасни што значи да се компајлира програма!

2. Кои се главните видови грешки во програмирањето?

3. Зошто е важно дебагирањето?

4. Како се користи breakpoint?

5. Што значи логичка грешка?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објасниш што е исказ во програмирањето, да разликуваш исказ за приказ на екран од други наредби,
- разбереш како комуницираат програмите со корисникот.

Во светот на програмирањето, секоја наредба што ја задаваме на компјутерот, се нарекува исказ. Исказите се основни градивни елементи на секоја програма – тие му кажуваат на компјутерот што точно да направи. Преку нив се изведуваат различни акции: пресметување, зачувување на вредности, или – што е најважно за оваа лекција – прикажување пораки и резултати на екранот. Приказот на информација е првата форма на интеракција меѓу програмата и корисникот. Кога ќе напишеме програма, важно е

не само таа да работи, туку и да може да покаже што направила. Токму затоа, учењето за исказите за приказ на екран претставува клучен чекор во развивање на логичко и комуникативно размислување кај учениците.

Во секоја програма што комуницира со човек, постои потреба да се прикаже информација – било тоа да е резултат од пресметка, упатство, предупредување или порака за успех. За таа цел, се користат посебни типови искази кои овозможуваат испраќање пораки до екранот. Со нив програмата станува разбирлива и корисна за човекот што ја користи. Кога ќе се изврши ваков исказ, на екранот се појавува точно она што било наведено – текст, број или комбинација од двете. Овој приказ е визуелен доказ дека нешто се случило во програмата. На пример, можеме да прикажеме резултат од некоја математичка операција, или едноставно да испратиме поздравна порака.

Покрај самостојни пораки, често се прикажуваат вредности што претходно биле пресметани или внесени. Во таков случај, програмата не само што го извршила бараниот пресметковен чекор, туку и му го соопштува резултатот на корисникот. Ова е исклучително важно за проверка – кога корисникот гледа што е прикажано, може да оцени дали програмата работи како што треба. На тој начин се создава интеракција – корисникот внесува нешто, програмата обработува и потоа прикажува резултат. Без оваа последна фаза, програмата би изгледала „нема“, без начин да покаже што се случува во неа.

Многу е важно да се разбере дека редоследот на исказите во програмата го одредува редоследот во кој се прикажуваат пораките. Програмите се извршуваат одозгора надолу, по ред, освен ако не е наведено поинаку. Тоа значи дека ако сакаме прво да се прикаже едно, а потоа нешто друго, тие искази мора да бидат подредени правилно. Ако се замени нивниот редослед, и резултатот на екранот ќе биде различен. Со ова учениците учат како „мисли“ програмата – нејзината логика се следи строго и точно и секоја промена има последици во излезот што го гледаме.

Понекогаш, сакаме пораките да се прикажуваат во посебни редови, една под друга. За таа цел, постојат механизми со кои се означува нов ред. Со нив може да се контролира изгледот на текстот на екранот – дали ќе биде во една линија или распоредено во повеќе, дали ќе има празно место или ќе се појави сè заедно. Ова е корисно за визуелно уредување на информацијата – така што излезот изгледа уредено, читливо и пријатно за гледање. Токму преку вакви детали се учи дека програмирањето не е само логика, туку и комуникација со човекот што го гледа резултатот.

Исказите за приказ се меѓу првите што се учат, бидејќи веднаш даваат видлив резултат. Тие носат чувство на постигнување – кога ќе се појави порака, ученикот знае дека програмата работи. Со нив почнува разбирањето на динамиката на програмскиот тек, бидејќи прикажаните пораки и резултати се директен одраз на тоа што се случува „во заднина“. Преку нив се постигнува и повратна врска – ако нешто не е прикажано правилно, тоа може да укаже на логичка грешка. Затоа, овие искази не се само алатка за приказ, туку и средство за проверка и подобрување на програмата.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што претставува еден исказ во програмирањето?

2. Зошто е важен исказот за приказ на екран?

3. Како дознава корисникот што направила програмата?

4. Што се случува ако се смени редоследот на исказите?

5. Направи сценарио со две пораки што мора да се прикажат во точен редослед!

6. Опиши како би изгледал излезот ако сите пораки се прикажат во ист ред!

7. Објасни зошто приказот на екран е важен дел од секој алгоритам!

Запомни што научи!

- Исказите се инструкции што програмата ги извршува по редослед.
- Исказот за приказ овозможува комуникација со корисникот.
- Програмата ги прикажува резултатите или пораките преку екран.
- Важно е како се редат исказите – од тоа зависи каков ќе биде излезот.
- Визуелното уредување на излезот помага да се разбере што се случува во програмата.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Истражи како се прикажуваат пораки во мобилни апликации!
- Замисли програма што покажува мени и објасни како би изгледал редоследот на прикажување!
- Напиши сценарио во кое прикажаната порака зависи од некоја претходна вредност!
- Размисли како би изгледал еден тест со повеќе прикажани прашања и резултати!

3.11 Псевдојазик

Веќе знаеш да...

- разликуваш бројчени и текстуални вредности,
- изведуваш едноставни математички операции,препознаваш што значи влез и излез на податоци, работиш со прости логички чекори во задача, знаеш дека променливите служат за чување податоци.

ЗАДАЧИ И ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

1. Зошто е важно прикажаната информација да биде уредна?

2. Замисли програма што прикажува поздравна порака! Објасни што прави секој чекор!

3. Осмисли ситуација каде што треба да се прикаже резултат од операција! Опиши ја логиката!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- препознаеш и запишеш валидна наредба во псевдојазик, декларираш променливи и доделуваш вредности, извршиш влез и излез со внеси и прикажи, примениш аритметички операции во псевдојазик, напишеш цел алгоритам со неколку чекори.

Псевдојазикот е поедноставена, описна форма на програмирање која се користи како чекор пред вистинското кодирање. Со негова помош се учи логиката на програмирање, без да се заглави во деталите на синтаксата. Тој не се извршува директно на компјутер, туку служи за планирање и објаснување на чекорите што би ги следела една програма. Особено е корисен кога се учи програмирање за првпат. Преку псевдојазикот, учениците добиваат претстава за тоа како работи кодот зад екранот – како се примаат податоци, како се обработуваат и како се добива резултат.

Доделување вредност:

Првата и најосновна операција што се користи во секоја програма е доделувањето вредност. Во псевдојазикот, тоа се прави со стрелка ←, која означува дека вредноста од десната страна треба да се смести во променливата од левата страна. На пример:

променлива цел број a ← 10

Ова значи дека се креира променлива по име **a**, која чува цел број и има почетна вредност 10. Оваа наредба може да се користи за иницијализација, но и за ажурирање на вредноста. Преку ова се воведува концептот на складирање податоци, што е темел на секоја понатамошна пресметка.

Влез на податоци:

Потребно е програмата да може да прима информации од корисникот. За таа цел се користи наредбата „внеси", која дозволува внесување вредности од тастатура. Така се прави програмата интерактивна. На пример:

внеси broj

Ова значи дека корисникот ќе внесе некоја вредност, која потоа ќе се запише во променливата broj. Без влезни податоци, програмата би работела само со однапред дефинирани вредности, што ја ограничува нејзината применливост. Наредбата „внеси" претставува чекор кон динамичко однесување на алгоритмите.

Излез на податоци:

Следен важен елемент е приказот на податоците. Со наредбата „прикажи", програмата може да испрати порака или резултат кон корисникот. На пример:

прикажи "Вредноста е: ", broj

Оваа наредба ќе прикаже текстот „Вредноста е: " заедно со моменталната вредност на broj. Излезот е неопходен за да се види резултатот од пресметките или да се потврди дека сè функционира како што треба. Без излез, корисникот не би имал увид во работата на програмата.

Аритметички операции:

Со доделени вредности и внесени податоци, може да се прават пресметки. Псевдојазикот поддржува основни аритметички операции како +, -, *, /. Пример за собирање:

променлива цел број zbir ← a + b

Ова значи дека се собираат вредностите на **a** и **b**, а резултатот се чува во zbir. Аритметичките операции се основа за математички задачи, пресметки со оценки, броење, статистика и многу повеќе. Со нив програмите добиваат можност да обработуваат информации.

Преку комбинирање на влез, излез, доделување и аритметика се добива функционален алгоритам. На пример:

прикажи "Внеси два броја:"

внеси x, y

променлива цел број proizvod ← x * y

прикажи “Производот е: “, proizvod

Овој алгоритам овозможува корисникот да внесе два броја, по што програмата ќе ги помножи и ќе го прикаже резултатот. Преку вакви примери се учи целосна структура на решавање проблеми преку алгоритамски начин на размислување. Овие знаења ќе бидат основа за следни посложени теми како услови, циклуси и функции.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што означува симболот ← во псевдојазик?

2. Кој збор се користи за внесување вредност од тастатура?

3. Објасни што прави наредбата „прикажи“!

4. Напиши алгоритам што ќе го прикаже квадратот на внесен број!

5. Побарај од корисникот да внесе три броја и прикажи го нивниот збир!

6. Направи алгоритам што ќе ја пресмета вредноста на (x + y) / 2!

7. Напиши алгоритам што ќе прикаже: “Добредојде, [име]”, каде име е внесено од корисникот.

Запомни што научи!

- Во псевдојазикот се користат едноставни и интуитивни наредби.
- Променливите се создаваат со променлива, а вредност се доделува со ←.
- Со „внеси“ се примаат податоци од корисникот, а со „прикажи“ се прикажуваат резултатите.
- Може да се изведуваат основни математички операции: +, -, *, /.
- Комбинацијата на влез, пресметка и излез формира функционален алгоритам.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Ако сакаме уште повеќе да ја формализираме логиката, можеме да воведеме псевдојазик со структури како ако ... тогаш, повторувај, и услови од типот x > y. Овие конструкции овозможуваат донесување одлуки и повторување на дејства. Исто така, може да се користат и псевдокод формати кои се приближни до вистински програмски јазици (како Python или C++), но без синтаксички правила. Ова го прави псевдојазикот мост помеѓу логичкото размислување и практичното програмирање.

3.12 Секвенца од искази

Веќе знаеш да...

- напишеш едноставен исказ за приказ на екран,
- ги разбереш поимите псевдојазик и инструкција,
- ја следиш логиката на основен програмски тек.

ЗАДАЧИ И ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

1. Како се запишува множење на две променливи?

2. Зошто е важно да се комбинираат влез, аритметика и излез?

3. Напиши псевдојазик што ќе пресмета разлика на два броја!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- креираш различен тип променливи.

Во програмирањето, секвенца од искази претставува група на инструкции кои се извршуваат по утврден редослед – една по друга, без условување или прескокнување. Тоа е основен механизам за градење програма, бидејќи овозможува дефинирање на јасни, логички чекори. Кога една програма се чита од почеток до крај, секој исказ извршува одредена функција и тоа во редот како што се појавува во кодот. Оваа последователност се нарекува секвенцијално извршување, а самите инструкции го формираат секвенцијалниот блок. Секвенцата е почетната точка од која понатаму се надградуваат програмските логики преку услови и циклуси и затоа е од суштинско значење нејзиното точно разбирање.

Секој исказ во програма претставува наредба која компјутерот ја извршува. Кога тие искази се наредени еден под друг, без дополнителни контроли како услови или повторувања, тие ја сочинуваат секвенцата. Во ваков блок, редоследот е исклучително важен – компјутерот ги извршува инструкциите линеарно, од горе надолу. Ова значи дека резул-

татот од една наредба може да влијае на следната. Доколку некоја наредба се постави предвремено, пред да се подготват потребните податоци, може да се појави грешка или несакан резултат. Затоа се препорачува програмирањето да се започне со пишување мали, логично поврзани секвенци од инструкции.

ПРИМЕР 1

```
променлива цел број a ← 10
променлива цел број b ← 5
променлива цел број zbir ← a + b
прикажи "Збирот е: ", zbir
```

Во овој пример, секоја линија зависи од претходната. Ако се промени редоследот, на пример ако се постави cout пред изразот zbir ← a + b, ќе се добие грешка. Затоа секвенцата треба да се планира внимателно.

Добро организирана секвенца овозможува лесно читање на кодот и предвидливо извршување. Во образовен контекст, ова значи дека учениците учат да размислуваат чекорно, логички и да разберат како се преминува од една операција кон друга. На тој начин, се развива не само програмска, туку и **аналитичка способност**. Ова е корисно не само за кодирање, туку и за решавање проблеми од секојдневниот живот, каде што исто така важи принципот на логичка последователност.

ПРИМЕР 2

```
прикажи "Внеси две цели броја: "
внеси x, y
променлива цел број proizvod ← x * y
прикажи "Производот е: ", proizvod
```

Во оваа секвенца, корисникот прво внесува вредности, потоа се прави пресметка и најпосле се прикажува резултатот. Секоја линија мора да дојде во точен ред за да се добие валиден резултат.

Покрај тоа, секвенцата овозможува **модулирање на логиката** – т. е. групирање на операции кои имаат заедничка намена. Така, може да се создадат логички блокови кои лесно се разбираат и се тестираат. Поради тоа, програмерите често ја организираат програмата во **функционални сегменти**, секој составен од сопствена секвенца. Дури и кога ќе се премине на посложени структури како услови и циклуси, секој дел внатре во нив ќе биде формиран од секвенца на инструкции.

ПРИМЕР 3

променлива цел број m ← 8
променлива цел број n ← 3
променлива цел број razlika ← m - n
прикажи “Разликата е: “, razlika

Повторно, наредбите мора да се извршат по ред. Ако се обидеме да користиме razlika пред нејзино пресметување, ќе се појави грешка.

ЗАКЛУЧОК

Секвенцата од искази е темелот на секоја програма. Разбирањето како се извршуваат инструкциите по ред претставува клучен чекор во учењето на програмирањето. Без овој концепт, не може да се премине на посложени логики. Затоа, учениците мора да научат не само да пишуваат инструкции, туку и да ги организираат логично и смислено во секвенца. Ова ќе им овозможи да создаваат стабилни, функционални и читливи програми.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Напиши програма која чита два цели броја и го испишува нивниот збир и производ!

2. Направи секвенца во која прво се внесуваат вредности, потоа се прави некоја пресметка, па се прикажува резултатот!

3. Промени го редоследот на инструкциите и спореди го излезот!

4. Зошто е важно да не се користи променлива пред да се дефинира?

Запомни што научи!

- Секвенцата претставува логички поврзани инструкции.
- Секој исказ се извршува по ред, без прескокнување.
- Грешки често настануваат поради нарушен редослед.
- Организирањето на кодот во логични блокови го олеснува читањето.
- Добро напишана секвенца е темел на стабилна програма.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Истражи како се применува секвенцијално извршување во роботиката!
- Препознај секвенци во секојдневни задачи: готвење, пешачење до училиште!
- Обиди се да напишеш програма што користи три различни секвенци!
- Испробај што се случува ако се прекине секвенцата во средина!

3.13 Променливи и константи

Веќе знаеш да...

- препознаеш редоследно извршување,
- напишеш секвенца од искази.

ЗАДАЧИ И ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

1. Објасни што е секвенца од искази!

2. Зошто е важен редоследот на инструкциите?

3. Напиши три инструкции што се поврзани логички!

4. Што ќе се случи ако се разместат инструкциите?

5. Кои се предностите на секвенцијално извршување?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- креираш променливи и константи,
- доделиш почетна вредност на променливи,
- разликуваш начин на функционирање кај променливите и константите.

Во секоја програма се користат информации кои се менуваат или остануваат исти. За да може компјутерот да работи со овие информации, тие се зачувуваат во таканаречени **променливи** или **константи**. Променливите служат за чување податоци кои може да се менуваат за време на извршување на програмата, додека константите се користат за податоци чии вредности не смее да се менуваат. Овие две структури се основа на секоја логика во програмирањето и овозможуваат флексибилно управување со информациите.

Променливата е именувано место во меморијата на компјутерот кое се користи за чување вредност што може да се менува. Секоја променлива има **име (идентификатор), тип на податок и вредност**. Името служи за препознавање, типот укажува какви вредности може да содржи, а самата вредност е конкретната бројка или текст. Пред да се користи, променливата мора да биде **декларирана**, односно да ѝ се каже на

програмата каков податок ќе чува. Дополнително, може веднаш да ѝ се даде почетна вредност – тоа се нарекува **иницијализација**.

ПРИМЕР 1

```
int vozrast = 16;
```

Во овој пример, се дефинира променлива по име „vozrast“ која содржи цел број. Типот int укажува дека вредноста што може да биде зачувана во оваа променлива е цел број, односно број без децимали. Почетната вредност која ѝ се доделува на променливата е 16. Оваа вредност се нарекува иницијална или почетна вредност и може да се промени во текот на извршувањето на програмата, во зависност од потребите на задачата или од интеракцијата со корисникот. Кога еднаш ќе се декларира променливата „vozrast“, таа станува достапна и може да се користи во различни делови од програмата, секаде каде што е потребно.

Откако променливата е веќе дефинирана, таа може слободно да се користи за различни операции и пресметки. На пример, ако на корисникот му се овозможи да внесе нова возраст, вредноста на променливата може лесно да се ажурира со внесената вредност. Исто така, променливата може да се користи за математички пресметки, како додавање, одземање или споредување со други променливи. Вредноста што ја содржи променливата може во секој момент да биде прикажана на екран, што ја прави програмата поинтерактивна и подинамична. Со користење на променливи како оваа, програмите добиваат поголема флексибилност и можност за прилагодување според ситуацијата или корисничкиот внес. Токму затоа променливите претставуваат основен елемент во програмирањето, бидејќи овозможуваат складирање и промена на информации за време на извршувањето на програмите.

ПРИМЕР 2

```
int const broj = 8;
```

Во овој пример се дефинира константна променлива со име „broj“ која содржи цел број. Типот на оваа променлива е int, што значи дека вредноста мора да биде цел број, а спецификаторот const укажува на тоа дека еднаш доделената вредност на оваа променлива не може да се промени во текот на извршувањето на програмата. Почетната вредност во овој случај е 8 и таа останува фиксирана и непроменлива за време на целиот тек на програмата. Користењето на вакви константи е особено корисно кога вредностите што треба да се користат во програмата се однапред познати и не треба никогаш да бидат променети.

Константните променливи се многу важни затоа што ја прават програмата почитлива, јасна и безбедна. Бидејќи вредноста на константата не може да се промени по нејзината иницијализација, програмерот е сигурен дека нема да дојде до случајна или ненамерна промена на нејзината вредност, што ја спречува можноста за грешки или недоразбирања во кодот. Константите се корисни за де-

▸▸▸

финирање фиксни вредности како што се математичките константи (на пример, π или гравитациската константа), но и за параметри кои остануваат исти низ целата програма, како број на денови во неделата или број на месеци во годината. Користењето на вакви константи придонесува и за полесна промена на вредностите во иднина, бидејќи тие се декларираат само на едно место во кодот. Ова овозможува полесно одржување и менаџирање на програмите, посебно ако програмата е сложена и има многу редови код.

ЗАКЛУЧОК

Променливите и константите се клучни градивни елементи на секоја програма. Тие овозможуваат складирање и управување со податоци, било тие да се менливи или фиксни. Преку нив се реализираат многу логички операции, пресметки и интеракции со корисникот. Разбирањето на тоа како се декларираат, користат и заштитуваат променливите и константите е суштински чекор во процесот на учење програмирање. Тие претставуваат алатки преку кои програмата „запомнува" и обработува информации.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Декларирај променлива висина и додели ѝ вредност!

2. Креирај константа MAX_OCENKA и постави ја на 5!

3. Што ќе се случи ако се обидеш да ја промениш константата?

4. Зошто е важно променливите да имаат описни имиња?

Запомни што научи!

- Променливите се користат за вредности што се менуваат.
- Константите чуваат вредности кои не се менуваат.
- Секоја променлива има тип, име и вредност.
- Иницијализација е доделување почетна вредност.
- Константите се декларираат со const и не може да се променат.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Истражи како се користат променливи во игри!
- Обиди се да направиш програма што користи и променливи и константи!
- Прочитај што се случува ако користиш променлива без иницијализација!
- Спореди како различни јазици (C++, Python, Java) се справуваат со константи • Напиши код во кој променливата ја менува вредноста повеќе пати!

3.14 Типови променливи

Веќе знаеш да...

- разликуваш променливи од константи,
- разликуваш текст и број во програма.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

1. Што е променлива?

2. Што е иницијализација на променлива?

3. Кога се користи константа наместо променлива?

4. Дали е можно да се промени вредноста на константа?

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- креираш различен тип променливи.

Во компјутерското програмирање секој податок што се користи мора да припаѓа на одреден **тип**. Типот на податокот ѝ кажува на програмата колку меморија треба да одвои и како да го обработи тој податок. На пример, целиот број 5 и текстот „5“ изгледаат исто на екран, но се сосема различни податоци. Токму затоа, изборот на правилен **тип променлива** е суштински чекор при пишување точна и функционална програма. Основните типови со кои најчесто се среќаваме се: цели броеви (int), децимални броеви (double или float), знаци (char) и логички вредности (bool).

ПРИМЕР 1

```
double prosek = 4.75;
```

Овде променливата prosek ја чува оценката на ученик/чка, што е децимален број. Ако се користел int, вредноста 4,75 би се заокружила на 4. Овде не станува збор за заокружување како во математиката, туку за „отсекување“ на децималниот дел. Всушност, во овој случај се зема само целиот број.

Друг интересен тип е char, кој се користи за **поединечни знаци**, како што се букви или специјални симболи. Овој тип е идеален кога се обработуваат иницијали, ознаки или симболи. Се декларира со апострофи околу знакот ('A', '%', '#'). Со него не се вршат математички операции, туку се користи за симболичко претставување.

ПРИМЕР 2

```
char bukva = 'A';
```

Типот на податок char служи за складирање поединечни знаци, како букви, цифри или симболи. На пример, char bukva = 'A'; дефинира променлива со име „bukva" која ја чува вредноста A. Знаците во C++ секогаш се запишуваат во единечни наводници. Овој тип се користи при работа со текстуални податоци на ниво на поединечни знаци или за проверка и споредба на конкретни карактери во програмата.

ПРИМЕР 3

```
string ime = "Ana";
```

Во овој пример се дефинира променлива од тип string (низа од знаци) со име „ime". Овој тип на променлива се користи за чување текстуални податоци, односно зборови, фрази или реченици. Во овој конкретен случај, променливата „ime" добива почетна вредност „Ana", што значи дека кога ќе се повика оваа променлива во текот на програмата, таа ќе содржи токму ваков текстуален податок. Типот string е многу корисен кога треба да се зачува и да се обработува текстуална информација, како што се имиња, адреси, пораки и други слични податоци кои се составени од еден или повеќе знаци.

Откако еднаш ќе се декларира, вредноста на променливата „ime" може да се промени по потреба, односно може да се додаде друг текст или да се направат промени во постоечкиот текст. На пример, доколку корисникот внесе ново име, вредноста на променливата ќе се ажурира со новата информација. Покрај тоа, текстуалните променливи овозможуваат и извршување различни операции, како што се спојување на повеќе зборови (конкатенација), споредба на текстови, пребарување на карактери во низата и многу други активности. Ова овозможува програмите да бидат многу подинамични и пофлексибилни, затоа што текстуалните информации се клучни за комуникацијата меѓу корисникот и компјутерот.

ПРИМЕР 4

```
bool aktiviran = true;
```

Во овој пример, се дефинира логичка променлива од тип bool со име „aktiviran". Логичките променливи од типот bool можат да имаат само две можни вредности: true (точно) или false (неточно). Во дадениот случај, променливата е иницијализирана со вредноста true, што значи дека условот што го претставува оваа променлива е во моментот исполнет или активен. Логичките променливи имаат значајна улога во контрола на текот на програмата, бидејќи од нивната вредност зависи како ќе се одвива понатамошното извршување на инструкциите во програмата.

Користењето на логичките променливи е неопходно во ситуации кога треба да се провери дали некој услов е исполнет или не. На пример, променливата „aktiviran" може да укажува дали корисникот има дозвола за пристап до некој дел

▸▸▸

▸▸▸
од програмата. Доколку оваа променлива ја има вредноста true, програмата ќе продолжи да извршува одредени операции. Ако пак, вредноста е false, програмата ќе изврши некоја друга операција или ќе ја прескокне одредената постапка. На овој начин, логичките променливи му овозможуваат на програмерот да направи програмата да донесува одлуки во зависност од тековната состојба на условите, што ја прави програмата интерактивна, динамична и приспособлива за различни ситуации и потреби на корисниците.

ЗАКЛУЧОК

Типовите на променливи овозможуваат структура и логика во програмирањето. Со правилен избор на тип се постигнува точност, оптимизација на меморијата и разбирливост на кодот. Целите броеви (int), децималите (float и double), знаковите (char) и логичките вредности (bool) се основа на речиси секоја програма. Разбирањето на нивната намена и разликите меѓу нив е клучно за секој почетник во програмирањето.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Кој тип на променлива ќе го користиш за чување број на страници во книга?
2. Како се разликуваат int и double во прецизноста?
3. Што ќе се случи ако double се декларира со цел број?
4. За што служи char? Дај пример.
5. Кога се користи тип bool?

Запомни што научи!

- Секој податок има свој тип.
- int се користи за цели броеви.
- double се користи за децимали.
- char чува еден знак.
- bool чува логичка вредност: true или false.
- Правилен избор на тип значи подобра програма.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Истражи ги типовите float, long, short!
- Прочитај што се случува при пренос на вредности меѓу типови!
- Напиши пример во кој bool управува со текот на програмата!
- Дали char може да биде и број? Истражи!
- Како се претвора double во int?

3.15 Доделување вредност на променлива

Веќе знаеш да...

- разликуваш променливи според нивниот тип,
- препознаваш цел број, децимален број, знак и логичка вредност,
- поведеш нова променлива со соодветно име и тип.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Декларирај променлива возраст од тип int.!

2. Напиши double променлива за тежина!

3. Креирај char променлива за иницијал!

4. Дефинирај bool што ќе провери дали е бројот позитивен!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- доделуваш вредности на променливи,
- препознаеш преклопување на вредност.

Во светот на програмирањето, секоја променлива има не само име и тип, туку и вредност. Оваа вредност може да биде одредена при создавање на променливата или подоцна, при извршување на програмата. Постапката со која се поставува или менува вредноста на една променлива, се нарекува доделување вредност. За таа цел се користи операторот =. Иако изгледа како математичко еднакво, во програмирањето = значи „додели“. Тоа значи дека десната страна на изразот се пресметува и резултатот се зачувува во променливата лево.

Доделувањето вредност на променлива може да се изврши на неколку начини. Првиот е иницијализација, односно доделување почетна вредност уште при креирање на променливата. Овој начин е едноставен и го олеснува читањето на кодот. На пример, ако сакаме веднаш да ја поставиме почетната температура, можеме да напишеме:

ПРИМЕР 1

```
int temperatura = 22;
```

Во овој случај, променливата temperatura од тип int добива вредност 22 уште при создавањето. Ова се нарекува иницијализација и е најчестата практика кога вредноста е позната однапред.

Во случај кога променливата треба да добие вредност по некоја пресметка, таа прво може да биде декларирана без вредност, а потоа вредноста се доделува преку израз. На пример, ако сакаме да ја пресметаме просечната оценка од два предмета.

ПРИМЕР 2

Овде, prosek првично е само дефинирана, но подоцна ѝ се доделува вредност што произлегува од изразот. Ова покажува дека променливата може динамички да добива вредност, што е клучно за интерактивни програми.

Дополнително, вредноста на една променлива може да се промени повеќепати за време на извршување на програмата. Тоа значи дека променливите се флексибилни и нивната содржина може да се ажурира. Ова се нарекува **преклопување на вредност** и тоа е вообичаена појава, особено кога се работи со броење, акумулација или услови.

ПРИМЕР 3

```
int rezultat = 0;
rezultat = rezultat + 5;
rezultat = rezultat * 2;
```

Прво, rezultat добива вредност 0. Потоа, таа се зголемува за 5, па се множи со 2. Крајната вредност е 10. Ова е јасен пример на преклопување на вредност, кое овозможува изградба на посложени логики во кодот.

Важно е да се напомене дека левата страна од операторот = секогаш мора да биде променлива – нешто што може да ја смени вредноста. Десната страна, пак, може да биде каков било израз: броеви, други променливи, математички операции, па дури и функциски повици. Тоа го прави операторот = еден од најважните делови во секој програмски јазик.

ЗАКЛУЧОК

Доделувањето вредност на променлива претставува основна и незаменлива алатка во програмирањето. Со неа се постигнува флексибилност, пресметка и складирање податоци. Без оваа операција, променливите би останале празни и бесмислени. Разбирањето на разликата меѓу иницијализација, пресметка и преклопување е клучно за прецизна и стабилна работа на програмите.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што значи иницијализација на променлива?

2. Кога се користи операторот =?

3. Што ќе се случи ако се направи обид да се додели вредност на константа?

4. Објасни ја разликата меѓу x = x + 1 и x = 1!

5. Може ли десната страна од израз да вклучува повеќе променливи?

Запомни што научи!

- = не значи еднаквост, туку доделување.
- Вредноста се доделува оддесно кон лево.
- Променливата може да се промени повеќе пати.
- Доделување е основа на секој пресметковен алгоритам.
- Иницијализацијата овозможува чист и организиран код.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Истражи што е „chain assignment“: a = b = c = 5!
- Што се случува ако се доделува израз со различни типови?
- Обиди се да иницијализираш const променлива – што се случува?
- Дали x = x + 1 е легално математички? А програмски?
- Колку вредности може да има една променлива во текот на програмата?

3.16 Приказ на вредност на променлива

Веќе знаеш да...

- креираш и доделуваш вредности на променливи,
- користиш основни типови: цели, децимални и знаковни,
- ги разликуваш текстуалните и нумеричките податоци.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Декларирај и иницијализирај променлива godini со вредност 18!

2. Креирај променлива cena, додели ѝ вредност 120, па зголеми ја за 30!

3. Изрази ја формулата povrsina = dolzina * sirina!

4. Променлива zbir нека се добие како збир на broj1 и broj2!

5. Додели вредност на променлива по вчитување на податок од тастатура!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- прикажуваш вредности од променливи на екранот,
- користиш основни команди за испис,
- форматираш излез со специјални знаци,
- комбинираш текст со променливи во приказ,
- прикажуваш повеќе променливи во една наредба,
- разбираш како работи испис во нов ред или со табулација.

Приказот на вредност на променлива претставува едно од најважните средства за комуникација меѓу програмата и корисникот. Со негова помош, резултатите од пресметките, статусот на податоците или пораки за грешки може јасно да се прикажат на екранот. Ова се постигнува со користење на специјализирани наредби за испис, при што најчесто се користи cout во C++ програмскиот јазик. Иако се работи за едноставна операција, правилниот приказ на вредности е од огромно значење за читливоста, точноста и професионалноста на програмата.

Во програмите кои се изработуваат во c++ на почеток стои:

```
#include <iostream>
using namespace std;
```

#include <iostream> му кажува на компајлерот да ја вклучи стандардната библиотека за **влезно-излезни операции во C++**, која се наоѓа во заглавната датотека iostream (input-output stream). Со неа се овозможува користење на основни објекти како cin за внес од тастатура, cout за приказ на екран и cerr за прикажување грешки. Без оваа наредба, програмата не би знаела што значи cout или cin и би се појавиле грешки при компилација.

using namespace std; ѝ кажува на програмата дека ќе се користат имиња од **просторот на имиња std** (standard), без потреба секогаш да се пишува std:: пред нив. На пример, наместо да пишуваме std::cout, можеме да користиме само cout. Ова ја поедноставува синтаксата кај почетниците и го прави кодот полесен за читање, но во поголеми проекти е подобро да се користи std::cout за да се избегнат конфликти со други простори на имиња.

Најосновниот начин за прикажување вредност е преку функцијата cout. Името cout значи **"character output"** (излез на карактери), и е дел од просторот std, што значи дека се користи како std::cout, или само cout ако има using namespace std;. Таа му овозможува на програмерот да прикаже текст, бројки или вредности од променливи директно на екранот. Наредбата cout секогаш се користи со операторот <<, кој го насочува податокот кон екранот. На пример:

ПРИМЕР 1

n.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int broj = 7;
8      cout << "Vrednosta na brojot e: " << broj << endl;
9      return 0;
10 }
11
```

Оваа наредба на екранот ќе прикаже: Вредноста на бројот е: 7. Забележливо е дека и текстот и променливата може да се комбинираат во една линија, што овозможува динамичен и разбирлив приказ.

Во повеќето случаи, потребно е да се прикаже повеќе од една вредност, или, пак, да се организираат податоците во линија или табела. Затоа, се користат специјални карактери како \n за нов ред или \t за табулација. Овие таканаречени „escape" знаци не се прикажуваат директно, туку влијаат на изгледот на излезот. На пример:

ПРИМЕР 2

```cpp
#include <iostream>

using namespace std;

int main()
{
    double pi = 3.14159;
    cout << "Priblizna vrednost na  pi: " << pi << endl;
    return 0;
}
```

Излезот би бил: Приближна вредност на pi: 3.14159. Во посложени случаи, може да се користат дополнителни библиотеки како <iomanip> за да се контролира бројот на децимали или за усогласување на текстот.

ПРИМЕР 3

```cpp
#include <iostream>

using namespace std;

int main()
{
    int a = 5, b = 10;
    cout << "a = " << a << "\tb = " << b << endl;
    return 0;
}
```

Излезот ќе биде уреден како табела: a = 5 b = 10. Ова покажува дека \t ја разделува информацијата како табела, а endl го започнува новиот ред, исто како и \n.

Многу често, исписот се користи за проверка на вредности, за дебагирање или за известување на корисникот за одредени настани. Во вакви ситуации, од клучно значење е да се прикажат информациите јасно и кратко. Форматирањето може да се подобри со поставување фиксна ширина, заокружување на децимали или усогласување на податоците во колони.

Сите овие техники овозможуваат јасен и уреден приказ на информациите, што е од клучно значење не само за естетика, туку и за точност и навремено разбирање на резултатите од програмата. Тоа посебно важи при анализа на големи податоци, при работа со табели или при следење на логички текови.

ЗАКЛУЧОК

Приказот на вредност на променлива е неопходен дел од секоја програма. Тој претставува интерфејс меѓу корисникот и машината. Со негово правилно користење, програмата станува читлива, функционална и професионална. Комбинирањето на променливи, текст и специјални знаци овозможува флексибилен и приспособен излез, а неговото совладување е чекор поблиску до градење реални апликации.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што е функцијата cout и за што се користи?

2. Кој оператор се користи за приказ на вредност?

3. Што е разликата меѓу \n и endl?

4. Како може да се прикаже вредност од повеќе променливи во една наредба?

5. Кој е ефектот на \t во исписот?

Запомни што научи!

- Функцијата cout е основен начин за приказ на вредности.
- Операторот << се користи за насочување кон екран.
- Escape знаци како \n и \t го форматираат текстот.
- Може да се комбинираат текст и вредности.
- Приказот е важен за точна комуникација со корисникот.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Истражи ја библиотеката <iomanip> и функциите setprecision и setw!
- Обиди се да прикажеш резултати во табеларна форма!
- Користи испис за дебагирање – покажи вредности во секој чекор!
- Направи програмска форма со линии и заглавја!
- Прикажи текст со кирилични знаци во C++!

3.17 Аритметички операции и изрази

Веќе знаеш да...

- користиш променливи и да им доделуваш вредности,
- го прикажуваш резултатот на екран,
- разликуваш цели и децимални броеви,

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

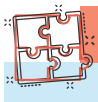
1. Декларирај променливи ime и godini, прикажи ги на екран!

2. Користи \n за приказ на вредности во нов ред!

3. Креирај променливи за висина и тежина, и прикажи ги со објаснување!

4. Прикажи три броеви по ред, разделени со табулација!

5. Користи cout за да прикажеш израз како 5 + 3 = 8!



ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- користиш аритметички операции: собирање, одземање, множење, делење,
- применуваш правилен редослед на операции,
- користиш загради за контрола на изрази,
- ја разбираш разликата меѓу целобројно и децимално делење,
- го користиш операторот за остаток,
- решаваш изрази што комбинираат повеќе операции.

Аритметичките операции се основа на речиси секоја пресметка што се прави во програмирањето. Тие овозможуваат манипулација со броеви – дали станува збор за собирање, одземање, множење или делење – и се применуваат во речиси сите логички решенија, од наједноставни до најсложени. Во оваа лекција ќе биде објаснето како се креираат и пресметуваат изрази, кои се операторите и како влијае редоследот на операции врз резултатот.

Аритметичките операции ги вклучуваат стандардните математички симболи: + за собирање, - за одземање, * за множење, / за делење и % за остаток при делење. Овие оператори се користат во комбинација со вредности и променливи за да се формираат изрази. Основниот израз може да изгледа вака: a + b * c, каде a, b и c се променливи.

ПРИМЕР 1

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       string ime;
8       int a = 5, b = 3, c = 2;
9       int rezultat = a + b * c;
10      cout << "Rezultatot e: " << rezultat << endl;
11      return 0;
12  }
13
```

Во овој случај, множењето има приоритет над собирањето, па се пресметува b * c = 6, а потоа a + 6 = 11.

Секогаш кога има повеќе операции во еден израз, мора да се земе предвид приоритетот на операторите. Правилото гласи: множење и делење имаат повисок приоритет од собирање и одземање. Ако сакате да го промените редоследот на пресметката, се користат загради.

ПРИМЕР 2

```cpp
#include <iostream>

using namespace std;

int main()
{
    int a = 5, b = 3, c = 2;
    int rezultat = (a + b) * c;
    cout << "Rezultatot e: " << rezultat << endl
    return 0;
}
```

Овде, прво ќе се изврши a + b = 8, а потоа 8 * c = 16, поради заградите.

Особено е важно да се разбере разликата меѓу целобројно и децимално делење. Кога се користат цели броеви, резултатот од делењето секогаш е цел број, без децимали. За да се добие точен децимален резултат, потребно е барем еден од броевите да е децимален (double или float тип).

ПРИМЕР 3

```cpp
#include <iostream>

using namespace std;

int main()
{
    int a = 5, b = 2;
    double rezultat = (double)a / b;
    cout << "Decimalen rezultat: " << rezultat << endl;
    return 0;
}
```

Овде, а е претворен во децимален број пред делењето, па резултатот ќе биде 2.5, а не 2.

Дополнително, операторот % се користи за да се добие остатокот при делење. На пример, 7 % 3 ќе врати 1, бидејќи 3 влегува два пати во 7, со остаток 1. Ова е особено корисно при решавање задачи поврзани со парност, броење, и проверка на деливост.

ЗАКЛУЧОК

Аритметичките изрази се составен дел од логиката на секоја програма. Правилното разбирање на операторите, нивниот приоритет и користењето на загради овозможуваат точни и предвидливи резултати. Умеењето да се комбинираат операции и да се контролира редоследот на извршување е клучно за решавање задачи и градење реални апликации.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Кои аритметички оператори ги познаваш?

2. Кој оператор има највисок приоритет?

3. Кога се користат загради во изразите?

4. Објасни ја разликата меѓу a / b и (double)a / b!

5. Што ќе биде резултатот од 8 % 3?

Запомни што научи!

- Основни оператори се: +, -, *, /, %.
- Приоритетот на операции влијае врз резултатот.
- Загради се користат за да се промени редоследот.
- Целобројно делење не враќа децимали.
- Операторот % враќа остаток.
- Точноста зависи од типот на променливите.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Истражи ги функциите pow(), sqrt(), fabs() од <cmath>!
- Напиши израз што пресметува средна вредност на три броеви!
- Реши изрази со повеќе загради и операции!
- Направи миникалкулатор кој ги користи сите пет операции!
- Анализирај како влијаат грешките во редоследот врз точноста на резултатите!

3.18 Внесување податоци

Веќе знаеш да...

- креираш променливи,
- доделуваш вредности на променливи,
- го прикажуваш резултатот на екран.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

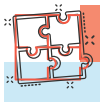
1. Пресметај го резултатот од 4 + 2 * 3 и објасни го редоследот!

2. Напиши израз кој ќе го собере x и y, а потоа ќе го подели со z!

3. Пресметај го остатокот од делењето на 25 со 6!

4. Претвори го a = 7 во децимален и подели го со 2!

5. Напиши израз каде a = (b + c) * d!



ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- користиш наредби за внесување податоци од тастатура,
- препознаеш различни типови на внесени податоци,
- спроведеш основна валидација на внесот,
- правиш разлика меѓу текстуален и нумерички внес,
- внесуваш повеќе податоци одеднаш,
- решаваш грешки при внес.

Во секоја интерактивна програма постои потреба корисникот да внесе податоци. Без овие податоци, програмата не би можела да биде флексибилна или корисна. Внесувањето информации претставува почетна точка на секој обработувачки процес, без разлика дали се работи за броеви, текст или комбинации. Преку овој процес, на програмата ѝ се овозможува пристап до вредности дефинирани од корисникот, што понатаму може да се користат за пресметки, условни одлуки и различни логички текови. Во оваа лекција ќе се научи како се внесуваат податоци, како се обработуваат и како се справува со можни грешки.

Основниот начин за внесување податоци од тастатурата се врши со користење на специјални функции или оператори кои читаат вредности и ги запишуваат во променливи. Во повеќето програмски јазици, како што е C++, за оваа намена се користи операторот cin. Со негово користење, на програмата ѝ се дава можност да „слуша“ од тастатура и да го смести внесеното во конкретна променлива. Ова е клучно за изградба на интерактивни апликации кои реагираат на внесени податоци.

ПРИМЕР 1

```cpp
#include <iostream>

using namespace std;

int main()
{
    int broj;
    cout << "Vnesi eden broj: ";
    cin >> broj;
    cout << "Go vnese brojot: " << broj << endl;
    return 0;
}
```

Во овој пример, програмата му овозможува на корисникот да внесе цел број, кој потоа се прикажува назад како потврда.

Важно е да се разбере дека типот на податокот што се внесува мора да одговара на типот на променливата во која се зачувува. Ако корисникот внесе текст кога се очекува број, ќе настане грешка. Ова ја потенцира потребата од основна проверка – валидација на внесот. Соодветните механизми за валидација зависат од програмскиот јазик и од сложеноста на апликацијата, но на почетно ниво, програмата може да биде дизајнирана така што ќе бара повторен внес ако се добие невалидна вредност.

ПРИМЕР 2

```cpp
#include <iostream>

using namespace std;

int main()
{
    string ime;
    cout << " Vnesi go tvoeto celosno ime : ";
    getline(cin >> ws, ime);  // ws отстранува можен претходен newline
    cout << "Tvoeto ime e: " << ime << endl;

    return 0;
}
```

Со getline() се овозможува внесување на целосен ред текст, вклучително и празни места помеѓу зборовите.

ЗАКЛУЧОК

Внесувањето податоци е основа на секоја интерактивна програма. Преку користење на оператори како cin, програмата комуницира со корисникот и реагира според внесените вредности. Разбирањето на типовите податоци, валидација и методи за читање повеќе податоци е од суштинско значење за пишување сигурни и корисни апликации. Правилното управување со влезните информации ја прави програмата стабилна и отпорна на грешки од страна на корисникот.

ПРАШАЊА ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што се случува ако во cin се внесе текст наместо број?

2. Како се внесуваат повеќе вредности одеднаш?

3. Зошто е важна валидацијата на внес?

4. Кој оператор се користи за внесување цел број?

5. Која е улогата на getline()?

Запомни што научи!

- За внесување податоци се користи cin.
- Вредностите мора да одговараат на типот на променливата.
- Валидацијата е потребна за да се избегнат грешки.
- Со getline() се чита цела линија текст.
- cin не чита празни места, затоа се користи getline за реченици.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Внеси два броја и пресметај го нивниот збир!

2. Побарај од корисникот да внесе број помеѓу 10 и 100. Повтори додека не внесе валиден број!

3. Внеси целосно име на ученик и прикажи го на екран!

4. Напиши програма што бара внес на број и проверува дали е парен!

5. Побарај три цели броеви и прикажи го нивниот просек!

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Истражи како се користи cin.fail() за проверка на невалиден внес!
- Напиши програма што внесува и проверува лозинка!
- Направи миниформулар за внес на име, возраст и адреса!
- Примени валидација за текстуален внес (на пример, минимум 3 карактери)!
- Експериментирај со внесување децимални броеви (double)!

3.19 Споредбени операции

Веќе знаеш да...

- напишеш едноставни C++ програми,
- користиш променливи,
- отпечатиш вредности на екран,
- напишеш основна математичка операција.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Напиши програма во која ќе пресметаш периметар на правоаголник со внесување на големината на страните од тастатура.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- користиш споредбени оператори во C++,
- разликуваш различни типови на споредбени релации,
- процениш дали изразот е вистинит или невистинит,
- комбинираш споредбени оператори со променливи,
- го користиш резултатот од споредба во програма.

Во секојдневниот живот, често се врши споредување на вредности: дали некој број е поголем од друг, дали две вредности се еднакви, или дали некој услов е исполнет. Исто така, во програмирањето, овие споредби се користат за да се донесуваат одлуки и да се контролира текот на програмата. Во оваа лекција ќе биде објаснето што се споредбени операции, како функционираат во програмскиот јазик C++, и како се применуваат во реални програмски примери.

Споредбените операции во програмирањето претставуваат основа за донесување одлуки. Преку нив може да се утврди дали два броја се еднакви, дали еден број е поголем или помал од друг, како и дали вредностите се различни. Во C++ постојат

шест основни споредбени оператори: == (еднакво), != (нееднакво), < (помало), > (поголемо), <= (помало или еднакво), >= (поголемо или еднакво). Овие оператори секогаш враќаат булова вредност, односно true (вистина) или false (невистина), што понатаму може да се користи во условни конструкции.

ПРИМЕР 1

```
#include <iostream>
using namespace std;

int main() {
    int a = 5, b = 10;
    bool rezultat = a < b;
    cout << "Rezultatot od a < b e: " << rezultat << endl;
    return 0;
}
```

Кога се споредуваат вредности во C++, типот на тие вредности игра важна улога. Ако се споредуваат цели броеви, споредбата е директна. Но, доколку се споредуваат децимални броеви или знаци, треба да се внимава на точноста и ASCII вредностите. При споредување на текстуални податоци (string) се користи посебна логика, која се базира на редоследот на знаците. На овој начин, може да се напишат програми кои споредуваат имиња, оценки, или други податоци од реалниот живот.

ПРИМЕР 2

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string ime1 = "Ana";
    string ime2 = "Bojan";
    cout << "Dali ime1 < ime2? " << (ime1 < ime2) << endl;
    return 0;
}
```

Многу често, резултатот од споредбена операција се зачувува во променлива од тип bool, која потоа се користи во други делови од програмата. На пример, врз основа на споредбата може да се испечати различна порака, да се смени текот на извршување или да се изврши одредена акција. Буловите вредности може да се комбинираат и со логички оператори за составување посложени услови, што е основа за следната лекција.

ПРИМЕР 3

```cpp
#include <iostream>
using namespace std;

int main() {
    int godina = 2025;
    bool prestapna = (godina % 4 == 0);
    cout << "Godinata e prestapna? " << prestapna << endl;
    return 0;
}
```

Важно е да се вежба со различни вредности и оператори за да се разбере логиката што стои зад споредувањата. Со зголемување на сложеноста на условите, се прави подготовка за наредните чекори во програмирањето – употреба на условни конструкции (if/else), циклуси и логички оператори. Во почетната фаза, најдобро е да се вежбаат едноставни споредби меѓу броеви и стрингови за да се усвои правилното пишување на операторите и очекуваните резултати.

ЗАКЛУЧОК

Споредбените оператори се неопходен дел од секоја програма што треба да донесе одлука. Тие овозможуваат проверка на односите меѓу вредностите и претставуваат основа за условни конструкции. Преку вежбање и практични примери, се создава разбирање на начинот на кој се донесуваат логички одлуки во програмирањето.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Претвори ги сите споредбени оператори во реченици со реален пример!

2. Вежбај со броеви од -100 до 100, користејќи <, >, ==, !=!

3. Провери што се случува кога споредуваш стрингови со исти први букви!

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што враќа операторот == во C++?

2. Кој оператор се користи за проверка дали две вредности се различни?

3. Што е bool променлива?

4. Каква вредност враќа споредбена операција?

5. Напиши програма која ќе провери дали бројот 15 е поголем од бројот 10!

6. Напиши програма која ќе провери дали две имиња се еднакви!

7. Напиши програма која ќе отпечати „ДА“ ако бројот е помал од 100!

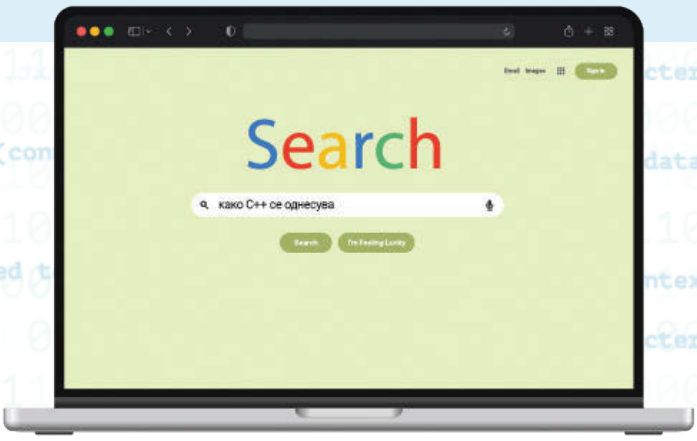
8. Напиши програма која ќе спореди два стринга и ќе отпечати кој е „помал“ лексикографски!

Запомни што научи!

- Споредбените оператори се користат за да се споредуваат вредности.
- Секој споредбен израз враќа true или false.
- Резултатот може да се зачува во bool променлива.
- Овие оператори се основа за условни конструкции.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи како се однесува C++ кога се споредуваат различни типови (на пример, int со double)! Обиди се да напишеш програма која ги комбинира споредбите со логички оператори, како && (и) и || (или), што ќе ги изучиме во следната лекција!



3.20 Логички изрази

Веќе знаеш да...

- користиш споредбени оператори (==, !=, <, >, <=, >=),
- провериш дали изразот е вистинит или невистинит,
- пишуваш прости услови во програма,
- добиеш булова вредност од споредба.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Како изгледа операторот за „помало или еднакво“?

2. Напиши програма која ќе провери дали еден број е парен!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- комбинираш повеќе услови користејќи логички оператори,
- објасниш што се логички изрази и како функционираат,
- користиш &&, ||, и ! во C++,
- ги разбереш правилата за приоритет на операции,
- идентификуваш кога се применува кратко пресметување.

Во програмирањето, често се среќаваат ситуации кога е потребно да се проверуваат повеќе услови истовремено. На пример, дали некој број е поголем од 10 и помал од 20, или дали корисникот внесува точна лозинка или точен ПИН-код. Во ваквите случаи, едноставна споредбена операција не е доволна. Наместо тоа, се комбинираат услови со т.н. логички оператори. Тие овозможуваат составување логички изрази што враќаат вистинитост или невистинитост врз основа на повеќе споредби.

Логичките оператори се основна алатка во секој програмски јазик кога се работи со услови. Во C++ постојат три главни логички оператори: && за логичка конјункција (и), || за логичка дисјункција (или), и ! за негација (не). Овие оператори се користат за комбинирање на едноставни услови во посложени логички изрази. На пример, ако сакаме да

провериме дали бројот x е меѓу 10 и 20, ќе напишеме x > 10 && x < 20. Овој израз ќе врати true само ако и двете споредби се вистинити. Ако барем една е лажна, целиот израз ќе биде невистинит.

ПРИМЕР 1

```
#include <iostream>
using namespace std;

int main() {
    int broj = 15;
    if (broj > 10 && broj < 20) {
        cout << "Brojot e megju 10 i 20." << endl;
    }
    return 0;
}
```

Кога се комбинираат услови, важно е да се внимава на приоритетот на операторите. Во C++, логичкиот оператор ! има највисок приоритет, потоа следува &&, а на крајот ||. Тоа значи дека ако во еден израз се користат сите три оператори, прво ќе се изврши негацијата, потоа конјункцијата, па дисјункцијата. За да се избегне нејаснотија, често се користат загради. На пример, изразот !(x < 10 || x > 20) ќе врати true само ако бројот x е во опсегот од 10 до 20 (вклучително). Заградите овозможуваат прецизна контрола на редоследот на пресметките.

ПРИМЕР 2

```
#include <iostream>
using namespace std;

int main() {
    int x = 25;
    if (!(x < 10 || x > 20)) {
        cout << "x e megju 10 i 20." << endl;
    } else {
        cout << "x e nadvor od opsegot." << endl;
    }
    return 0;
}
```

Една важна карактеристика на логичките оператори во C++ е краткото пресметување (short-circuit evaluation). Ова значи дека ако првиот услов во && е невистинит, вториот воопшто не се проверува, бидејќи целиот израз веќе е невистинит. Слично, ако првиот услов во || е вистинит, вториот не се проверува, бидејќи е доволно еден услов да биде вистинит за изразот да биде true. Ова не само што ја зголемува ефикасноста на програмата, туку и овозможува пишување услови кои без тоа би предизвикале грешки. Пример за тоа е проверка дали променливата е различна од нула пред да се изврши делење.

ПРИМЕР 3

```
#include <iostream>
using namespace std;

int main() {
    int broj = 0;
    if (broj != 0 && 100 / broj > 1) {
        cout << "Delenjeto e povekje od 1." << endl;
    } else {
        cout << "Delenjeto ne se izvrshi." << endl;
    }
    return 0;
}
```

Комбинирањето на логички изрази овозможува пишување услови кои се блиску до природниот јазик. На пример, проверка дали корисник е полнолетен **и** дали има лична карта, или дали има дозвола **или** придружба. Овие примери се пренесуваат во програми преку логички оператори и го прават кодот појасен. Дополнително, користењето на логички изрази во циклуси и услови е клучно за создавање реални и функционални програми. Преку нив, се реализира логиката што стои зад секоја автоматизација.

ЗАКЛУЧОК

Логичките изрази се неопходни за прецизна и условна контрола на извршување на програмите. Тие овозможуваат комбинирање на повеќе услови и рационално донесување одлуки во кодот. Правилното користење на операторите &&, || и !, како и вниманието кон приоритетот и краткото пресметување, се основа за сигурни и ефикасни програми.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Истражи како се комбинираат логички оператори со споредбени и како се користи во структури if и while! Напиши програма каде што се применуваат услови за логирање (внесено корисничко име и лозинка) и анализирај што се случува ако еден дел од условот не е точен!

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Кои се трите логички оператори во C++?

2. Што е кратко пресметување (short-circuit evaluation)?

3. Кој е приоритетниот редослед на логичките оператори?

4. Зошто се користи ! оператор?

5. Напиши програма што проверува дали број е позитивен и парен!

6. Напиши програма што проверува дали корисник има 18 или повеќе години или има дозвола!

7. Напиши програма што проверува дали стринг не е празен и има повеќе од 3 букви!

8. Напиши програма која со && и || комбинира 3 услови за број!

Запомни што научи!

- Логичките оператори &&, || и ! комбинираат повеќе услови.
- && враќа true само ако и двата изрази се вистинити.
- || враќа true ако барем еден израз е вистинит.
- ! ја негира вистинитоста на условот.
- Се применува кратко пресметување за ефикасност.

3.21 Структура на избор од две можности

Веќе знаеш да...

- користиш споредбени оператори,
- испишеш текст на екран,
- вршиш аритметички операции.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Напиши програма која користи ! за негација на услов!

2. Провери што се случува кога се комбинираат повеќе && и || без загради!

3. Напиши сопствени примери со вистински ситуации (на пр., услов за прием во училиште)!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- разликуваш што претставува избор од две можности,
- препознаваш кога треба да се користи структура на избор во програмирање,
- разбираш како логички услов влијае врз извршување на кодот,
- составуваш примери со if-else структура во C++,
- ги анализираш резултатите од различни логички услови.

Во секојдневниот живот, често се донесуваат одлуки меѓу две можности. Таквите ситуации се среќаваат и во светот на програмирањето. Структурата на избор од две можности овозможува компјутерот да донесе одлука врз основа на некој услов и да изврши еден од два можни блока на инструкции. Оваа лекција ја разгледува основната

логичка структура која му овозможува на програмерот да имплементира одлуки. Таа се нарекува if–else структура и е дел од условното управување со текот на програмата. Користењето на такви услови го прави кодот пофлексибилен и погоден за решавање реални проблеми.

Структурата на избор од две можности е еден од наједноставните, но многу важни контролни механизми во програмирањето. Со неа ѝ се овозможува на програмата да избере помеѓу две алтернативи. Ова се реализира со клучниот збор if, кој проверува дали некој логички услов е исполнет. Ако е вистинит условот, се извршува еден блок од инструкции; во спротивно, се извршува вториот блок, означен со else. Овој начин на контрола овозможува програмата да реагира различно во зависност од дадените податоци или ситуации. Одлуките во програмите се темелат на споредбени и логички изрази, за кои беше зборувано во претходната лекција.

ПРИМЕР 1

```
#include <iostream>
using namespace std;

int main() {
    int broj;
    cout << "Vnesi eden broj: ";
    cin >> broj;

    if (broj > 0)
        cout << "Brojot e pozitiven." << endl;
    else
        cout << "Brojot ne e pozitiven." << endl;

    return 0;
}
```

Во овој пример се прикажува избор меѓу две можности: дали бројот е поголем од нула или не. Врз основа на резултатот од условот, се печати соодветна порака.

Покрај едноставните споредбени услови, во структурата на избор од две можности може да се комбинираат повеќе услови со користење на логички оператори. Оваа можност го прави условот покомплексен и овозможува пофина контрола врз одлуките што ги донесува програмата. На пример, може да се испита дали кориснички внес е во зададен опсег или дали истовремено се исполнети два независни услова.

ПРИМЕР 2

```cpp
#include <iostream>
using namespace std;

int main() {
    int poeni;
    cout << "Vnesi poeni (0-100): ";
    cin >> poeni;

    if (poeni >= 51 && poeni <= 100)
        cout << "Polozeno." << endl;
    else
        cout << "Ne e polozeno." << endl;

    return 0;
}
```

Во примерот се проверува дали внесените поени се во рамките на минималниот праг за положување. Забележи дека е користен логички оператор && за да се постави опсег.

Често се поставува прашањето дали може да се комбинираат повеќе if-else изрази. Одговорот е да – кога има потреба од сложени одлуки, се вгнездуваат if структури или се користи структурата else if. Сепак, секогаш е добро да се одржи јасна и читлива структура на кодот, бидејќи тоа овозможува полесно одржување и разбирање на логиката на програмата.

ПРИМЕР 3

```cpp
#include <iostream>
using namespace std;

int main() {
    int broj;
    cout << "Vnesi cel broj: ";
    cin >> broj;

    if (broj % 2 == 0)
        cout << "Brojot e paren." << endl;
    else
        cout << "Brojot e neparen." << endl;

    return 0;
}
```

Програмата од примерот проверува дали еден број е парен или непарен. Овој тип проверка е чест во програмирањето кога се анализира видот на бројот или се одредува дали одредена операција треба да се изврши.

Веќе знаеш да...

- пресметуваш аритметички изрази и да користиш променливи;
- ја користиш командата cin и cout;
- ги разликуваш споредбените оператори и логичките вредности.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што претставува структурата if–else?

2. Кој дел од кодот се извршува ако условот не е исполнет?

3. Како се користат логичките оператори во if–else?

4. Што значи broj % 2 == 0?

5. Кога е подобро да се користи else if наместо повеќе if?

6. Напиши програма што ќе провери дали внесениот број е негативен или не!

7. Напиши програма што ќе провери дали ученик/чка има положено испит ако има над 60 поени!

8. Напиши програма што ќе провери дали температурата е во нормала (меѓу 36 и 37,5 степени)!

9. Напиши програма што ќе провери дали буквата е самогласка!

10. Напиши програма што ќе провери дали годината е престапна!

Запомни што научи!

- Структурата на избор од две можности (if–else) овозможува извршување на различен код зависно од логички услов. Таа е основен механизам за контрола на текот на програмата и се користи за изразување одлуки. Преку неа, програмата може да биде прилагодлива на различни ситуации и внесени вредности. Користењето споредбени оператори и логички изрази овозможува креирање на моќни услови. Примена на if–else структурата го прави кодот подинамичен и пофункционален во реални проблемски сценарија.

3.22 Задачи за вежбање од структура if else

Лесни задачи (1 – 7)

1. Напиши програма што ќе провери дали внесениот број е поголем од 100!

2. Напиши програма што ќе провери дали даден број е негативен!

3. Напиши програма што ќе провери дали внесениот број е делив со 5!

4. Напиши програма што ќе провери дали корисникот има наполнето 18 години!

5. Напиши програма што ќе провери дали два внесени цели броеви се еднакви!

6. Напиши програма што ќе испечати „Денес е викенд“, ако корисникот внесе „сабота“ или „недела“, а во спротивно „Работен ден“!

7. Напиши програма што ќе провери дали ученикот/чката има добиено оценка 5, ако бројот на поени е над 90!

Средно тешки задачи (8 – 14)

8. Напиши програма што ќе одреди дали внесен број е делив и со 3 и со 7!

9. Напиши програма што ќе провери дали бројот што го внел корисникот се наоѓа во интервалот од 10 до 99!

10. Напиши програма што ќе провери дали корисникот внесол правилна лозинка (да претпоставиме лозинката е „tajna123“)!

11. Напиши програма што ќе пресмета дали дадена личност има право да гласа, ако има најмалку 18 години и е државјанин!

12. Напиши програма што ќе испечати „парен и позитивен“ ако бројот е позитивен и парен, во спротивно „не ги исполнува условите“!

13. Напиши програма што ќе провери дали три страни можат да формираат триаголник!

14. Напиши програма што ќе прикаже „одлично“ ако бројот на поени е над 90, „добро“ ако е меѓу 70 и 90, а во спротивно „доволно“! (Користи if–else, но само за два исхода, третото нека биде default.)

Потешки задачи (15 – 20)

15. Напиши програма што ќе одреди кој од два внесени броја е поголем!

16. Напиши програма што ќе пресмета дали дадена температура е во границите на нормалата (36,0 до 37,5 степени)!

17. Напиши програма што ќе провери дали внесена буква е голема или мала буква!

18. Напиши програма што ќе провери дали дадена година е престапна!

19. Напиши програма што ќе провери дали внесен број е трицифрен и парен!

20. Напиши програма што ќе провери дали внесеното корисничко име и лозинка се точни (на пример: username = “admin”, password = “1234”)!

3.23 Внесување податоци

Веќе знаеш да...

- градиш едноставни условни искази,
- користиш if и else за донесување одлуки,
- креираш основни програми со влез и излез.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Како изгледа израз за проверка дали број е во опсег меѓу 10 и 20?

2. Препиши ги примерите и промени ги вредностите за да добиеш различен излез!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- препознаваш што значи еден исказ да биде „вгнезден“ во друг,
- разликуваш едноставни од вгнездени услови,
- користиш вгнездување за решавање посложени проблеми,
- напишеш програма со повеќекратно вгнездување,
- разбираш каде треба внимателно да се користи вгнездување.

Во процесот на создавање условни структури во програмирањето, често се среќаваат ситуации каде што е потребно да се направи проверка во рамките на друга проверка. Оваа појава, наречена вгнездување на искази, претставува моќен концепт кој овозможува креирање сложени логички одлуки. Вгнезден исказ е услов кој се наоѓа внатре во друг услов, што овозможува програмирање со повеќеслојна логика. Употребата на вакви структури се појавува често при моделирање реални проблеми каде што одредени дејства треба да се преземат само доколку се исполнат повеќе услови во одреден редослед. Вгнездените услови може да бидат и во облик на if-во-if, или комбинации на if и else if, каде што едната структура содржи друга.

Вгнездените услови се користат кога една одлука зависи од претходно донесена одлука. Во практични примери, често се случува да треба да се направи проверка дали некој број е позитивен, а потоа да се провери дали е парен. Двете проверки не се независни, бидејќи втората проверка нема смисла ако бројот не е позитивен.

Токму ваквите ситуации се решаваат со вгнездување. Се започнува со еден услов, а во неговото тело се поставува друг услов. На тој начин, вториот услов ќе се провери само ако првиот е вистинит, со што се избегнуваат непотребни или нелогични проверки. Ваквиот начин на организирање логика придонесува за појасен код и подобра контрола на текот на програмата.

```cpp
#include <iostream>
using namespace std;

int main() {
    int broj;
    cin >> broj;
    if (broj > 0) {
        if (broj % 2 == 0) {
            cout << "Brojot e pozitiven i paren." << endl;
        }
    }
    return 0;
}
```

Вгнездените услови не се ограничени само на две нивоа. Тие може да се организираат и во повеќе слоеви, особено кога се анализираат различни својства или се споредуваат повеќе променливи. Меѓутоа, колку повеќе услови се вгнездуваат, толку стануваат потешки читањето и одржувањето на кодот. Препорачливо е да се користи вгнездување со умереност и секогаш кога е логички оправдано. Се применува кога редоследот на проверките е битен – некои услови треба да се проверат само ако се исполнети претходните. На пример, ако треба да се испита дали ученик има доволен број поени за положување, а потоа дали има и доволен број часови, логично е второто да се провери само ако е исполнето првото.

```cpp
#include <iostream>
using namespace std;

int main() {
    int poeni, prisustvo;
    cin >> poeni >> prisustvo;
    if (poeni >= 50) {
        if (prisustvo >= 75) {
            cout << "Ucenikot e polozen." << endl;
        }
    }
    return 0;
}
```

Во реални проблеми, честопати е потребно да се направи избор меѓу различни сценарија. Тогаш вгнездените искази може да се комбинираат и со else гранки, што овозможува различни однесувања во зависност од тоа кои услови се исполнети. Ова дополнително го зголемува потенцијалот на вгнездувањето, но бара внимателно организирање на кодот за да не дојде до логички грешки. Многу често, структурата на if-else може да се комбинира со повеќекратни вгнездени услови, со што се добива дрвовидна логика. Овие ситуации се типични при избор на цена во зависност од повеќе критериуми, како возрасна група и членство.

```cpp
#include <iostream>
using namespace std;

int main() {
    int godini;
    bool clen;
    cin >> godini >> clen;
    if (godini >= 18) {
        if (clen) {
            cout << "Cena e 100 denari." << endl;
        } else {
            cout << "Cena e 150 denari." << endl;
        }
    } else {
        cout << "Cena e 50 denari." << endl;
    }
    return 0;
}
```

Покрај придобивките, вгнездувањето носи и неколку предизвици. Прекумерната употреба може да го направи кодот тешко читлив и склон на грешки. Затоа, добрата практика препорачува да се користи кога е логички неопходно и каде што не може да се замени со поедноставена структура. Доколку се забележи дека кодот станува предолг или тешко следлив, може да се разгледа можноста делови од логиката да се издвојат во посебни функции. Тоа овозможува повторна употреба и подобро организирање на програмата. Исто така, треба да се внимава на прегледноста – користење на табулатори и коментари кои појаснуваат каде завршува кој услов.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што претставува вгнезден исказ?

2. Кога се употребува вгнезден if во реална ситуација?

3. Кои се ризиците од претерано вгнездување?

4. Како можеме да го направиме кодот попрегледен при вгнездување?

5. Напиши програма која ќе провери дали број е делив со 3, а потоа дали е парен!

6. Напиши програма која за ученик/чка внесува поени и присуство и одлучува дали положил/а!

7. Напиши програма која според возрасна група и членство пресметува цена на билет!

8. Напиши програма која проверува дали ученик/чка е со одличен успех и дали има награди!

Запомни што научи!

- Вгнездените искази се услови вметнати во други услови.
- Тие се користат кога логиката бара да се направат проверки само под одредени услови.
- Прекумерното вгнездување ги отежнува читливоста и одржувањето на кодот.
- Секогаш внимавај да биде јасно кој услов каде припаѓа.
- Вгнездувањето овозможува сложено одлучување базирано на повеќе поврзани критериуми.

3.24 Задачи за вежбање од вгнездена структура if else

Лесни задачи

1. Напиши програма што ќе провери дали внесениот број е позитивен и делив со 5.

2. Напиши програма што ќе провери дали бројот е делив со 10, и ако е, дали е поголем од 100.

3. Напиши програма што ќе провери дали бројот е непарен, и ако е, дали е едноцифрен.

4. Напиши програма што ќе провери дали број е делив со 4, а потоа дали е делив и со 6.

5. Напиши програма што ќе провери дали број е позитивен и делив со 3.

6. Напиши програма што ќе провери дали буквата е самогласка, и ако е, дали е голема буква.

Средно тешки задачи

7. Напиши програма што ќе провери дали температурата е под 0, и ако е, дали има дожд!

8. Напиши програма што ќе провери дали лицето е полнолетно, и ако е, дали има возачка дозвола.

9. Напиши програма што ќе провери дали годината е престапна, а потоа дали е парна.

10. Напиши програма што ќе провери дали името започнува со голема буква, и потоа дали има повеќе од 5 букви.

11. Напиши програма што ќе провери дали ученик/чка е запишан во средно училиште, и потоа дали има просек над 4.

12. Напиши програма што ќе провери дали внесениот број е двоцифрен, и ако е, дали последната цифра е парна.

13. Напиши програма што ќе провери дали денот е сабота, и ако е, дали температурата е над 25 степени.

Тешки задачи

14. Класниот раководител сака да провери кој од учениците заслужува посебна пофалница. За секој ученик/чка се знае колку поени освоил од тестови и колку домашни задачи има направено. Напиши програма што ќе провери дали ученик/чка има над 80 поени, а потоа дали ги има направени сите домашни задачи!

15. На крајот од полугодието, класниот сака да утврди кој од учениците можат да добијат позитивна забелешка за ангажираност. За секој ученик/чка се знаат освоените поени од предмет и бројот на неоправдани отсуства. Напиши програма што ќе провери дали ученик/чка има повеќе од 60 поени, и ако има дали има помалку од 5 неоправдани отсуства!

16. Во канцеларијата за запишување на возачки испити, за секое лице се проверува дали исполнува услови. За секое лице се знаат неговата возраст и тоа дали има извадена лична карта. Напиши програма што ќе провери дали лице има 18 или повеќе години, а потоа дали има лична карта!

3.25 Структура за избор од повеќе можности

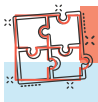
Веќе знаеш да...

- разликуваш меѓу услови со една и повеќе опции,
- користиш структури за избор од две можности (if-else),
- препознаеш логички изрази во програми,
- применуваш споредбени оператори,
- решаваш едноставни програмски проблеми со услови.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Како се поврзуваат if и else во повеќекратни слоеви?

2. Напиши програма што ќе проверува дали внесен број е позитивен и поголем од 100!



ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- разликуваш избор од повеќе можности од двојна условна структура,
- примениш if - else if - else структура,
- препознаеш кога е соодветна switch структурата,
- примениш повеќекратни услови во програма,
- напишеш програма со повеќе од две алтернативи,
- избегнуваш логички грешки при разгранување.

Кога се решаваат проблеми во секојдневниот живот или при пишување програма, често се соочуваме со ситуации кога треба да избереме помеѓу повеќе можности. На пример, ако температурата е под 0 °C, се облекува зимска јакна; ако е меѓу 0 и 20 °C – есенско палто, а ако е над 20 °C – тенка кошула. Тоа претставува избор меѓу повеќе можности. Во програмирањето, ваквото однесување се моделира со **структура за избор од повеќе можности**, која е логично продолжение на структурата за избор од две можности. За разлика од if-else, која овозможува само две патеки на извршување, структурата if - else if - else или switch нуди обработка на повеќе алтернативи. Ваквиот

тип логика е од клучно значење во програмирањето и во моделирањето на различни сценарија. Во оваа лекција ќе се обработи како и кога да се користат овие структури, како да се избегнуваат грешки при нивна употреба и како преку примери да се развие чувство за правилна примена на разгранети програмски текови.

Во реални програмски ситуации, често се јавува потреба од проверка на повеќе различни услови, каде што само еден од нив треба да биде исполнет. Во такви случаи се користи структурата if - else if - else. Таа овозможува редоследно проверување на условите, при што програмата влегува во првиот блок чиј услов е вистинит, а останатите се игнорираат. Важно е да се напомене дека оваа структура се користи кога условите се заемно исклучиви, односно кога само една гранка треба да се изврши. Овој пристап обезбедува јасен и уреден код, а воедно ги зголемува читливоста и одржливоста на програмата.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int broj;
6      cout << "Vnesi eden broj: ";
7      cin >> broj;
8
9      if (broj > 0)
10         cout << "Brojot e pozitiven." << endl;
11     else if (broj < 0)
12         cout << "Brojot e negativen." << endl;
13     else
14         cout << "Brojot e nula." << endl;
15
16     return 0;
17 }
18
```

Во примерот се користи структурата if - else if - else за да се провери дали бројот е позитивен, негативен или нула. Само еден од овие услови може да биде исполнет, а програмата ја избира соодветната гранка и ја игнорира останатата логика.

Иако if - else if - else структурата е моќна, понекогаш проверките се прават според вредности на една иста променлива. Во такви случаи, поедноставна и поефикасна алтернатива е switch структурата. Таа дозволува селекција на блок на код во зависност од вредноста на цел број или знак. Клучниот елемент тука е case, кој прет-

ставува една можност, додека default се користи како последна опција ако ниту еден case не одговара. Оваа структура ја подобрува читливоста, особено кога се работи со многу услови кои се проверуваат на истата променлива.

```
#include <iostream>
using namespace std;

int main() {
    int den;
    cout << "Vnesi broj od 1 do 7: ";
    cin >> den;

    switch (den) {
        case 1: cout << "Ponedelnik"; break;
        case 2: cout << "Vtornik"; break;
        case 3: cout << "Sreda"; break;
        case 4: cout << "Chetvrtok"; break;
        case 5: cout << "Petok"; break;
        case 6: cout << "Sabota"; break;
        case 7: cout << "Nedela"; break;
        default: cout << "Nema takov den!";
    }

    return 0;
}
```

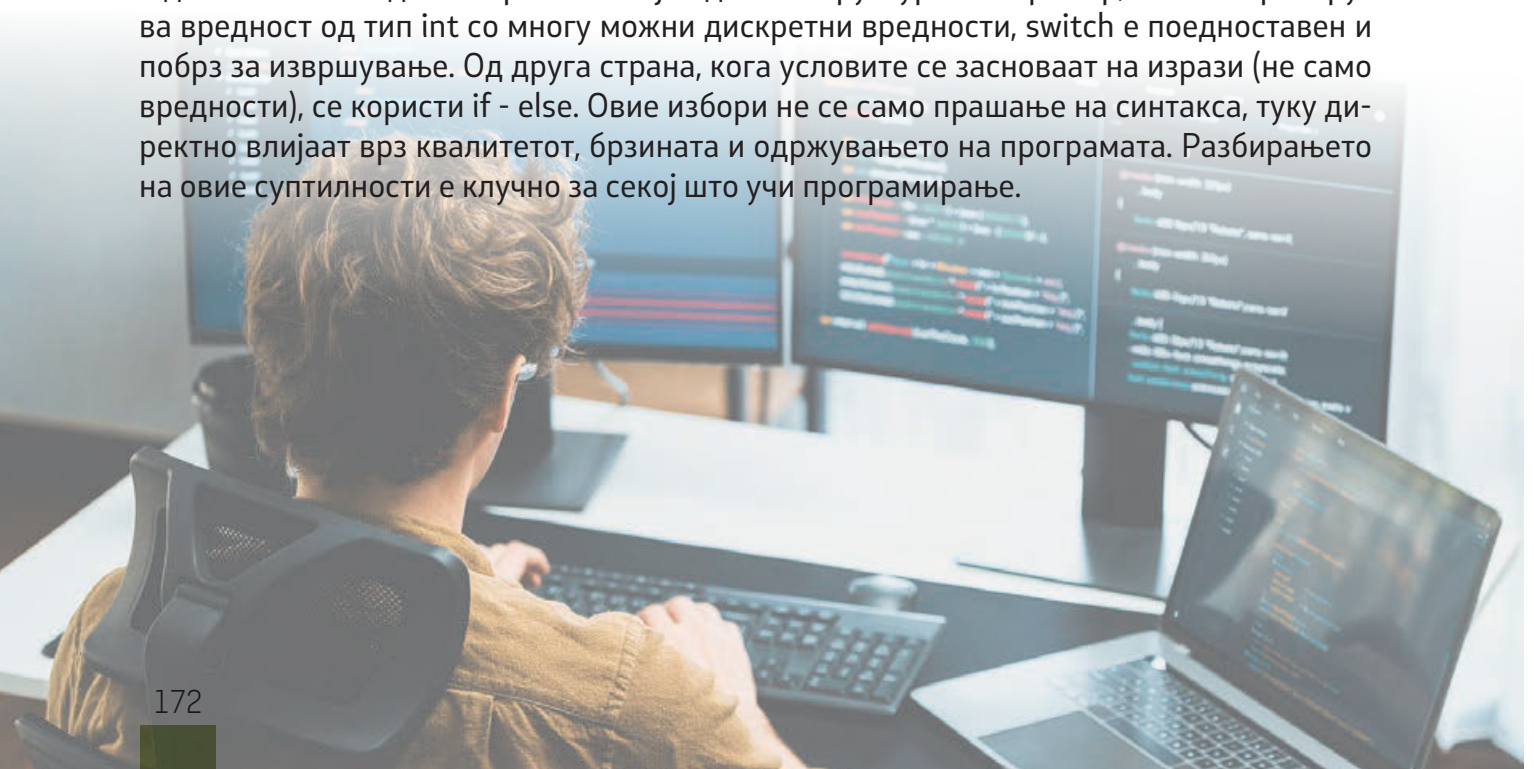
Овој пример користи switch за избор на ден во неделата. Ова е добар пример кога се врши избор според вредноста на една променлива и за секоја вредност се презема соодветна акција.

Во случаи кога се проверуваат многу услови, важно е да се внимава на редоследот на условите и да се избегне преклопување. Прво треба да се проверат најограничените (најстрогите) услови, а потоа поширокиte. Исто така, честа грешка е пропуштањето на else, поради што може да дојде до непредвидено извршување на повеќе блокови. Поради тоа, се препорачува користење на добро документиран и читлив код. Покрај тоа, променливите вклучени во условите треба да се изберат така што ќе обезбедуваат логичка конзистентност и лесно отстранување на грешките. Со внимателно распоредување на условите, се минимизираат ризиците од логички грешки и се добива појасен програмски тек.

```cpp
#include <iostream>
using namespace std;

int main() {
    int poeni;
    cout << "Vnesi broj na poeni (0-100): ";
    cin >> poeni;

    if (poeni >= 90)
        cout << "Ocena: 5" << endl;
    else if (poeni >= 75)
        cout << "Ocena: 4" << endl;
    else if (poeni >= 60)
        cout << "Ocena: 3" << endl;
    else if (poeni >= 50)
        cout << "Ocena: 2" << endl;
    else
        cout << "Ocena: 1" << endl;

    return 0;
}
```

Примерот покажува избор меѓу повеќе оценки врз основа на поени. Важно е да се забележи дека условите се распоредени по строгиот редослед: од највисоки спрема најниски. Така се овозможува точна категоризација.

Покрај правилното користење на if - else if - else и switch, од суштинско значење е да се знае кога да се користи секоја од овие структури. На пример, кога се проверува вредност од тип int со многу можни дискретни вредности, switch е поедноставен и побрз за извршување. Од друга страна, кога условите се засноваат на изрази (не само вредности), се користи if - else. Овие избори не се само прашање на синтакса, туку директно влијаат врз квалитетот, брзината и одржувањето на програмата. Разбирањето на овие суптилности е клучно за секој што учи програмирање.



ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што претставува структурата if - else if - else и кога се користи?

2. Објасни ја разликата меѓу if и switch структурите!

3. Зошто е важно условите да се распоредат од најстроги кон најопшти?

4. Кога се препорачува користење на switch наместо if?

5. Напиши програма што ќе прикаже која сезона е врз основа на внесен број на месец!

6. Напиши програма што ќе даде опис на ученик според бројот на поени (1 – 100)!

7. Напиши програма која ќе изврши проверка на оценка и ќе прикаже порака („Одлично“, „Добро“ итн.)!

8. Напиши програма што ќе прикаже вид на транспорт според внесен тип (1 – Автомобил, 2 – Авион, 3 – Воз)!

Запомни што научи:

- Структурата if - else if - else се користи кога треба да се избере само една од повеќе можности.
- Структурата switch овозможува избор според точна вредност на променлива.
- Условите секогаш треба да се поставуваат внимателно и логично.
- Само еден блок од кодот во if - else if - else структура се извршува.
- Во switch, секој case мора да завршува со break за да се спречи извршување на следниот.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Покрај основните структури, во напредното програмирање се користат и вгнездени switch изрази, како и комбинаторика на if услови со логички оператори (&&, ||). Во некои програмски јазици постојат и конструкции како match или pattern matching кои овозможуваат покомплексен избор на гранки. Исто така, во функционалното програмирање, изборите често се моделираат како податочни структури, што овозможува повисок степен на апстракција.

3.26 Задачи за вежбање од структура за избор од повеќе можности

Лесни задачи

1. Напиши програма што ќе провери дали внесениот број е позитивен, негативен или нула!

2. Напиши програма што ќе прикаже ден во неделата според внесен број од 1 до 7!

3. Напиши програма што ќе провери дали ученик има оценка 1, 2, 3, 4 или 5 и ќе испечати соодветен опис!

4. Напиши програма што ќе прикаже тип на облека според внесена температура (во °C)!

5. Напиши програма што ќе испечати име на месец според внесен број од 1 до 12!

6. Напиши програма што ќе испечати порака според внесен степен на образование (1 – основно, 2 – средно, 3 – високо)!

7. Напиши програма што ќе опише ниво на вода во резервоар според процент (низок, среден, висок)!

8. Напиши програма што ќе одреди временска зона според број на часови од GMT!

9. Напиши програма што ќе даде препорака за пијалак според возраста (под 14 – сок, до 18 – безалкохолно, над 18 – кафе)!

10. Напиши програма што ќе прикаже тип на сметка (1 – струја, 2 – вода, 3 – телефон)!

Средно тешки задачи

11. Напиши програма што ќе пресмета транспортна цена според растојание (под 10 км, до 50 км, над 50 км)!

12. Напиши програма што ќе класифицира музички жанр според внесен број (1 – рок, 2 – џез, 3 – поп, 4 – класика)!

13. Напиши програма што ќе одреди опис на квалитет на воздух според AQI вредност (0 – 50 – одличен, 51 – 100 – добар итн.)!

14. Напиши програма што ќе испечати тип на корисник според улога (админ, гостин, регистриран)!

15. Напиши програма што ќе одреди која боја има сигналното светло според внесен број (1 – црвено, 2 – жолто, 3 – зелено)!

16. Напиши програма што ќе пресмета данок според категорија на доход (ниска, средна, висока)!

17. Во училишната лабораторија се тестираат таблети за електронски учебници. За секој таблет се мери процентот на наполнетост на батеријата (од 0 до 100). Техничарот сака автоматски да се прикажува состојбата на батеријата:

од 0 до 20 % – „критично“,
од 21 до 60 % – „средно“,
од 61 до 100 % – „полна“.

Напиши програма што ќе прочита процент на батерија и ќе испечати соодветна порака за состојбата на батеријата според овие интервали!

3.27 Циклус while

Веќе знаеш да...

- разликуваш логички и споредбени изрази,
- користиш структури за избор во C++,
- разбираш што значи да се направи споредба меѓу две вредности,
- препознаваш кога е потребно да се повтори некоја постапка.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Кои се можните грешки при користење повеќекратни услови?

2. Напиши програма која според внесен број на ден од 1 до 7 ќе прикаже име на денот!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- разликуваш циклус од другите структури на програмирање,
- објасниш што претставува циклус и кога се користи,
- препознаеш различни типови циклуси во програмски јазик,
- разбираш зошто и кога се користи повторување на инструкции,
- напишеш едноставни програми со повторувачки структури,
- анализираш дали циклусот ќе се изврши еднаш, повеќепати или никогаш.

Во многу ситуации при решавање задачи, постои потреба од повторување на одредени активности повеќе пати. Дали станува збор за пресметување збир на елементи во низа, за прикажување мени или за верификација на лозинка, потребно е некои инструкции да се извршат повеќе пати. Во програмирањето, ова се реализира преку такаНаречени **циклуси**, кои овозможуваат делови од програмата да се извршуваат повторно, сè додека е исполнет одреден услов. Циклусите претставуваат една од основните и најважни структури за контрола на текот во програмите. Разбирањето на нивната логика и правилно користење е суштински чекор во совладување на програмирањето. Тие овозможуваат програми што се пофлексибилни, пократки и поефикасни.

Во програмирањето, структурата наречена **циклус** претставува начин на извршување на ист блок од код повеќепати, сè додека одреден услов е вистинит. Притоа, за разлика од секвенцијалното извршување каде што секоја наредба се извршува еднаш и во даден редослед, циклусот овозможува дел од програмата да се „врати назад" и да го изврши истиот код повторно. Ова повторување може да биде однапред дефинирано (на пример, 10 пати) или условно – сè додека не се постигне некоја состојба. Постојат неколку вида циклуси, од кои секој има своја намена и карактеристики, но сите имаат заедничка идеја: **повторување на инструкциите** за да се избегне нивно рачно повторување повеќе пати во кодот. Структурата на while циклусот изгледа вака:

```
while (услов) {
   // команди што се извршуваат
}
```

ПРИМЕР 1

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       int i = 0;
6       while (i < 5) {
7           cout << "Zdravo!" << endl;
8           i++;
9       }
10      return 0;
11  }
12
```

Претпостави дека треба да се испечати „Здраво!" пет пати. Без циклус, би требало да напишеш пет cout наредби. Но, со циклус, ова може да се направи многу поелегантно и поефикасно.

Во овој пример, променливата i се зголемува со секое извршување на циклусот и штом ќе достигне 5, условот i < 5 станува невистинит и циклусот престанува.

Една од најважните предности на користењето циклуси во програмирањето е **автоматизацијата** на повторувачки задачи. Наместо програмерот рачно да копира и повторува код, циклусите овозможуваат истиот блок код да се користи повеќепати со минимални измени. Ова не само што ја прави програмата пократка и полесна за одржување туку и ја намалува можноста за грешки. Притоа, секоја итерација на циклусот обично се контролира преку променлива која се нарекува **бројач**, што овозможува дополнителна логика при секое повторување. Бројачот може да се користи за индексирање на елементи, за собирање вредности, за проверка на услови и многу други функции. Кога е добро испланирано, користењето циклус значително ги подобрува **структурата и читливоста** на кодот.

ПРИМЕР 2

Сакаме да извршиме пресметка на збирот на првите 10 природни броеви (од 1 до 10).

```
#include <iostream>
using namespace std;

int main() {
    int zbir = 0;
    for (int i = 1; i <= 10; i++) {
        zbir += i;
    }
    cout << "Zbirot na prvite 10 broevi e: " << zbir << endl;
    return 0;
}
```

Тука, for циклусот автоматски иницијализира, проверува услов и го ажурира бројачот i. На секоја итерација, i се додава на zbir. По завршување, zbir ја содржи сумата.

Циклусите не само што овозможуваат повторување, туку и создаваат услови за **динамично однесување на програмите**. Во зависност од внесот на корисникот, резултатот од некоја пресметка или друго надворешно влијание, циклусот може да продолжи да се извршува или да се прекине. Таквите циклуси се нарекуваат **условни циклуси**, и најчесто се користи while или do-while структурата. Во while циклусот, условот се проверува пред секое извршување, додека do-while најмалку еднаш го извршува телото на циклусот пред да ја направи проверката. Вакви циклуси често се користат кога не е познато однапред колку итерации ќе бидат потребни – на пример, додека корисникот не внесе валидна вредност или додека не се исполнат некои динамички критериуми. Тоа му дава на програмскиот јазик флексибилност да реагира на различни сценарија без повторување на ист код.

Програмерите често комбинираат циклуси со услови и операции над податоци за да изградат сложени алгоритми, од симулации до анализа на податоци. Вештината за избирање соодветен тип циклус, поставување точен услов, и избегнување **бесконечни циклуси**, претставува еден од темелите на доброто програмирање. Понекогаш, само мала грешка во условот или ажурирањето на бројачот може да резултира со програма што никогаш не завршува – затоа циклусите се алатка што бара дисциплина и внимание.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што претставува циклус во програмирањето?

2. Кои типови циклуси ги познаваш и по што се разликуваат?

3. Зошто се користат бројачи во циклуси?

4. Што ќе се случи ако условот во while циклус никогаш не се исполни?

5. Напиши програма која ќе ги испечати сите парни броеви од 1 до 50!

6. Напиши програма која ќе ја пресмета сумата на сите броеви од 1 до N, каде што N се внесува од тастатура!

7. Напиши програма која ќе бара од корисникот да внесе позитивен број сè додека не го направи тоа!

8. Напиши програма која ќе пресмета производ на сите броеви од 1 до 10!

Запомни што научи!

- Циклусите овозможуваат повторување на код.
- Се користат три основни типови: for, while и do-while.
- Циклусите се ефикасен начин за автоматизација.
- Бројачите и условите го контролираат бројот на повторувања.
- Погрешно поставен циклус може да доведе до бесконечно извршување.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во натпреварувачкото програмирање и задачите од типот на „Дабар“, циклусите се употребуваат за решавање проблеми во кои се бара анализа или процесирање на големи серии податоци. На пример, броење колку пати се повторува некоја буква во даден текст, или симулација на движење низ лавиринт. Преку комбинација на циклуси и услови се решаваат многу алгоритамски задачи, вклучувајќи сортирање, пребарување и анализа на матрици. Колку што порано ќе ја совлада ученикот логиката на циклусите, толку побрзо ќе може да пристапува и кон понапредни концепти како рекурзија, структури од податоци и оптимизација.

3.28 Задачи за вежбање од структура за повторување while

Лесни задачи

1. Напиши програма што ќе ги испечати броевите од 1 до 15!
2. Напиши програма што ќе ги испечати броевите од 10 до 1 во обратен редослед!
3. Напиши програма што ќе ги испечати сите парни броеви од 20 до 40!
4. Напиши програма што ќе испечати секој трет број почнувајќи од 5 до 30!
5. Напиши програма што ќе ги испечати сите броеви од 1 до 100 што се деливи со 10!
6. Напиши програма што ќе ја испечати фразата „Вежбам while циклуси“ точно 6 пати!
7. Напиши програма што ќе го прикаже збирот на броеви од 30 до 40!

Средно тешки задачи

8. Напиши програма што ќе ги пресмета броевите од 1 до N (внесен од тастатура) што се деливи со 3!
9. Напиши програма што ќе бара од корисникот да внесе 10 цели броеви и ќе го прикаже најголемиот меѓу нив!
10. Напиши програма што ќе изброи колку броеви од 1 до 50 се деливи со 7!
11. Напиши програма што ќе го пресмета збирот на сите непарни броеви од 1 до 100!
12. Напиши програма што ќе изброи колку цифри има внесениот број!
13. Напиши програма што ќе го испечати збирот на сите броеви меѓу два внесени броја A и B!
14. Напиши програма што ќе провери дали внесениот број е делив и со 2 и со 5!

Тешки задачи

15. Во клубот по математика учениците учат за „совршени броеви“ – броеви кај кои збирот на сите нивни позитивни делители (без самиот број) е еднаков на самиот број. На пример, за 6 важи: 1 + 2 + 3 = 6. Напиши програма што ќе прочита еден цел број и ќе провери дали е тој совршен!
16. Напиши програма што ќе ја најде најмалата цифра во внесениот број!
17. Напиши програма што ќе ја пресмета сумата на квадратите на броевите од 1 до N!
18. Напиши програма што ќе брои колку броеви се деливи со 3 и 4 од 1 до 100!
19. Напиши програма што ќе ги изброи сите палиндромски броеви од 10 до 999!
20. Во проверка на безбедносен код, дозволени се само броеви составени исклучиво од парни цифри (0, 2, 4, 6, 8). Ако во бројот се појави барем една непарна цифра, кодот е невалиден!

3.29 Циклус со услов – do while структура

Веќе знаеш да...

- разликуваш основни типови циклуси во програмирањето,
- препознаваш ситуации кога треба да се користи повторување на инструкции,
- го разбереш значењето на условите во контрола на текот на програмата,
- прочиташ и да разбереш едноставен циклус напишан во C++.

ЗАДАЧИ И ПРАШАЊА ЗА ПОВТОРУВАЊЕ:

1. Дали секој циклус може да се напише со while?

2. Напиши програма која ќе ги испечати сите броеви од 100 до 1 во обратен редослед!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- објаснуваш како функционира do while циклусот,
- препознаваш ситуации во кои е соодветно да се користи do while,
- напишеш програма што користи do while циклус за повторување додека не се исполни услов,
- избегнуваш бесконечни циклуси со правилна формулација на услов,
- вклучиш променливи, услови и ажурирање на вредности во циклусот.

Во програмирањето, често се среќаваат ситуации кога некој блок од код мора да се изврши **барем еднаш**, без разлика на тоа дали условот е вистинит. Во вакви случаи, се применува do while циклусот, кој нуди уникатна структура – прво се извршува телото на циклусот, а потоа се проверува условот. Ако условот остане вистинит, циклусот се повторува. Овој пристап е корисен кога програмата бара **иницијално извршување**,

како што е внесување на број од корисникот, избор на опција од мени, или активирање на некоја акција што мора да се случи барем еднаш. Структурата на do while циклусот изгледа вака:

```
do {
   // команди што се извршуваат
} while (услов);
```

ПРИМЕР 1

Следниот пример покажува како се користи do while циклус за да се осигури внесување позитивен број од страна на корисникот. Циклусот ќе се повторува сè додека не биде внесена валидна вредност:

```
#include <iostream>
using namespace std;

int main() {
    int broj;
    do {
        cout << "Vnesi pozitiven broj: ";
        cin >> broj;
    } while (broj <= 0);

    cout << "Validen broj e: " << broj << endl;
    return 0;
}
```

Во овој случај, корисникот има можност да го внесе бројот најмалку еднаш, а ако внесот не е валиден (не е позитивен број), програмата ќе го бара повторно сè додека не биде внесена точна вредност. Овој пристап се користи и за безбедна интеракција со корисникот, кога е потребна проверка на валидноста.

do while циклусот најчесто се применува кога се работи со **менаџирање на интеракција со корисникот**, на пример во менија, кога корисникот треба да избере опција, а потоа да му се даде можност да избере повторно додека не избере „излез“. За разлика од while циклусот, кој може да не се изврши воопшто ако условот не е исполнет од самиот почеток, do while е гаранција дека логиката во телото на циклусот ќе се изврши најмалку еднаш. Таа предност станува суштинска кога првиот повик кон корисникот или кон некоја функција **мора** да се изврши, без разлика на логичката состојба на условот. На тој начин се обезбедува предвидливо и контролирано однесување на програмата.

ПРИМЕР 2

Во следниот пример е прикажано мени со три опции. Корисникот може да избира додека не внесе опција 3 – „излез“:

```cpp
#include <iostream>
using namespace std;

int main() {
    int izbor;
    do {
        cout << "=== Meni ===" << endl;
        cout << "1. Zemi podatoci" << endl;
        cout << "2. Prikazi podatoci" << endl;
        cout << "3. Izlez" << endl;
        cout << "Tvoj izbor: ";
        cin >> izbor;

        if (izbor == 1)
            cout << "Se zemaat podatoci..." << endl;
        else if (izbor == 2)
            cout << "Se prikazhuvaat podatoci..." << endl;
        else if (izbor != 3)
            cout << "Nepoznata opcija!" << endl;

    } while (izbor != 3);

    cout << "Programata zavrshi." << endl;
    return 0;
}
```

Овој пример демонстрира типична примена на do while кај интерактивни програми. Прво се прикажува мени, потоа се чека внес од корисникот, а процесот се повторува сè додека не се избере излезната опција. Ова е класичен начин за реализација на **интерактивни конзолни апликации** со јасна и безбедна логика.

Главната карактеристика што го издвојува do while е **редоследот на извршување:** телото на циклусот се извршува **пред** да се провери условот, додека кај другите циклуси, условот се проверува **пред** секоја итерација. Тоа значи дека do while секогаш се извршува барем еднаш, без оглед на тоа дали условот е вистинит од самиот почеток. Кај while и for, може да се случи циклусот воопшто да не се изврши ако условот не е исполнет од самиот старт.

Дополнително, треба да се внимава на можноста за **бесконечен циклус**, ако условот никогаш не стане лажен. Затоа се препорачува во телото на do while да постои **јасна логика која води до прекин**, како што беше прикажано во претходните примери со внес, избор или случајна генерација. Од аспект на читливост и одржување на кодот, do while треба да се користи само кога има вистинска потреба за иницијално извршување.

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Кој е главниот структурен принцип по кој се разликува do while циклусот од while циклусот?

2. Кога е најпогодно да се употреби do while наместо while или for?

3. Дали е можно do while циклусот никогаш да не се изврши? Објасни зошто!

4. Зошто do while се користи во интерактивни менија?

5. Напиши програма што ќе бара од корисникот да внесе број поголем од 100. Ако внесе помал број, повторно ќе го бара, користејќи do while!

6. Напиши програма што ќе симулира фрлање две коцки додека не се добијат двојки (исти броеви)!

7. Напиши програма што ќе го праша корисникот дали сака да продолжи со програмата („да“ или „не“), сè додека не внесе „не“!

8. Напиши програма што ќе ги собира броевите што ги внесува корисникот додека не внесе нула, и ќе го прикаже збирот!

Запомни што научи!

- do while циклусот се користи кога сакаме **најмалку еднаш** да се изврши блок од инструкции.
- Условот за продолжување се проверува **по извршувањето** на циклусот.
- Се применува при **внес на податоци, интерактивни менија, случајни симулации**, и **иницијални проверки**.
- Секогаш мора да има јасна логика која ќе доведе до **прекин** на циклусот за да не се создаде бесконечно извршување.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Во некои програмски јазици како Python, do while циклусот не постои како вградена конструкција. Во такви случаи, неговата логика се реализира со while True: и вграден услов за прекин со break. Дополнително, напредните симулации, како симулација на хазардни игри или менаџери за дијалог во видеоигри, исто така ја користат логиката на do while. Во вакви системи, програмерите често го комбинираат do while со графички интерфејси за да овозможат повторлива интеракција со корисникот.

3.30 Задачи за вежбање од структура за повторување do while

Лесни задачи

1. Напиши програма што ќе ги испечати сите непарни броеви од 1 до 30!

2. Напиши програма што ќе ги испечати броевите од 10 до 1 во надолен редослед!

3. Напиши програма што ќе го пресмета збирот на сите парни броеви од 1 до 100!

4. Напиши програма што ќе бара од корисникот да внесе број сè додека не внесе 0!

5. Напиши програма што ќе прикажува порака „Повторно внеси!“ сè додека не се внесе број поголем од 100!

6. Напиши програма што ќе ги испечати сите броеви деливи со 5 меѓу 1 и 50!

7. Напиши програма што ќе го пресмета збирот на сите броеви што се делат со 3 меѓу 1 и 50!

8. Напиши програма што ќе го пресмета збирот на цифрите на еден внесен позитивен број!

Средно тешки задачи

9. Напиши програма што ќе одредува дали внесен број е прост!

10. Напиши програма што ќе го прикажува обратниот редослед на цифрите од внесен број!

11. Напиши програма што ќе го пресмета производот на сите непарни броеви меѓу 1 и 15!

12. Напиши програма што ќе го најде најголемиот број од низата броеви што корисникот ги внесува, додека не внесе -1!

13. Напиши програма што ќе брои колку цифри има внесен број!

14. Напиши програма што ќе бара внес на броеви и ќе ја пресмета нивната средна вредност додека не се внесе негативен број!

Тешки задачи

15. Во една нумеричка загатка се бара „таен број“ скриен помеѓу 100 и 1 000. Знаеме две работи за него: бројот мора да биде делив со 17, збирот на неговите цифри мора да биде 15.

Напиши програма што ќе го пронајде и ќе го испечати првиот број помеѓу 100 и 1 000 (по растечки редослед) кој е делив со 17 и има збир на цифрите еднаков на 15!

16. Напиши програма што ќе го пресмета факториелот на внесен број!

17. Напиши програма што ќе го пресмета збирот на квадратите на првите N природни броеви, каде што N го внесува корисникот!

18. Учениците внесуваат низа од цели броеви во компјутер за да ја анализираат. Внесувањето на броевите завршува со 0, а нулата служи само како знак за крај и не се смета како дел од низата. Потребно е да се најде вториот најголем број од сите внесени (различни) броеви. Напиши програма што ќе внесува целобројна низа (еден по еден број), додека не се внесе 0 како крај, и потоа ќе го одреди и испечати вториот најголем број од внесените броеви!

3.31 Циклус со for структура

Веќе знаеш да...

- разликуваш основни контролни структури како услов и циклус,
- препознаеш кога програмата треба да повтори одредена акција,
- користиш do while за најмалку едно извршување,
- следиш логика на редоследно извршување во програма.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Како можеш да избегнеш бесконечен do while циклус?

2. Напиши програма што ќе бара лозинка од корисникот додека не внесе точно определена лозинка („tajna123“)!

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- користиш for циклус за броење со почетна и крајна вредност,
- објасниш како функционираат иницијализација, услов и ажурирање во for,
- пишуваш програми со фиксни бројки на повторување,
- споредуваш for, while и do while циклуси,
- решаваш реални задачи со повторување на активности.

Во програмирањето, потребата од повторување на одредени инструкции е секојдневна. Овие повторувања најчесто се појавуваат во ситуации кога се знае **точно колку пати** треба да се изврши некоја акција. Тогаш најсоодветна е контролна структура која овозможува броење на повторувања. Таквата структура во C++ (и во многу други програмски јазици) се нарекува **for циклус**. Оваа структура е компактна, лесна за читање и јасно ја прикажува логиката на почетна вредност, услов за прекин и чекор (ажурирање). Токму поради овие карактеристики, for циклусот често се користи при **обработка на низи**, **табели**, **внесување податоци** и **изведување пресметки во точно**

определен број чекори. Во оваа лекција ќе се обработи неговата форма, начинот на извршување, како и конкретни примери од реалниот живот и програмски ситуации.

For циклусот во C++ е конструиран така што ги содржи трите основни делови: **иницијализација**, **услов за продолжување** и **ажурирање на вредноста**, сите вградени во заградите на циклусот. Структурата изгледа вака:

```
for (иницијализација; услов; ажурирање) {
   // инструкции што се повторуваат
}
```

Во делот „иницијализација" се внесуваат почетни вредности за променливите. Доколку имаме потреба од повеќе променливи да се иницијализираат, тие се одделуваат со запирка. Исто така и во делот ажурирање каде што најчесто се стават итератори, ако имаме потреба од повеќе ажурирања, тие се одделуваат со запирка. Оваа форма овозможува сите клучни делови од циклусот да бидат концентрирани на едно место, што ја зголемува читливоста на кодот. Иницијализацијата обично поставува почетна вредност на бројачот (на пример int i = 1;), условот одредува до кога ќе се повторува циклусот (на пример i <= 5;), а ажурирањето ги менува вредностите при секоја итерација (на пример i++).

ПРИМЕР 1

Печатење броеви од 1 до 5

Во овој пример, циклусот започнува со i = 1, и се повторува сè додека i е помало или еднакво на 5. По секоја итерација, i се зголемува за 1. На овој начин, на екранот се прикажуваат броевите од 1 до 5.

Во многу ситуации, програмите бараат бројач што се движи во обратен правец – од повисока кон пониска вредност. Ова може да биде потребно при **одбројување, приказ на елементи во обратен редослед, или барање до исполнување на некој услов при крајот на низа**. For циклусот со обратен бројач се дефинира со појдовна вредност што е поголема, услов кој завршува кога бројачот ќе падне под некоја вредност и ажурирање со намалување на бројачот. Овој вид имплементација е интуитивна и ја покажува флексибилноста на for циклусот, кој може да се прилагоди според логиката на проблемот.

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       for (int i = 1; i <= 5; i++)
6       {
7           cout << i << endl;
8       }
9       return 0;
10  }
11
```

ПРИМЕР 2

Одбројување од 10 до 1

Овде циклусот почнува со i = 10 и се намалува со i-- додека i е поголемо или еднакво на 1. На екранот ќе се испечатат броевите 10 9 8 ... 1. Ваквото одбројување се користи, на пример, во тајмери, аларми и интерактивни игри кога треба да се прикаже колку време останува.

Дополнително, for циклусот може да се користи за **собирање, множење или пресметување збирни вредности**. Ова е особено корисно кога се работи со **внесени податоци од корисникот или изведување пресметки со одреден број итерации**. Наместо само печатење вредности, циклусот може да го обработува бројачот и да ја складира резултатската вредност во променлива. Тоа е чест пристап при имплементација на задачи што вклучуваат пресметка на сума, просек, факториел, па дури и наоѓање максимум или минимум.

```
#include <iostream>
using namespace std;

int main() {
    for (int i = 10; i >= 1; i--)
    {
        cout << i << " ";
    }
    return 0;
}
```

ПРИМЕР 3

Пресметка на збир на првите 100 природни броеви

Во овој пример, збирот започнува со 0 и на секоја итерација му се додава тековната вредност на i. По завршување на циклусот, sum ќе ја содржи конечната вредност: 5050. Таков тип задачи се среќаваат и во почетно програмирање и во математички модели.

Во реални програмски решенија, често се користи for циклус за **пристап до елементи во низа или серија на податоци**. Овој пристап овозможува прецизно и контролирано движење низ секој елемент на низа со точно дефиниран број повторувања. Структурата на for циклусот овде е идеална, бидејќи ја користи големината на низата за да постави услов за завршување на циклусот и автоматски го ажурира индексот со секоја итерација. Ваквата употреба на циклусот е фундаментална при обработка на податоци, статистика, автоматизација на пресметки и многу други алгоритамски задачи.

```
#include <iostream>
using namespace std;

int main() {
    int sum = 0;
    for (int i = 1; i <= 100; i++)
    {
        sum += i;
    }
    cout << "Zbirot e: " << sum << en
    return 0;
}
```

ПРАШАЊА И ЗАДАЧИ ЗА ПОВТОРУВАЊЕ НА ЗНАЕЊЕТО

1. Што е for циклус и кога се користи?

2. Која е разликата меѓу for и while циклус во однос на синтаксата и примената?

3. Како изгледа структурата на for циклусот и што претставуваат неговите три делови?

4. Што ќе се случи ако во for циклусот се пропушти делот за ажурирање?

5. Објасни што ќе испечати следната програма:

```
for (int i = 1; i <= 5; i++) {
cout << i * i << " ";
}
```

6. Напиши програма што ќе ги испечати сите парни броеви од 2 до 20 користејќи for циклус!

7. Напиши програма што ќе го пресмета производот на првите 10 природни броеви!

8. Напиши програма што ќе го испечати збирот на сите непарни броеви меѓу 1 и 99!

9. Напиши програма што ќе ја прикаже абецедата од A до Z користејќи for циклус!

10. Напиши програма што ќе одбројува од 50 до 1 и ќе испечати „ГОТОВО!“ на крај!

Запомни што научи!

- For циклусот е структура за повторување со познат број итерации.
- Се состои од иницијализација, услов и ажурирање.
- Идеален е за бројачки задачи, како броење, пресметки и обработка на низи.
- Може да се користи во растечки или опаѓачки редослед.
- Се применува во многу реални задачи во програмирање и анализа на податоци.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

- Во C++ постојат и вложени for циклуси во кои еден циклус е вграден во друг. Тие се користат за обработка на матрици и табели.
- For циклусот може да биде комбиниран со условни искази, како if, за да се извршат специфични пресметки.
- При напредно програмирање, се користи итерација со итератори во структури како вектори (vector), мапи (map) и множества (set).
- Можете да користите и опсег-базирани for циклуси (range-based for) во C++11 и понови верзии.

3.32 Задачи за вежбање од структура за повторување for

Лесни задачи

1. Напиши програма што ќе ги испечати сите броеви од 1 до 20!
2. Напиши програма што ќе ги испечати сите непарни броеви од 1 до 50!
3. Напиши програма што ќе го испечати збирот на броевите од 1 до 100!
4. Напиши програма што ќе ги испечати броевите од 10 до 1 во обратен редослед!
5. Напиши програма што ќе го испечати производот на броевите од 1 до 5!
6. Напиши програма што ќе ги испечати сите броеви деливи со 3 од 1 до 30!
7. Напиши програма што ќе го испечати квадратот на сите броеви од 1 до 10!

Средно тешки задачи

8. Напиши програма што ќе го испечати збирот на сите парни броеви од 1 до 100!
9. Напиши програма што ќе ги испечати сите броеви од 1 до N (внесено од корисникот) што се деливи со 5!
10. Напиши програма што ќе го испечати збирот на сите броеви од 1 до N што не се деливи со 2!
11. Напиши програма што ќе испечати за секој број од 1 до 10, негов квадрат и куб!
12. Напиши програма што ќе изброи колку броеви од 1 до 100 се деливи со 7!
13. Напиши програма што ќе пресмета факториел на број N (внесен од корисникот)!
14. Напиши програма што ќе испечати аритметичка низа со почеток 3, разлика 5 и 10 елементи!

Тешки задачи

15. Напиши програма што ќе испечати колку пати во броевите од 1 до 100 се појавува цифрата 3!
16. Напиши програма што ќе го испечати најмалиот и најголемиот број од N броеви внесени од тастатура!
17. Напиши програма што ќе ја пресмета средната вредност на N внесени броеви!
18. Напиши програма што ќе ги испечати сите броеви од 100 до 999 кај кои збирот на цифрите е еднаков на 15!
19. Напиши програма што ќе ги испечати сите броеви од 1 до 100 кај кои цифрата на десетки е 2!
20. Напиши програма што ќе испечати броеви од 1 до 100, при што наместо броевите деливи со 3 ќе печати „Fizz“, со 5 „Buzz“, а со 3 и 5 „FizzBuzz“!

ЗАКЛУЧОК

Во темата „Програмирање во C++“ се заокружува една целина во која на учениците им се поставува темелот за понатамошно изучување на информатиката и алгоритамското размислување. На почеток се разгледуваат логички и алгоритамски задачи, со што се развива навика проблемите да се анализираат чекор по чекор и да се претвораат во јасни упатства. Потоа се воведуваат основните информатички концепти и бинарните броеви, со што се разбира како се претставуваат и се обработуваат податоците во компјутерот, а со воведот во кодирање и енкрипција се посочува и важноста на безбедното ракување со информациите.

Понатаму, вниманието се насочува кон поимот програмирање, програмските јазици и преведувачи, како и разликите меѓу различни јазици, каде што C++ се поставува како главна алатка за учење, но истовремено се согледува и местото на други јазици како Scratch, Java, Python, PHP и Lisp. Со интегрираната развојна околина се прикажува како се претвораат теоретските концепти во практична работа: се пишуваат, се извршуваат и се дебагираат програми. Преку псевдокод и секвенца од искази се учи како решението најнапред се опишува на разбирлив „човечки“ јазик, а потоа се претвора во конкретни C++ наредби.

Во следниот сегмент се воведуваат променливите и константите, нивните типови и правилата за доделување и приказ на вредности, со што се создава претстава како се чуваат податоците во меморијата. Со аритметичките операции и изрази, внесувањето податоци, споредбените и логичките операции, се поставуваат математичката и логичката основа на програмата. На таа подлога се гради контролата на текот: структурата за избор од две и повеќе можности, вгнездените услови и трите основни видови циклуси (while, do while и for), со што се овозможува да се решаваат и посложени задачи со повторување и разгранување на извршувањето.

Со бројните задачи за вежбање по секоја целина се зацврстува усвоеното знаење, се поттикнува самостојно размислување и се гради сигурност во користењето на јазикот C++. На крајот на темата ученикот се доведува во состојба да ја разбере логиката на една програма, да осмисли алгоритам, да го запише во псевдокод и да го претвори во исправен C++ код кој чита, обработува и прикажува податоци. На тој начин се поставува цврста основа врз која, во следните теми и години, ќе можат да се надградуваат понапредни концепти и техники на програмирање.

ПОВТОРУВАЊЕ ЗА ТЕМА 3

1. Објасни со свои зборови што е алгоритам и како се разликува од обична описна постапка во секојдневниот живот (на пример, рецепт за јадење)!

2. Осмисли секојдневна ситуација (патување до училиште, подготовка за тест, пазарење) и напиши ја како јасна низа од чекори! Подвлечи кои чекори се влез, кои се обработка, а кои се излез!

3. Состави логичка задача со три услови (на пр., за распоред на луѓе, предмети или бои) и напиши барем две можни решенија!

4. Претвори го децималниот број 37 во бинарен систем! Опиши ја постапката чекор по чекор!

5. Даден е бинарниот број 101101_2. Пресметај ја неговата децимална вредност и објасни како е добиен резултатот!

6. Наведи два примера од секојдневниот живот во кои се користи кодирање на информации (надвор од компјутери) и објасни што се кодира во секој пример!

7. Објасни што е енкрипција (шифрирање) и дај еден пример во кој би било важно пораката да биде енкриптирана!

8. Објасни со свои зборови што значи дека една програма е детерминистичка и зошто е важно тоа при решавање задачи!

9. Напиши три сличности и три разлики меѓу програмски јазик и природен јазик (како македонски или англиски)!

10. Објасни ја разликата меѓу компајлер и интерпретер! Наведи пример кога би ти било важно да знаеш кој тип се користи!

11. Опиши ја улогата на интегрирана развојна околина (IDE) при пишување програма! Наведи најмалку три функции што му ја олеснуваат работа на програмерот!

12. Напиши алгоритам во псевдојазик што чита три цели броја и го прикажува нивниот аритметички просек!

13. Напиши алгоритам во псевдојазик што за внесени должина и ширина на правоаголник ги пресметува и ги прикажува неговата плоштина и периметарот!

14. За секоја од променливите: возраст на ученик, температура на воздух, цена на билет и име на град – одреди соодветен тип на податок (цел, децимален, текстуален) и објасни зошто!

15. Запиши во псевдојазик израз што ја пресметува вредноста на $3 \cdot x^2 - 2 \cdot x + 5$ за дадено x! Објасни како би се пресметувал овој израз чекор по чекор!

16. Осмисли еден пример за споредбен израз и еден пример за логички израз! Објасни во кои ситуации нивната вредност ќе биде „вистина“, а во кои „невистина“!

17. Напиши услов (if–else структура) во псевдојазик што проверува дали ученик има положено предмет ако прагот за положување е 50 поени! Во зависност од резултатот прикажи соодветна порака!

18. Објасни ја разликата меѓу циклусите while и do while! Наведи по еден пример (описно) кога е попогодно да се користи првиот, а кога вториот!

19. Напиши алгоритам со for циклус кој ги прикажува сите парни броеви од 1 до 50 на екран, во еден ред!

20. Осмисли проблем од секојдневниот живот кој бара и услови и циклуси (на пример, пресметка на вкупната цена со попуст при купување повеќе производи)! Опиши ја логиката на решението како низа од чекори што би можеле да се претворат во програма!


BOLD
ALPHABET
A
A
CONCE
esc
tab
caps lock
shift
fn
control
option
command

ТЕМА 4: Работа со текст

Денес компјутерите секојдневно се користат за изработка на текстуални документи и нивно уредување. Најчесто користени уредувачи за работа со текст се MS Word и Writer од пакетот LibreOffice, како и онлајн верзијата на Word и Google Docs. Едноставни се за ракување и нудат многу можности при уредување текст во различен формат и со различен стил. Во овој учебник ќе бидат објаснети функционалностите на првите два уредувачи кои во голема мера се слични со онлајн уредувачите.

Нови поими

- Наслов, поднаслов, содржина, индекс, волшебник за образец, форма, поле за контрола на текст, копче за избор, паѓачко мени.

Веќе знаеш да...

- вршиш промена на фонт, боја и стил на букви;
- определиш проред на пасус и растојание меѓу повеќе пасуси;
- пишуваш текст во колони;
- додаваш слики, фотографии, табели, графици и други графички елементи.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

1. Направи плакат за концерт на твојата омилена група според следниве упатства:

- Местото и времето на одржување да се напишани со кирилични букви и големина 20 точки;
- Да бидат поставени централно и во горниот дел на плакатот;
- Името на групата и нејзините членови да се напишани со латинични букви. Местоположбата определи ја по сопствен избор така што ќе привлече повеќе посетители;
- Постави соодветна рамка на името која значително ќе го истакне;
- Името на организаторот да е напишано со кирилични букви и со сина боја;
- Додај графички елементи по твој избор.

2. Напиши кратко сценарио од десетина реда за видеоигра! Текстот да е со големина 12 точки и проред 1,15 линија. Насловот уреди го со букви со големина 14 точки и да е поставен централно. Текстот подели го на два пасуса со меѓусебно растојание од 6 точки. Маргините на страницата да имаат вредност 2 см. Во заглавието напиши го името на видеоиграта, а во подножјето напиши го твоето име и презиме.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- креираш и уредуваш текстуален документ со различни стилови;
- уредуваш содржина во документи според одредени барања;
- креираш индекси во документ;
- користиш шаблони и формулари.
- применуваш заштита на документ.

4.1 Работа со стилови

Колку начини на уредување текст знаеш?
Со кои наредби е уреден текстот подолу?

„Во денешниот дигитален свет, способноста за јасна и професионална комуникација е **моќна** вештина – да **започнеме** со изучување на правилата со кои вашите идеи стануваат влијателни ПОРАКИ."

Уредување текстови со стилови во Microsoft Word

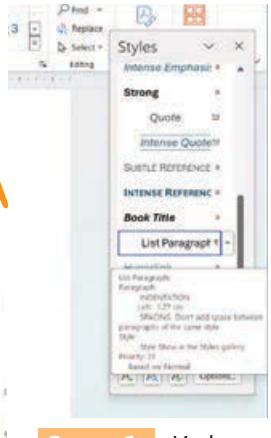
4.1.1 Разбирање на стиловите во текстуален документ

Во програмите за работа со текст, **стилот** претставува збир од уредувања што се применуваат врз текст, вклучувајќи вид и големина на фонт, боја, порамнување на пасус/и, растојание помеѓу редови и други карактеристики. Употребата на стилови овозможува уреден изглед на документот, автоматизирано креирање содржина (табела на содржина), брзо менување на изгледот на текстот и конзистентност низ целиот документ. Воедно, користењето стилови овозможува унифицираност на форматите на документите, негова едноставна промена, подобра пристапност и автоматизирање на документите.

Има **три основни вида стилови**:

- **Стилови на пасус –** се применуваат на цели пасуси (на пр. Heading 1 (Наслов 1), Normal (Нормален или Основен);
- **Стилови на карактери –** се применуваат на поединечни зборови или фрази (на пр. Emphasis (Нагласено), Strong (Силно нагласено);
- **Табеларни стилови –** се применуваат на цели табели.

Дополнителни информации за секој стил може да најдеш во листата со стилови која се отвара со кликнување на копчето за отворање на дијалог прозорец во групата Styles или со кратенката Ctrl+Alt+Shift+S.



Слика 1 *Информации за стилови*

4.1.2 Примена на постоечки стилови

Microsoft Word нуди широк избор на претходно дефинирани стилови како што се Наслов 1, Наслов 2, Цитат и други. Овие стилови може да се применат со селектирање на текстот и избор на соодветен стил од табулаторот Home > Styles.

Слика 2 *Видови стилови*

ПРИМЕР

- **Heading 1 –** се користи за главни наслови на секции.
- **Heading 2 –** за поднаслови, поттеми.
- **Normal –** за обичен текст.
- **Quote –** за истакнување цитати или посебен текст.
- **Title и Subtitle –** за главниот наслов и поднаслов на документот.

Практична примена на стилови:

1. Означи го пасусот што сакаш да го форматираш!

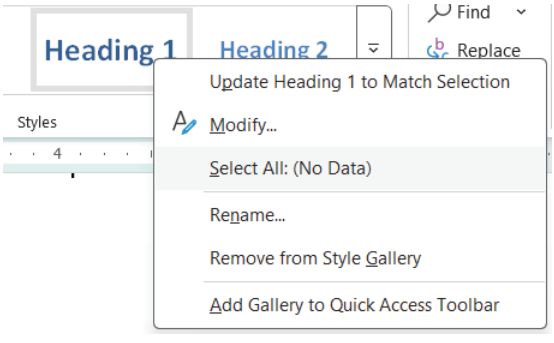
2. Избери стил од панелот Styles!

3. Ќе забележиш дека избраниот Стил автоматски ќе се примени на целиот пасус.

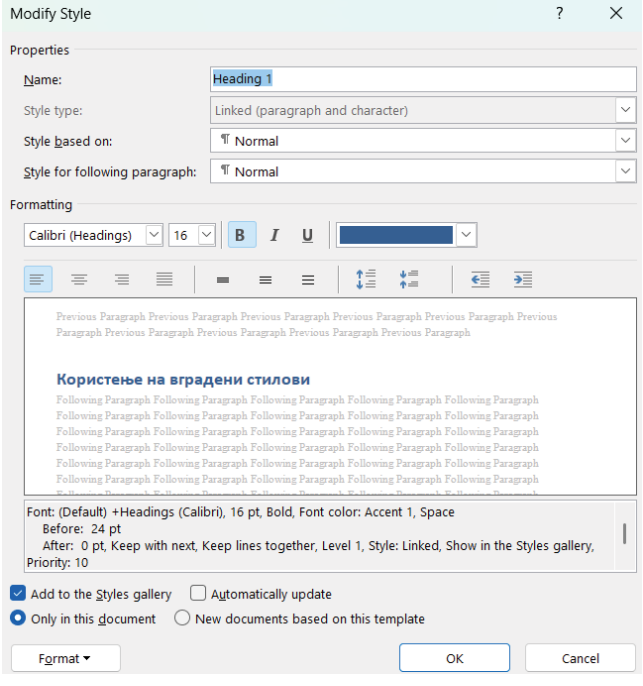
4.1.3 Уредување на постоечки стилови

Во случаи кога сакаш да направиш мали промени во изгледот на стилот (на пример, друг фонт или боја), не е потребно да го форматираш секој пасус рачно.

Доволно е да го уредиш самиот стил:



Слика 3 *Уредување постоечки стил*



Слика 4 *Прозорец за промена на стил*

Чекори за уредување:

1. Десен клик врз стилот што сакаш да го измениш (на пр. Heading 1).

2. Избери Modify...

3. Во прозорецот што ќе се појави, направи ги саканите промени: фонт, боја, големина, растојание, порамнување.

4. Со потврдата на OK, сите делови што го користат тој стил ќе се ажурираат автоматски.

Оваа функција е особено корисна за документи со повеќе страници каде што е потребна брза промена на дизајнот.

4.1.4 Креирање сопствени стилови

Ако документот бара специфичен изглед што не е застапен во постоечките стилови, можеш да креираш свој стил.

Чекори:

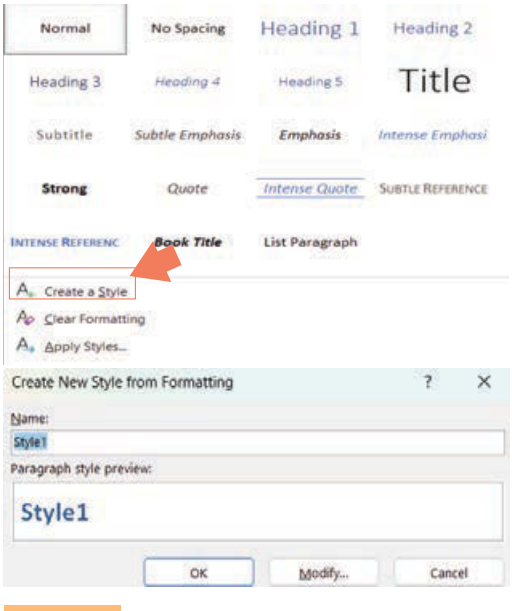
1. Форматирај го текстот како што сакаш да изгледа (тип на фонт, големина, боја, растојание).

2. Означи го текстот и кликни на Styles > Create a Style > New Style.

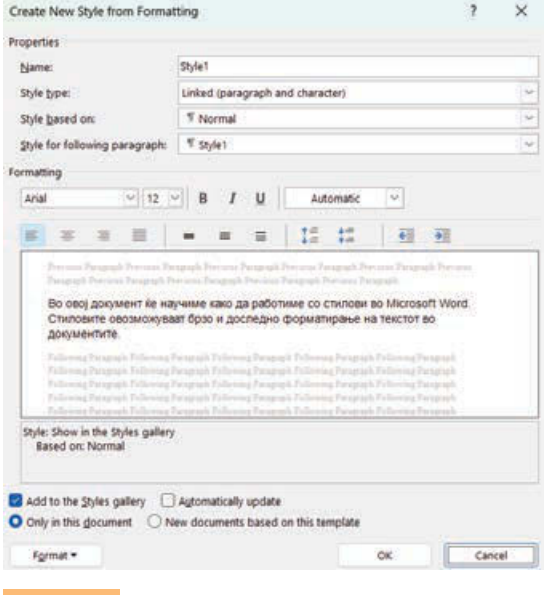
3. Додели име на стилот (на пр. „Поднаслов – сив“).

4. Потврди со OK и стилот ќе се појави во листата на достапни стилови.

Креирањето сопствени стилови е корисно кога работиш на документи со специфични стандарди или при креирање документи за печат, презентација или објавување.



Слика 5 *Прозорец за креирање нов стил*



Слика 6 *Прозорец за креирање нов стил*

Во лентата Name се пишува името на новиот стил со што тој ќе се појави во постоечката листа на стилови. Тој, подоцна може да се модификува со клик на копчето Modify според одредени барања.

- Во лентата Style се избира видот на стилот (за параграф, букви, табела, листа);
- Во лентата Based on се избира врз основа на кој веќе вграден формат ќе се базира;
- Во лентата Style for following paragraph се избира каков ќе биде стилот на следниот параграф;
- Во лентата Formating се избира вид на фонт, големина, боја, стил);
- Со избор на копчето Add to Quick Style List стилот ќе се прикаже во галеријата и во листата со стилови.

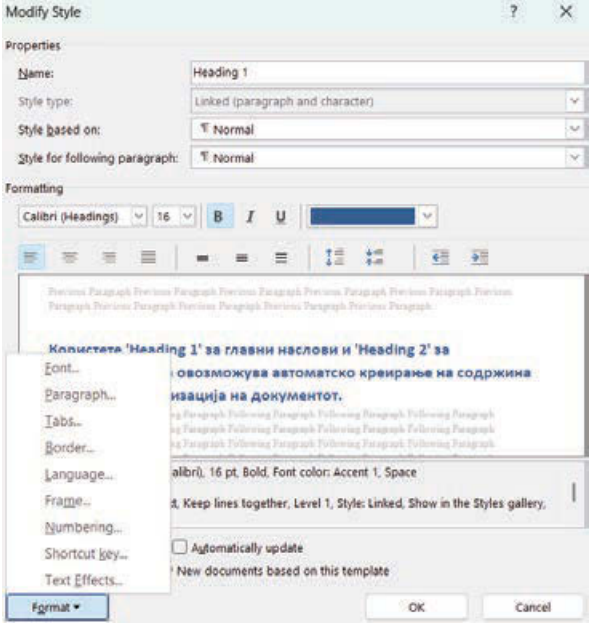
4.1.5 Наслов и поднаслов

Во секој професионално структуриран документ, **насловот** е првиот елемент што го привлекува вниманието. Вообичаено се форматира со стил Title и содржи:

- Голема големина на фонт,
- Централно порамнување,
- Избор на посебна боја или ефект.

Поднасловот, пак, го поддржува насловот со дополнително објаснување или детали. Тој се форматира со стил Subtitle, обично е со помала големина, но со сличен фонт и стил.

Исто така, можеш да користиш 'Heading 1' за главни наслови и 'Heading 2' за поднаслови. Ова овозможува автоматско креирање содржина и подобра организација на документот.



Слика 7 *Прозорец за уредување наслов и поднаслов*

ПРИМЕР

Напиши текст со наслов „Развој на дигиталната писменост кај учениците“ и поднаслов „Управување, пристап и алатки во ера на вештачка интелигенција“. Уреди го со научените техники за уредување стил и наслов.

4.1.6 Зошто да се користат стилови?

Користењето стилови носи повеќе предности:

- автоматско генерирање содржина (табела на содржина),
- конзистентност во изгледот,
- едноставно преуредување,
- професионален изглед,
- подобра навигација (особено при употреба на Navigation Panel).

Примената на стилови е неопходна особено во документи од тип на извештаи, есеи, дипломски работи, образовни прирачници или публикации.

Совети за доследно форматирање:

- **Користете стилови наместо рачно форматирање.**
- **Избегнувајте мешање на различни фонт типови и големини.**
- **Користете теми за унифициран изглед на документот.**

Примери за стилови во различни документи:

- **Извештаи:** Наслови, поднаслови, текст, цитати.
- **Есеи:** Наслов, вовед, главен дел, заклучок.
- **Писма:** Заглавие, поздрав, содржина, потпис.

4.1.7 Заклучок

Работата со стилови во Microsoft Word е суштинска вештина за секој што креира документи со повеќе секции или наменети за формална употреба. Покрај тоа што се применуваат веќе дефинирани стилови, корисникот има можност да ги уредува или да креира нови, прилагодени стилови според конкретните потреби. Разбирањето на поимите наслов и поднаслов, како и нивната правилна примена, дополнително го збогатува документот и го прави поразбирлив и визуелно привлечен. Со користење на стиловите, документите стануваат поорганизирани и полесни за читање.

4.1.8 Практична вежба: Работа со стилови во Microsoft Word

Цел на вежбата:

- да се применат постоечки стилови;
- да се уредат стилови според потребите;
- да се креираат нови стилови;
- да се користат наслов и поднаслов правилно.

1. Подготовка на документот

1. Отвори нов Word документ.

2. Внеси го следниот текст, без да применуваш никакво форматирање:

Историја на вештачката интелигенција

Вештачката интелигенција (ВИ) е гранка на компјутерските науки која има цел да создаде системи што можат да размислуваат, учат и решаваат проблеми како луѓе.

Подобласт: Машинско учење

Машинското учење е метод на анализа на податоци кој автоматизира градење на аналитички модели.

Подобласт: Длабоко учење

Длабокото учење користи невронски мрежи со повеќе слоеви за анализа на комплексни податоци како што се слики и звук.

Заклучок

Вештачката интелигенција се развива брзо и има огромен потенцијал да го трансформира општеството.

2. Примена на постоечки стилови

1. Означи ја фразата **Историја на вештачката интелигенција** → примени **Title стил**.

2. Означи ги зборовите **Подобласт: Машинско учење**, **Подобласт: Длабоко учење** и **Заклучок** → примени **Heading 2 стил**.

3. За останатиот текст во пасусите користи **Normal стил**.

3. Уредување на постоечки стил

1. Десен клик на Heading 2 → Modify...
2. Промени:

Боја на текст: темносина,

Големина: 14,

Додај Bold,

Порамни го лево.

3. Потврди со OK.

4. Креирање сопствен стил

1. Означи го пасусот што почнува со „Вештачката интелигенција (ВИ)...“

2. Во табулаторот Styles, избери Create a Style → New Style.

3. Име: **Вовед**

4. Избери:
   - Фонт: Georgia, 12 точки
   - Италик
   - Простор помеѓу редови: 1,5
   - Justify порамнување

5. Потврди со OK и примени го стилот.

5. Дополнителна задача (по избор)

Додај автоматска **табела на содржина** на почетокот од документот преку: References > Table of Contents > Automatic Table 1

ПРАКТИЧНИ ЗАДАЧИ:

1. Отвори нов Word документ и напиши краток текст со три наслови и неколку параграфи. Примени стилови „Heading 1“, „Heading 2“ и „Normal“!

2. Креирај нов стил со име „Мој стил“ кој користи син фонт, големина 14 pt и порамнување по средина! Примени го на еден параграф!

3. Уреди го стилот „Heading 1“, така што ќе користи црвена боја и Bold фонт! Примени го на сите главни наслови во документот!

4. Генерирај автоматска содржина базирана на стиловите што си ги применил!

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Кој таб во Microsoft Word го содржи панелот со стилови?

а) Insert.

б) Home.

в) Layout.

г) Review.

2. Што се случува кога ќе примениш стил „Heading 1“ на текст?

а) Текстот се претвора во табела.

б) Се применува форматирање за главен наслов.

в) Се брише текстот.

г) Се додава слика.

3. Кој е точниот начин за уредување на постоечки стил?

а) Двоен клик на стилот.

б) Десен клик > Modify.

в) Клик на Insert > Style.

г) Клик на File > Options.

4. Што овозможува користењето на стилови во документ?

а) Само форматирање.

б) Доследен изглед и автоматска содржина.

в) Повеќе грешки.

г) Само промена на боја.

5. Објасни што е стил во Microsoft Word и зошто е корисен!

6. Како можеш да креираш сопствен стил во Word?

7. Кои се предностите од користење стилови за наслови и поднаслови?

8. Што се случува ако промениш стил кој веќе е применет на повеќе места во документот?

4.2 Содржина и индекси

Кога работиме на подолги документи (реферати, есеи, проекти, книги), многу е важно тие да бидат прегледни и лесни за навигација. За таа цел, Microsoft Word нуди алатки за креирање содржина (табела на содржина) и индекс, кои автоматски се генерираат и се ажурираат врз основа на стиловите и ознаките во текстот.

4.2.1 Што е содржина (табела на содржина)?

Содржина е листа од сите главни делови и поднаслови во еден документ и вообичаено се поставува на почетокот на документот. Таа служи како патоказ за читателот и овозможува брзо преминување до саканиот дел.

Како се креира содржина?

За да можеш да креираш содржина, потребно е прво да користиш стилови на наслови („Heading 1“, „Heading 2“, итн.) за сите наслови и поднаслови во документот што го пишуваш.

Со следните чекори можеш да креираш содржина:

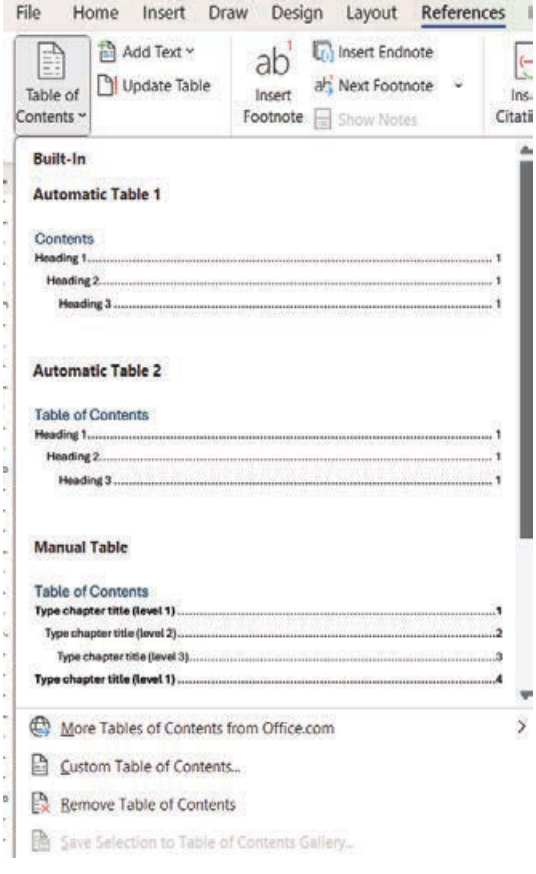
1. Постави го курсорот каде што сакаш да се прикаже содржината (на пр. на почетокот на документот).

2. Од лентата на алатки избери: References (Референци) → Table of Contents (Содржина).

3. Избери стил на содржина од понудената листа (Automatic Table 1 или 2).

4. Word автоматски ќе генерира содржина од сите елементи форматирани со стилови на наслови.

Генерираната содржина ќе содржи броеви на страници и хиперврски со кои може едноставно да се навигира низ документот до бараните наслови.



Слика 7 *Креирање содржина во документ*

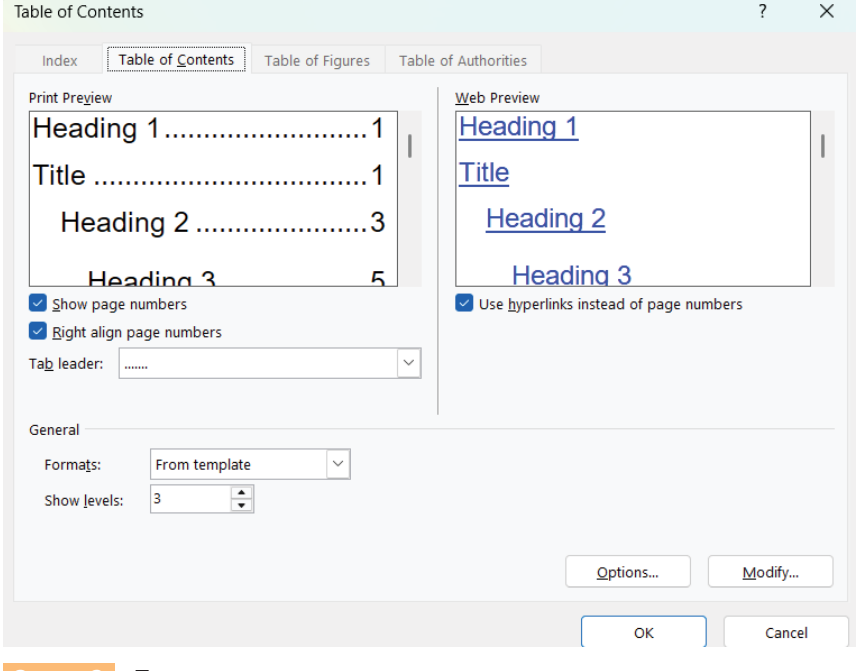
Совет:

Користи постојано стилови за наслови низ целиот документ за поедноставно уредување на содржината. Исто така, страниците треба да се нумерирани. Програмата, при генерирање содржина, ги бара насловите и ги додава во табелата на содржината со соодветниот број на страница.

4.2.2 Прилагодување на содржината

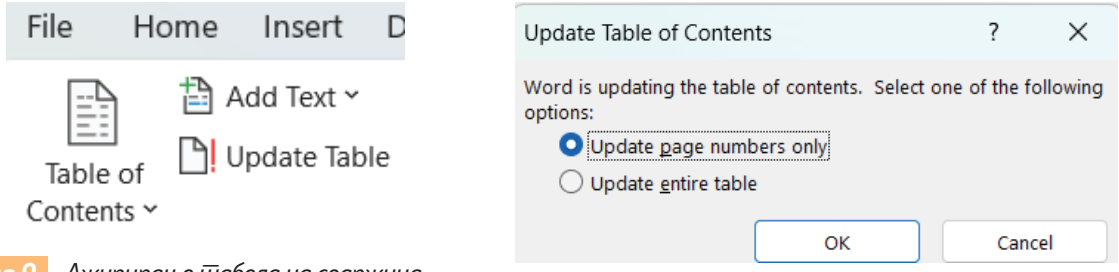
Содржината може да се прилагоди според потребите со:

- Вклучување на повеќе нивоа на поднаслови (на пр. Heading 3);
- Промена на изглед со повторен избор на References (Референци) → Table of Contents (Содржина) и се избира Custom Table of Contents со што се појавува прозорец како на слика 8;
- Со избор на опцијата Show page number се избира дали ќе се прикажат редните броеви на страниците;
- Со избор на опцијата Right align page numbers се избира дали броевите да бидат израмнети од десната страна;
- Во листата Tab leader се избира стил на линија водилка до броевите на страниците (празен простор, линија, точки или цртички);
- Во полето Format се избира еден од форматите на табелата на содржина;
- Во полето Show levels се определува до кое ниво ќе се прикажат насловите;
- Доколку документот ќе се објавува на веб-локација елементите на содржината треба да се уредуваат како хиперврски. За таа цел се потврдува опцијата Use hyperlinks instead of page numbers;
- На крајот избраните поставувања се потврдуваат со избор на копчето OK.



Слика 8 *Прилагодување содржина во документ*

Табелата со содржина може да се ажурира и при промени во текстот со опцијата Update Table. Се избира Update page numbers only доколку се ажурира само бројот на страниците или Update entire table со која се врши целосна промена.



Слика 9 *Ажурирање табела на содржина*

Добро организиран документ треба да има:

- Јасна структура: наслов → поднаслов → текст.
- Хиерархија на информации: главна тема (Heading 1), поттеми (Heading 2), потпоттеми (Heading 3).
- Употреба на празни простори, набројувања и слики кога е потребно.

4.2.3 Што е индекс?

Индекс е список на клучни зборови или поими од документот, прикажани на крајот заедно со страниците каде што се појавуваат. Вообичаено се користи во книги, енциклопедии или научни трудови.

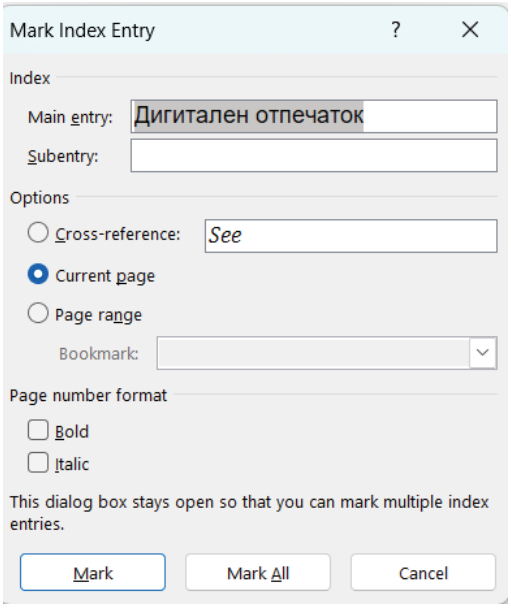
Слика 10 *Индекс*

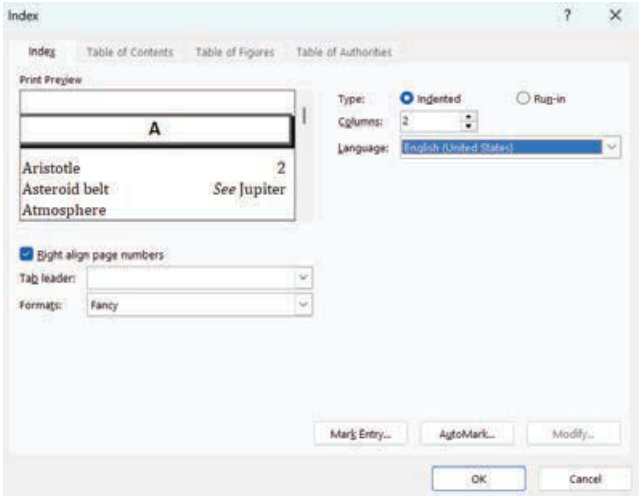
Како се додава список или табела со индекс?

- Означи го зборот или поимот што треба да биде дел од индексот;
- Одбери ја картичката References и потоа одбери Mark Entry;
- Во полето „Main entry“ се појавува означениот збор;
- Кликни на Mark или Mark All ако се појавува повеќе пати. MS Word во документот додава специјално поле XE (елемент на индекс). Ова поле нема да се прикажува при печатење на документот исто како и ознаката за нов ред ¶ ;

  Дигитален·отпечаток{·XE·"Дигитален·отпечаток"·}¶
- Потоа постави го курсорот на крајот од документот.
- Одбери References → Insert Index.
- Избери формат во кој ќе се прикажува, тип на водилка, број на колони и потврди го изборот со клик на OK.

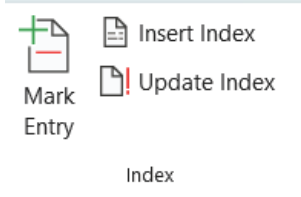
Индексот ќе се прикаже азбучно, со броеви на страниците за секој термин.





Слика 11 *Креирање индекс*

Доколку се додадат нови зборови индексот може да се ажурира со избор на опцијата Update Index.



За да се отстрани означување на некој збор или поим, се селектира ознаката XE (заедно со заградите {} и зборот во заградите) и се кликнува на копчето Delete.

4.2.4 Практична активност

Задача: 1

1. Напиши краток текст со три наслови и поднаслови.

2. Примени ги стиловите (Heading 1 и 2).

3. Додај табела на содржина на почетокот.

4. Означи три збора како индекс и додај индекс на крајот.

5. Направи измени во документот и ажурирај ги и содржината и индексот.

Задача: 2

1. Креирај нов документ во Microsoft Word со наслов: „Дигитална писменост и користење на ИКТ во образованието“

2. Структурирај го документот со следните делови (секции):

↪ Наслов на документот:

- „Дигитална писменост и користење на ИКТ во образованието“ → Heading 1
- Вовед → Heading 1

- Што претставува дигитална писменост? → Heading 2
- ИКТ-алатки што се користат во училиштата → Heading 2
- Предности и предизвици при користење технологија во наставата → Heading 2
- Заклучок → Heading 1

Секој дел треба да содржи околу четири до пет реченици текст.

1. Примени соодветни стилови на секој наслов (Heading 1, Heading 2).
2. Додај автоматска табела на содржина на почетокот на документот:
   - References → Table of Contents → Automatic Table.
3. Означи ги за индекс следните зборови:
   - технологија, ученик, дигитална писменост
   - За секој збор: References → Mark Entry → Mark All
4. Додај индекс на крајот на документот:
   - References → Insert Index
5. Зачувај го документот со име: „Индексиран_документ_ТвоетоИме.docx“

Совет:

Кога ќе додадеш нов наслов или поднаслов во текстот, не заборавај да ја ажурираш табелата на содржина: Клик на содржината → Update Table → *Update entire*

ЗАКЛУЧОК

Работата со содржина и индекси во Microsoft Word овозможува подобра организација, прегледност и професионален изглед на текстуалните документи. Со користење на стиловите за наслови, можеме брзо да креираме автоматска табела на содржина која се ажурира според промените во документот. Индексот, пак, им помага на читателите лесно да ги најдат клучните зборови и теми. Овие алатки овозможуваат поефикасна работа со текстуални документи во секојдневното работење.

ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

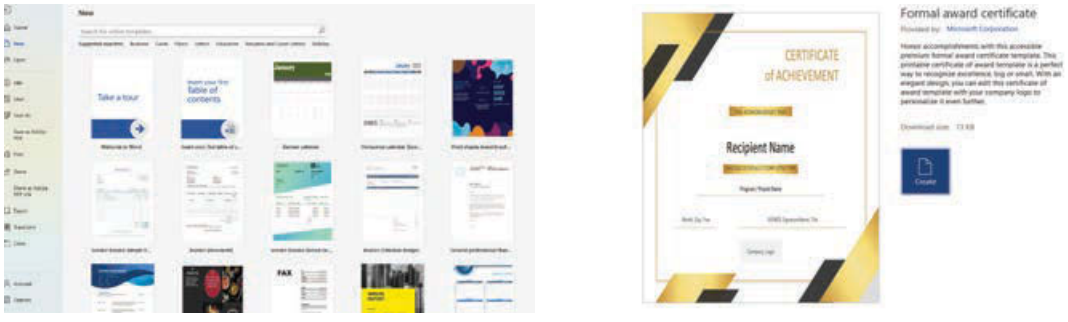
1. Како се создава табела на содржина во Word?
2. Зошто е важно да користиме стилови за наслови?
3. Како се означува збор за индекс?
4. Кога и како се ажурира табелата на содржина и списокот на индекс?
5. Што треба да направиш пред да креираш табела со индекс?
6. Кој дел од документот најчесто го содржи списокот со индекс?

4.3 Шаблони и формулари

Кога работиме со документи што често се повторуваат — како пријави, барања, договори или извештаи — можеме да користиме шаблони. Шаблоните (templates) се однапред подготвени документи со одредени формати, структура и содржина која може да се приспособи. Освен тоа, Word овозможува креирање формулари (форми) кои се користат за внесување податоци на стандарден начин.

4.3.1 Шаблон

При отворање нов документ, MS Word нуди голем број категории на готови шаблони (на пример, за писма, покани, сертификати, насловни страници за семинарски работи и други). Се креира со избирање на опциите File → New со што од десната страна се појавува збирката категории. Со опцијата **Search** може да се пребарува онлајн шаблон. Шаблоните се отвораат со клик врз категоријата со што се отвора лепеза од шаблони, се избира посакуваниот шаблон и се кликнува на копчето **Create**. Се отвора нов документ со дефинирани стилови, кој може да се уредува по сопствени потреби или одредени барања. Има различна наставка од обичен документ, .dotx и се чува во посебна папка во Documents → Custom Office Templates.



Слика 12 *Креирање шаблон (MS Word)*

Чекори: како се користи шаблон?

1. Отвори Word → File → New.
2. Во прозорецот избери шаблон за агенда за работилница за прва помош.
3. Кликни Create и прилагоди го документот според твоите потреби.
4. Кога ќе завршиш со уредување и пишување, зачувај го документот со име „Агенда за работилница".

Чекори: Како се креира сопствен шаблон?

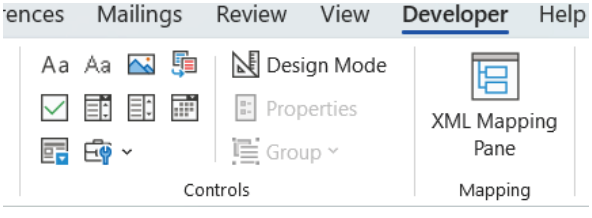
1. Направи нов документ.
2. Додај текст, заглавие, форматирање и елементи што се повторуваат.

3. Одбери: File → Save As.
4. Под Save as type избери: Word Template (.dotx).
5. Додели име (на пр. „Мој шаблон за извештај“) и зачувај го.
6. Новиот документ автоматски ќе се зачува во папката Custom Office Templates.

4.3.2 Формулар

Формулар е документ што содржи полиња за внесување податоци, како што се имиња, датуми, адреси и одговори на прашања и се користат за прибирање податоци со креирање прашалници, анкети, пријави итн. Содржи однапред дефинирани полиња како што се поле за контрола на текст, копче за избор, полиња за чекирање и паѓачки менија во кои може да се внесуваат податоци. Исто така, содржи и текст кој не може да се менува, на пример, прашањата за анкета или барањата за пријава (Внеси датум на раѓање, име и презиме, омилена музичка група, итн.).

Формулар се креира со отворање нов документ и се избира картичката Developer. Потоа, во менито Controls се избираат полињата што треба да ги содржи формуларот. Доколку картичката Developer ја нема во лентата со картички, можеш да ја додадеш преку File → Options каде што се избера Customize Ribbon. Во прозорецот од десната страна, селектирај Developer и кликни OK.



Слика 13 *Креирање полиња во формулар (MS Word)*

1. Поле за контрола на текст (Text Control Field)

Ова поле им овозможува на корисниците да внесат текст во формуларот. На пример, кога некој пополнува име или адреса, тоа се прави во текстуално контролно поле. Се додава преку: Developer → Controls → Rich Text Content Control или се кликнува на копчето Aa.

2. Копче за избор (Option Button или Radio Button)

Ова е кружно копче што се користи кога треба да се избере само една од повеќе опции. Често се користи во прашалници (на пример: пол: → машки → женски). Се додава со клик на копчето.

3. Паѓачко мени (Drop-down list)

Паѓачкото мени е листата со избор што се отвора кога ќе се кликне на стрелката. Корисникот може да избере една од однапред дефинираните опции (на пример: градови, занимање, одделение). Се додава со клик на копчето.

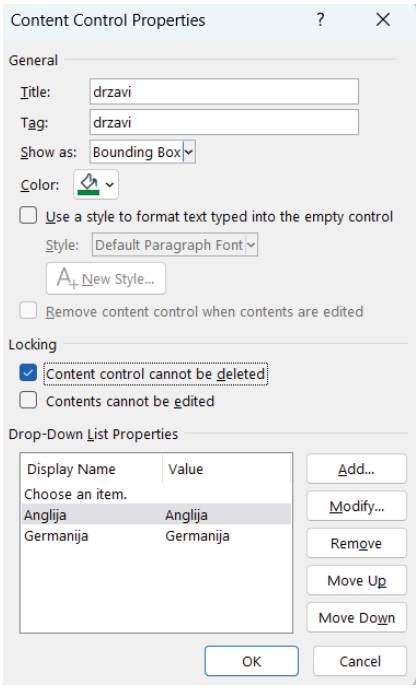
4. Поле за селектирање (Check Box)

Се користи за избор на повеќе опции (на пр. интереси: читање, спорт, музика). Се додава со клик на копчето .

5. Паѓачко мени (Drop-down List)

Паѓачко мени со однапред дефинирани опции и корисниците избираат која им е најсоодветна од понудените. Се додава со клик на копчето.

Односно **Drop-Down List Content Control,** потоа избери **Properties** за да додадеш опции со клик на копчето Add (слика 13). Можеш да додадеш и име на паѓачкото мени. Внесените опции може да ги менуваш со копчето Modify, да ги отстраниш со Remove и да го менуваш редоследот со Move up и Move down. Селектирај ја опцијата Content control cannot be deleted за корисникот да не го избрише полето. Доколку е селектирана опцијата Content control cannot be edited, корисникот нема да може да одбере ништо.



Слика 14 *Додавање опции во формулар (MS Word)*

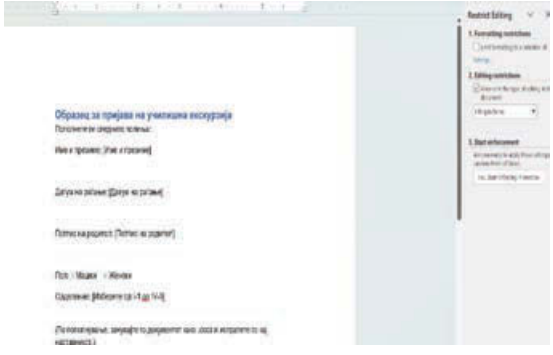
6. Поле за избор на датум

Во него се внесуваат датуми за избор, на пример, одржување состанок, избор на денови за екскурзија, итн.

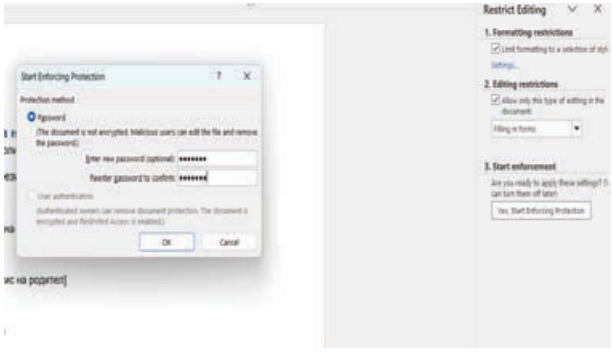
Заклучување на формуларот

Откако ќе го дизајнираш формуларот: избери Developer > Restrict Editing

Во делот „Editing restrictions“, избери Filling in forms, со што корисниците само ќе можат да пишуваат одговори и потоа кликни Yes, Start Enforcing Protection за да поставиш лозинка. Ова ќе им овозможи на корисниците да го пополнуваат формуларот, но да не го менуваат дизајнот.



Слика 15 *Заклучување формулар без лозинка (MS Word)*



Слика 16 *Заклучување формулар со лозинка (MS Word)*

Зачувување како шаблон

Избери File > Save As потоа избери тип на документ Word Template (*.dotx) и кликни Save. Ова ќе ти овозможи да го користиш формуларот повеќепати.

ЗАКЛУЧОК

Користењето шаблони и формулари во Word го олеснува креирањето на стандардизирани документи. Шаблоните заштедуваат време и обезбедуваат доследен изглед, додека формуларите овозможуваат структура за прецизно и уредно внесување информации. Ученикот кој ќе ги совлада овие алатки, ќе биде способен да подготвува документи на напредно ниво, прилагодени за реални образовни и административни ситуации.

ПРАКТИЧНА ЗАДАЧА

Креирање образец за пријава на ученик во училишна активност

1. Отвори нов документ во Word.
2. Внеси заглавие: „Образец за пријава на училишна екскурзија“.
3. Вклучи ги следниве полиња:
   - Име и презиме → Text Field
   - Одделение → Drop-down list (од I-1 до IV-5)
   - Пол → Option Buttons (Машки / Женски)
   - Датум на раѓање → Text Field
   - Потпис на родител → Text Field
4. Активирај заштита со Restrict Editing, така што ќе може само полињата да се пополнуваат.
5. Зачувај го документот како шаблон: Save as → Word Template (.dotx)

ПРАКТИЧНА ЗАДАЧА

1. Отвори нов документ во Microsoft Word.
2. Активирај ја лентата „Developer“ преку File > Options > Customize Ribbon и означи „Developer“.
3. Креирај формулар за пријава на настан со следниве елементи:

Формуларот треба да содржи:

- Текстуални полиња за: Име, Презиме, Е-пошта
- Паѓачко мени за избор на град (на пр. Скопје, Битола, Тетово)
- Копчиња за избор (Radio Buttons) за пол: Машки / Женски
- Поле за коментари (Text Area)
- Поле за избор (Check Box) за интереси: Технологија, Уметност, Спорт

ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Што е шаблон и за што се користи?

2. Како се разликува форма од обичен документ?

3. Каде се наоѓаат контролите за текст и паѓачко мени?

4. Како се зачувува сопствен шаблон?

5. Што е разликата помеѓу копче за избор и паѓачко мени?

6. Каде се наоѓаат алатките за креирање формулари во Word?
а) File.
б) Developer.
в) Insert.
г) View.

7. Кое поле се користи за внес на текст од корисникот?
а) Drop-down.
б) Text Control.
в) Check Box.
г) Table.

8. Кое копче се користи кога треба да се избере само една опција?
а) Check Box.
б) Drop-down.
в) Radio Button.
г) Text Field.

9. Што овозможува паѓачкото мени?
а) Внес на слободен текст.
б) Избор од понудени опции.
в) Вметнување слика.
г) Печатење.

10. Дали можеме да заклучиме формулар за да се пополнува, но не и да се уредува?
а) Не.
б) Да.

11. Објасни ја улогата на лентата Developer во Word!

12. Кои типови контроли може да се користат во еден формулар?

13. Опиши како се заклучува формулар за пополнување!

4.4 Заштита на документи

Заштитата на Word документите е важна за да се спречи неовластено читање, уредување или бришење на содржината. Microsoft Word нуди неколку начини за заштита на документите.

Видови заштита

1. Само за читање (Read-only):

Овој режим овозможува документот да се отвори, но не и да се уредува без дозвола на сопственикот. Корисникот ќе добие известување дека документот е само за читање.

2. Заштита со лозинка (Password protection):

Може да се постави лозинка за отворање или уредување на документот. Без точната лозинка, документот не може да се отвори или да се измени.

3. Ограничување на уредувањето (Editing restrictions):

Овозможува да се дозволи само одреден тип уредување, како пополнување формулари, коментари или следење на промени.

Чекори за поставување заштита

1. Отвори го документот во Microsoft Word.

2. Оди во табот 'File' > 'Info'.

3. Кликни на 'Protect Document'.

4. Избери една од опциите:

- 'Always Open Read-Only'
- 'Encrypt with Password'
- 'Restrict Editing'

5. Следи ги инструкциите за поставување лозинка или ограничувања.

6. Зачувај го документот.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Дали може да се постави лозинка за отворање Word документ? (Да/Не)

2. Дали режимот 'Read-only' дозволува уредување? (Да/Не)

3. Дали може да се дозволи само пополнување формулари во документ? (Да/Не)

4. Објасни како се поставува лозинка за документ во Word!

5. Кои се предностите на ограничување на уредувањето?

6. Во кои ситуации би користел режим 'Read-only'?

ПРАКТИЧНА ЗАДАЧА

1. Креирај документ со наслов ‹Мој проект›. Напиши краток текст. Потоа постави заштита на документот така што:

2. Ќе биде само за читање.

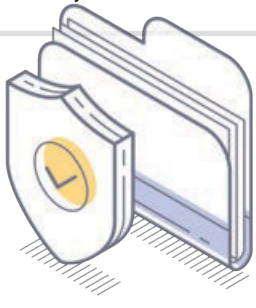
3. Ќе има лозинка за отворање.

4. Ќе дозволува само пополнување формулари.

5. Зачувај го документот и обиди се да го отвориш без лозинка.

ЗАКЛУЧОК

Со користење на алатките за заштита во Word, можеме да ги зачуваме доверливоста и интегритетот на нашите документи. Важно е да се знае како да се постават различни типови заштита и да се применат во практични ситуации.



Што научив

Работата со документи во Microsoft Word не се сведува само на пишување текст – таа вклучува вештини за уредување, организација, автоматизација и заштита на содржината. Во текот на оваа тема, учениците се запознаа со неколку клучни алатки и концепти.

Стилови

Со користење стилови, документите добиваат професионален изглед и доследност. Стиловите овозможуваат брзо форматирање наслови, поднаслови и текст, како и автоматско креирање содржина.

Содржина и индекси

Автоматската содржина (Table of Contents) и индексите се корисни за навигација и организација на поголеми документи. Тие се базираат на стиловите и овозможуваат лесно пребарување и прегледност.

Шаблони

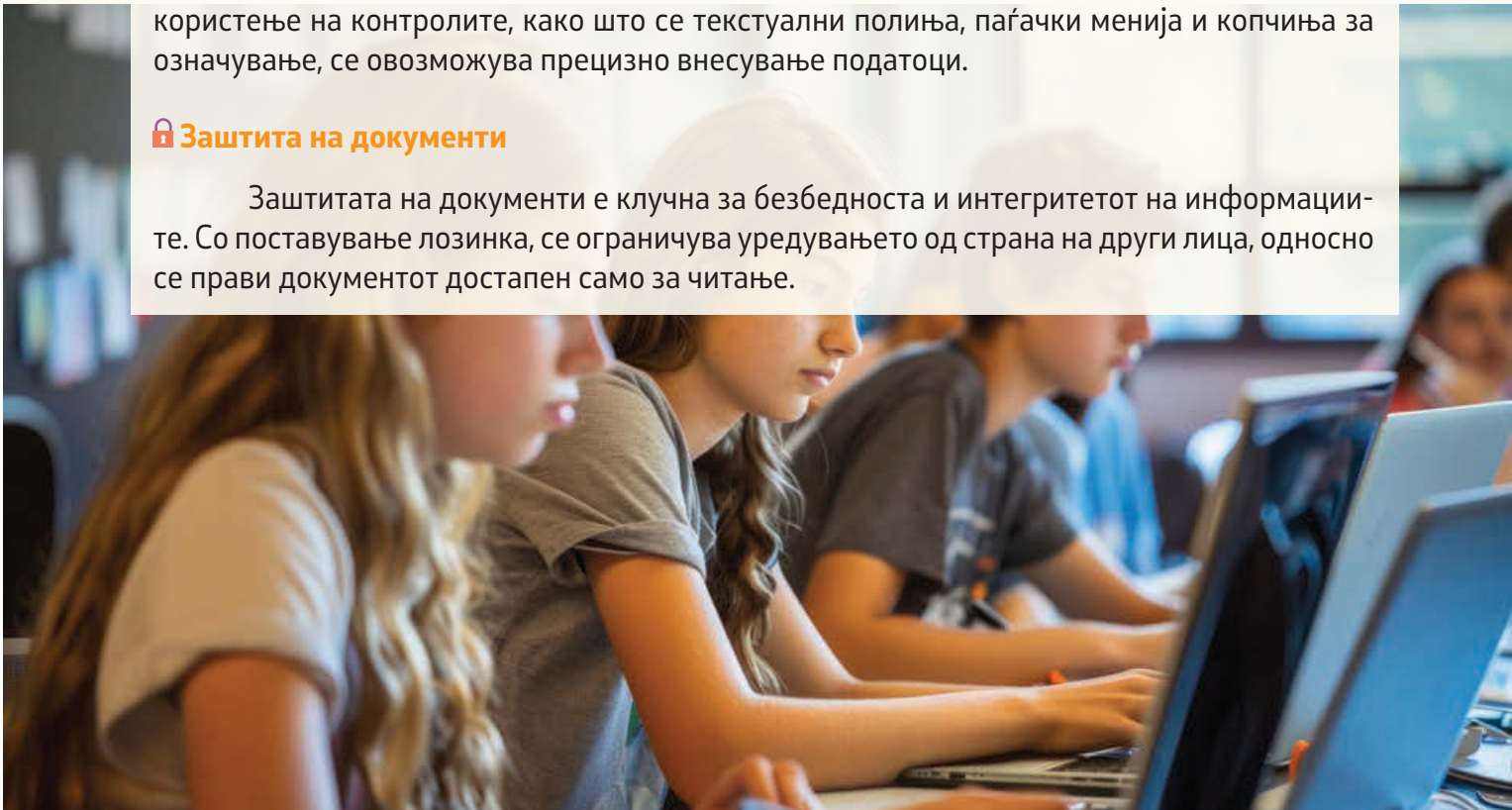
Шаблоните (templates) се однапред дефинирани формати кои штедат време и обезбедуваат унифициран изглед на документите. Се користат шаблони за различни намени – извештаи, писма, формулари и други.

Формулари

Формуларите се интерактивни документи во кои има полиња за пополнување. Со користење на контролите, како што се текстуални полиња, паѓачки менија и копчиња за означување, се овозможува прецизно внесување податоци.

Заштита на документи

Заштитата на документи е клучна за безбедноста и интегритетот на информациите. Со поставување лозинка, се ограничува уредувањето од страна на други лица, односно се прави документот достапен само за читање.



ПОВТОРУВАЊЕ ЗА ТЕМА 4

1. Што претставува „стил“ во текстуален документ? Наведи најмалку два примери (на пр: Heading 1, Normal)!

2. Објасни зошто е корисно да се користат стилови наместо рачно форматирање на текст!

3. Како би го уредил документот ако наставникот бара сите наслови да бидат со ист стил? Опиши ги чекорите!

4. Во документ имаш листа со термини. Како би креирал индекс што води до секој клучен збор?

5. Спореди ги можностите на шаблон (template) и формулар (form). Во што се слични, а во што се разликуваат?

6. Анализирај ја следната ситуација: еден документ има различни фонтови, боја на текст и големини! Како би ги организирал стиловите за да изгледа уредно?

7. Процени дали е безбедно да се прати документ преку електронска пошта без заштита, ако содржи чувствителни информации! Објасни ја твојата одлука!

8. Процени кој метод на заштита е поефективен во конкретен случај: лозинка, ограничување на уредување или само-читачки режим!

9. Креирај краток документ користејќи три различни стилови (наслов, поднаслов и текст) и објасни зошто ги избра тие стилови!

10. Изработи сопствен шаблон за училиштен извештај! Вклучи место за наслов, автор, датум и индекс! Опиши како би го користел!

ТЕМА 5: Програма за табеларни пресметувања

Програмите за табеларни пресметки претставуваат моќни дигитални алатки што овозможуваат систематско организирање, анализа и визуализација на податоци. Благодарение на нивната флексибилност и функционалност, тие наоѓаат широка примена во различни области од секојдневниот и во професионалниот живот — вклучувајќи ги образованието, деловното работење, научните истражувања и администрацијата. Овозможуваат лесна обработка на информации, креирање формули и автоматизирани пресметки, како и визуелен приказ преку графици и табели. Освен математички операции, поддржуваат и напредни функции како сортирање, филтрирање, условно форматирање и анализа на податоци, со што значително придонесуваат за поефикасно работење и донесување информирани одлуки. Најчесто користени програми за табеларни пресметувања се Microsoft Excel, Google Sheets како и Calc LibreOffice. Во овој учебник ќе биде објаснета функционалноста на Microsoft Excel – дел од Microsoft 365.

Нови поими

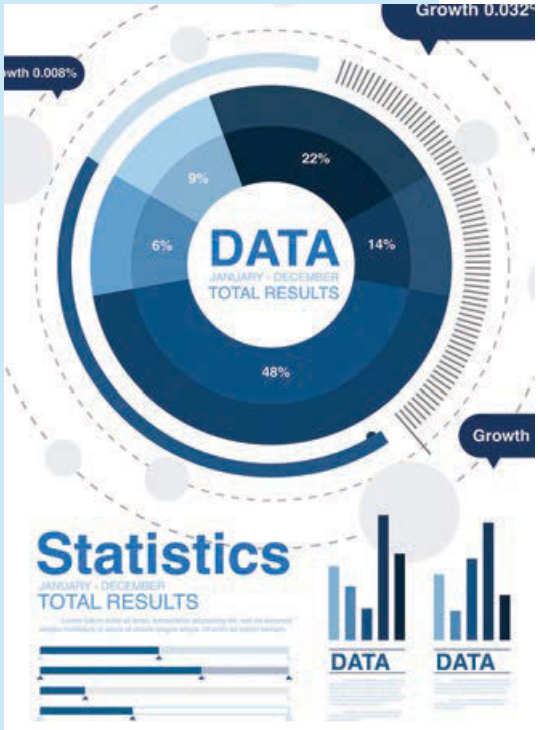
- табела, редови, колони, клетки, работен лист, додавање, бришење, преименување, преместување и копирање на работен лист, формула, функција, графикон, типови графикони, сортирање податоци, филтрирање податоци, пивот табела, заштита на клетки, заштита на работна книга

Веќе знаеш да...

- внесуваш податоци од различен формат;
- креираш и уредуваш едноставни табели;
- креираш едноставни формули за табеларни пресметки користејќи ја лентата за формули;
- користиш едноставни функции за пресметки со податоци во табела преку соодветно мени;
- креираш графикони од различен тип;
- филтрираш податоци во табела;
- сортираш податоци во табела.

ВО СОДРЖИНИТЕ ШТО СЛЕДУВААТ ЌЕ НАУЧИШ ДА:

- креираш и уредуваш табела во програма за табеларни пресметувања;
- создаваш база на податоци;
- применуваш формули и функции на напредно ниво за пресметување податоци во табела;
- избираш и уредуваш различни типови графикони, според дадени критериуми;
- подредуваш (сортираш) и филтрираш податоци од табела, според дадени критериуми;
- користиш условно форматирање во креирање интерактивна табела;
- користиш апсолутно и релативно адресирање;
- уредуваш пивот табели;
- применуваш заштита на податоци во табела.



ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

Замисли дека си дел од тим што организира музички фестивал. Ќе има бендови, добра музика, вкусна храна и многу забава. Твојата задача е да подготвиш табела во Excel и да пресметаш колку пари ќе бидат потребни за сите подготовки – изнајмување бендови, озвучување, реклами, храна за волонтерите и сè што е потребно за фестивалот да биде одлично организиран!

5.1 Табела како база на податоци

Базата на податоци е организирана збирка на податоци што се лесно пристапни и може да се управуваат, ажурираат и да се анализираат. Наместо податоците да се чуваат расфрлано или во несредени списоци, базата овозможува систематско складирање на информациите, со што се зголемуваат ефикасноста и сигурноста при нивна употреба. Базите на податоци се користат во речиси сите области на современата технологија — од мали лични апликации, па сè до сложени системи во банкарството, здравството, образованието и во е-трговијата. Основните карактеристики на базата на податоци се: организираност, структура, поврзаност меѓу податоците, можност за пребарување, заштита и одржување на интегритетот на податоците.

Табелите претставуваат основен и најчесто користен елемент во базите на податоци. Тие служат како основна структура за складирање на информациите. Секоја табела е составена од редови и колони, при што клетката претставува поле, редот (или запис) претставува поединечен елемент на податок, а колоната претставува една од неговите карактеристики. Името на колоната всушност претставува име на полето. На пример, табела со податоци за ученици може да има колони со име „Име“, „Презиме“, „Одделение“ и „Број на изостаноци“, додека секој ред ќе претставува еден ученик со конкретни вредности за тие полиња. Она на што треба да се внимава, е да нема празни редици и колони во табелата.

Во табелите може да се содржат различни типови податоци: текст (како имиња и адреси), броеви (како оценки и цени), датуми, логички вредности (да/не), па дури и поврзани информации од други табели. Оваа разновидност овозможува табелите да бидат флексибилни и применливи во различни ситуации. На пример, во апликација за онлајн продавница, една табела може да чува информации за производи, друга за корисници, а трета за нарачки — при што сите се поврзани преку заеднички вредности (клучеви), што ја создава моќната мрежа на податоци во релационите бази. Во овој учебник повеќе ќе се задржиме на работа со табелите и пресметките. Начинот на кој се креираат релации во база на податоци ќе го научиме следната учебна година.

Табелите се клучни за управување со информации, бидејќи обезбедуваат јасна структура, овозможуваат брзо пребарување и филтрирање и гарантираат дека податоците остануваат конзистентни. Без нив, базата би била само несреден куп податоци без смисла и контрола. Затоа, разбирањето на табелите и нивната правилна употреба е основа за секој што сака да научи како функционираат базите на податоци и како се користат во реалниот свет.

ДА СЕ ПОТСЕТИМЕ:

Табела

Табела е структура за уредено прикажување податоци, составена од редови и колони. Се користи за систематско прикажување и анализа на информации. Табелите може да содржат текст, броеви, формули или други податоци.

Редови

Редови во табела се хоризонтални линии што одат од лево кон десно. Секој ред претставува еден запис или една целина од поврзани податоци. На пример, еден ред може да ги содржи сите податоци поврзани со еден производ или со еден ученик. Редовите се означени со броеви (1, 2, 3, ...), и секој нов запис обично се става во нов ред.

Колони

Колоните се вертикални линии, означени со букви од горниот дел на работниот лист (A, B, C, ...). Секоја колона содржи податоци од ист вид. На пример, една колона може да биде резервирана само за имиња, друга за оценки, трета за датуми итн.

Клетки

Клетка (или ќелија) е пресекот на еден ред и една колона, и тоа е основната единица во која се внесуваат податоци. Секоја клетка има своја адреса според колоната и редот, Пример: A1 е клетката што се наоѓа во првата колона и првиот ред, додека C3 е во третата колона и третиот ред.

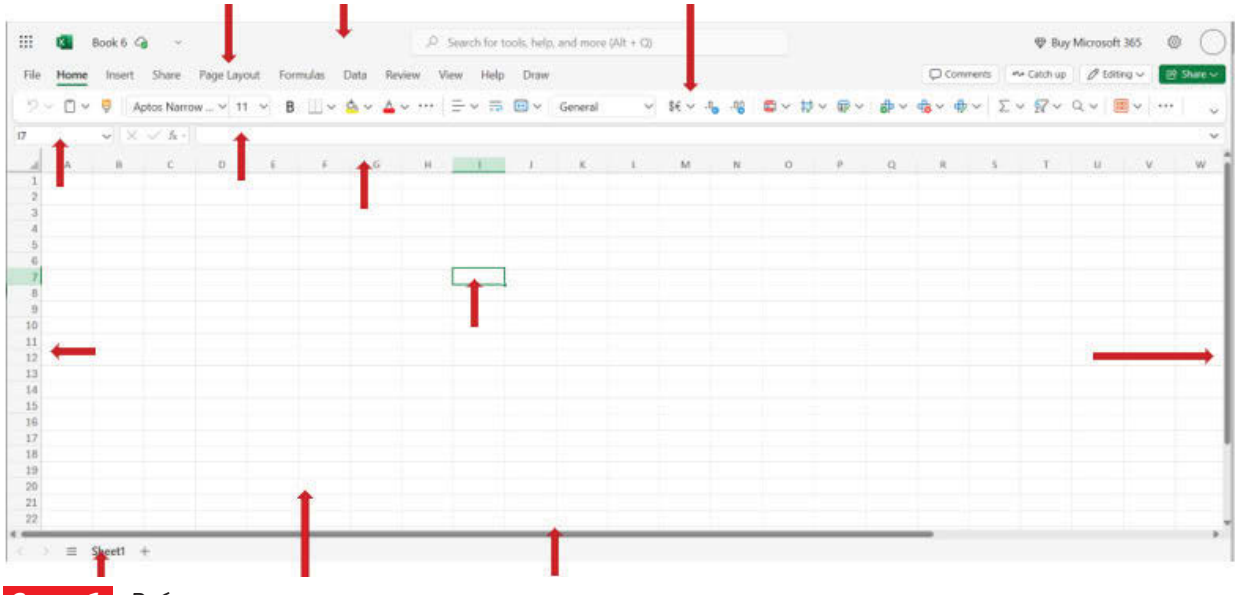
Работна книга

Работниот документ изготвен во програмата за табеларни пресметки се вика работна книга (или работна тетратка) – Workbook. Работната книга се состои од работни листови – Worksheet.

Работен лист (worksheet)

Работен лист е една страница во работната книга. Секој работен лист содржи мрежа составена од редови и колони и се користи за различни цели – од едноставни списоци до сложени пресметки и анализи. Секоја работна книга може да содржи еден или повеќе работни листови, што се означени со јазичиња на дното (Sheet1, Sheet2...) во кои се впишуваат, обработуваат и зачувуваат податоци (броеви, текст, формули). Работните листови можеш да ги додаваш и да ги бришеш. Податоците во работните листови може да бидат целосно независни едни од други, но може да бидат и меѓусебно поврзани.

Кои се елементите на еден работен прозорец во програма за табеларни пресметки? Именувај ги означените делови од слика 1!



Слика 1 *Работен прозорец*

Поими: Лента со наслов, Лента со алатки, Работен лист, Вертикален лизгач, Активна клетка, Име на колона, Име на редица, Работна табела, Хоризонтален лизгач, Лента за формули, Лента со мени (наредби), Адреса на активна клетка / ранг / именуван објект.

5.1.1 Операции со работни листови

Кога работиш на поголеми проекти или со задачи во кои има голем обем на податоци, внесувањето сè на еден ист работен лист може да создаде неред и да ја отежни прегледноста. Наместо тоа, подобро е податоците да ги организираш во посебни работни листови, при што секој лист ќе има своја јасна намена или содржина, но сите заедно ќе припаѓаат на истиот документ.

Со ваков начин на организација, работата станува поедноставна и поефикасна – добиваш подобар преглед врз податоците, полесно управуваш со информациите и можеш да се насочиш само кон делот што го обработуваш, без да го изгубиш чувството за целината. Истовремено, се намалува ризикот од грешки и се овозможува попрецизна анализа.

Работата со повеќе работни листови бара извршување на различни активности како што се нивно додавање, преместување, преименување, копирање или бришење. Овие активности се нарекуваат операции со работни листови.

ДА СЕ ПОТСЕТИМЕ!

Додавање нов работен лист

- Кликни на плус (+) иконата до последниот работен лист (долу лево). Новиот лист ќе се појави веднаш до активниот лист и ќе има име „Sheet2“, „Sheet3“ итн.

Бришење работен лист

- Десен клик на името на листот што сакаш да го избришеш (долу кај табовите). Избери Delete од менито. Потврди го изборот со OK.

Преименување работен лист

- двојно кликни на името на листот, впиши ново име и притисни Enter или
- десен клик на листот > Rename > впиши ново име > потврди го изборот со OK.

Преместување работен лист

- Кликни и задржи го листот што сакаш да го преместиш, потоа влечи го (drag and drop) лево или десно. Кога ќе дојдеш до саканата позиција отпушти го кликот.

Копирање работен лист

- Десен клик на работниот лист > од прозорецот што се појавува се избира Copy Sheet > кликни каде сакаш да го копираш листот> Paste Sheet (Excel 365 online).
- Десен клик на листот > Move or Copy > во прозорецот што се појавува одбери каде сакаш да го копираш листот > чекирај го полето Create a copy > кликни OK.

Копијата ќе има исто име како оригиналот, но со додадена бројка (на пр. „Sheet1 (2)“).

5.1.2 Уредување на табела

Често се случува содржината во клетките да биде премногу долга или широка за да се прикаже целосно. Во такви случаи, многу е важно да умееш да ги прилагодиш димензиите на редиците и колоните според твоите потреби.

Во Excel постојат неколку можности за промена на димензиите на редовите и колоните.

Ширината на колоните може да се промени со неколку техники:

- со покажувачот се доаѓа до десниот раб на колоната (помеѓу буквите на колоната). Кога курсорот ќе го добие изгледот  со лев клик се влече во саканата насока;
- се избира Home > Format > Column Width... Во полето Column Width се впишува саканата вредност;
- десен клик на името на колоната и избор на Column Width од менито.
- Висината на редовите може да се промени со неколку техники:
- со покажувачот се доаѓа до долниот раб на редот. Кога курсорот ќе го добие изгледот  со лев клик се влече во саканата насока;
- се избира Home > Format > Row Height... Во полето Row Height се впишува саканата вредност;
- десен клик на името на редот и избор на Row Height од менито.

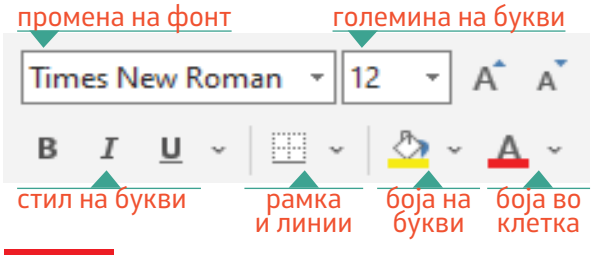
Ширината на повеќе колони или висината на повеќе редови може да се прилагоди одеднаш. За да го направиш тоа, најпрво треба да ги селектираш сите колони или редови чии димензии сакаш да ги измениш, а потоа ја применуваш истата постапка како при менување на димензија на поединечна колона или редица.

ЗАБЕЛЕШКА:

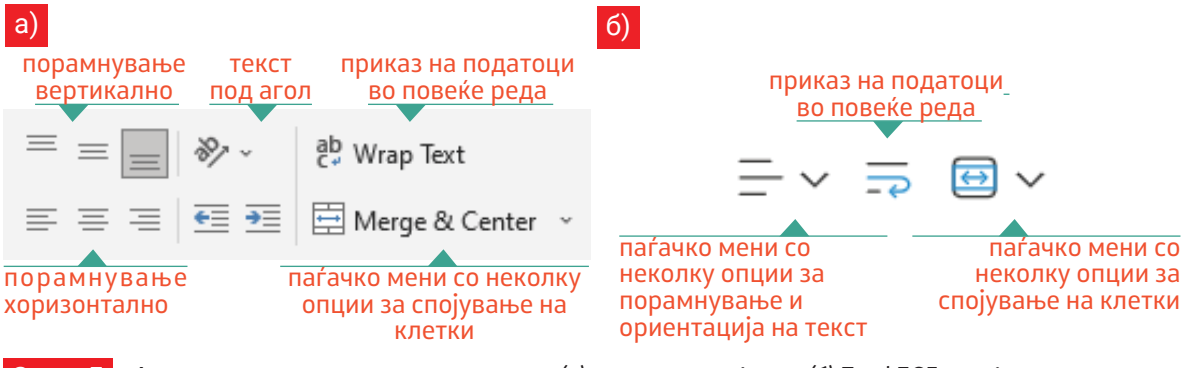
Ако во клетката со нумерички податок се појави знакот тараба (#) тоа значи дека во клетката нема доволно „место“ за податокот и дека е потребно колоната да се прошири.

5.1.3 Уредување клетки

За да бидат податоците јасни, прегледни и функционални, често има потреба од уредување на клетките. Ова опфаќа промена на форматот, промена на фонтот и изгледот на фонтот, порамнување на податоците во клетките, додавање бои и граници, спојување повеќе клетки и слично. Со правилно уредени клетки, табелите стануваат полесни за читање и анализа.



Слика 2 *Брзи алатки за форматирање клетки*



Слика 3 *Алатки за уредување текст и клетки (а) десктоп варијанта (б) Excel 365 онлајн*

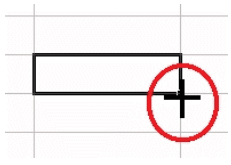
Уредувањето на табелата, односно на клетките, може да ги направиш и преку прозорецот Format Cells (во Number: можеш да го избереш и да го уредиш типот на податоците во Alignment: можеш да ги порамниш податоците; во Font: да го избереш и да го уредиш фонтот додека во Border: можеш да избереш стил, дебелина и боја на рабовите на клетките).

Забелешка: Форматирање на клетки со готов формат се изведува со избор на Cell Styles (во јазичето Home). Форматирањето на табелите со готов формат се изведува со избор на Format As Table (во јазичето Home).

5.1.4 Автоматско пополнување

Автоматското пополнување е многу корисна функција која овозможува автоматски да се пополнат низа од клетки врз основа на примерокот од претходните клетки. Примената на автоматското пополнување овозможува олеснета работа и заштеда на време при изработка на табелите.

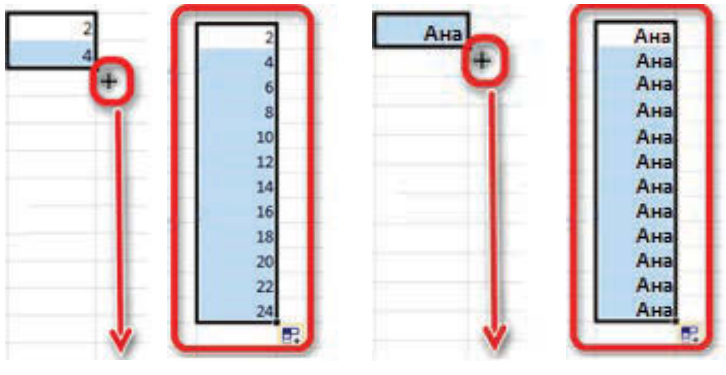
Автоматското пополнување најлесно се прави со повлекување на рачката за автоматско пополнување во која било насока. Рачка за автоматско пополнување всушност е малото црно квадратче кое се наоѓа на долниот десен агол од рамката на клетката. Кога со покажувачот ќе се дојде до него, тој добива форма „+“. Треба само да се држи левото копче на глувчето и да се повлече на страната каде што се сака да се пополнат клетките.



Слика 4 *Автоматско пополнување*

Всушност, постапката за автоматско пополнување е следната:

- Се избира некоја клетка која ќе се користи како основа за пополнување на дополнителните клетки. За група како што е 1,2,3,4,5.... се внесува 1 и 2 во првите две клетки. За група 2,4,6,8..... се внесува 2 и 4. За група 2,2,2,2.... се внесува само 2 во првата клетка. Текстуалниот податок се внесува само во првата клетка.
- Се селектираат внесените броеви/внесениот текст и се повлекува рачката за автоматско пополнување во посакуваната насока.



Слика 5 *Постапка за автоматско пополнување*

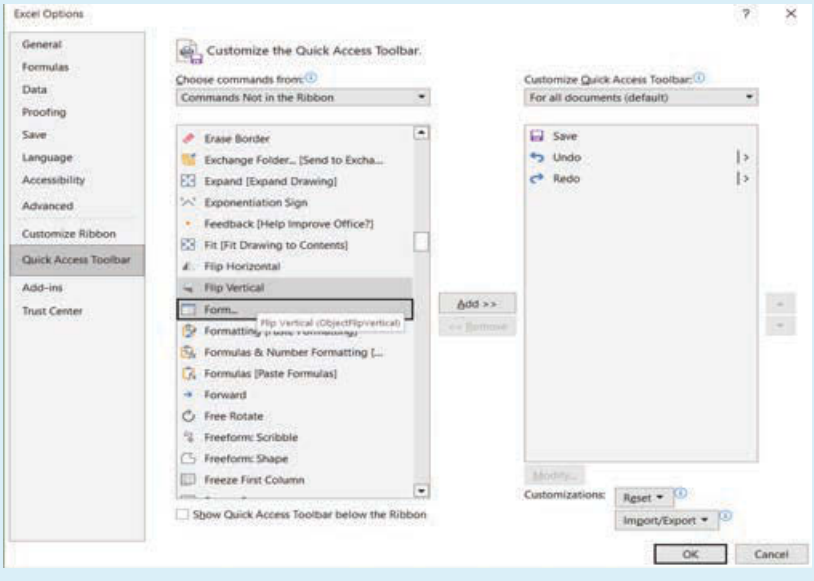
- По потреба, може да се кликне на опцијата за автоматско пополнување  и да се избере посакуваната опција.

ЗА ОНИЕ ШТО САКААТ ДА ЗНААТ ПОВЕЌЕ

Креирање образец за внесување податоци во база

Базите на податоци вообичаено содржат табели со многу податоци. Рачното внесување на овие податоци е: заморно и долготрајно, склоно кон човечки грешки и постои можност за случајна промена на претходни внесови. Постапката за внес на податоци во табела може да се подобри преку креирање образец. Но, најпрво треба да ја внесеш соодветната алатка во лентата со алатки за брз пристап.

Додавање Form во лентата со алатки



Слика 6 *Прозорец Excel Options*

Чекори:

1. отвори File -> Options - > Quick Access Toolbar,
2. во прозорецот Chose commands from одбери Commands Not in the Ribbon,
3. во долното пано кликни на Form,
4. кликни на Add,
5. Form ќе се префрли во десното пано,
6. потврди со OK.

Сега во алатките за брз пристап ќе се јави нова алатка Form.

Слика 7 *Алатка Form*

Изработка на образец

Пред употреба на образецот за внес на податоци ќе мора табелата да ја претвориш во „паметна“ Excel табела. Тоа можеш да го направиш со селектирање на табелата и избор на Insert -> Table. Во прозорецот Create Table, провери дали е означена опцијата „My table has headers“ .

	A	B	C	D	E
1	Име на производ	Категорија	Количина на залиха	Цена	Датум на набавка
2	телевизор	електроника	32	25,000.00 ден	24-Jan-2025
3	лаптоп	електроника	45	42,000.00 ден	24-Jan-2025
4	мобилен телефон	електроника	16	28,000.00 ден	20-Mar-2025
5	таблет	електроника	56	18,000.00 ден	11-Mar-2025
6	маици	облека	220	480.00 ден	5-Mar-2025

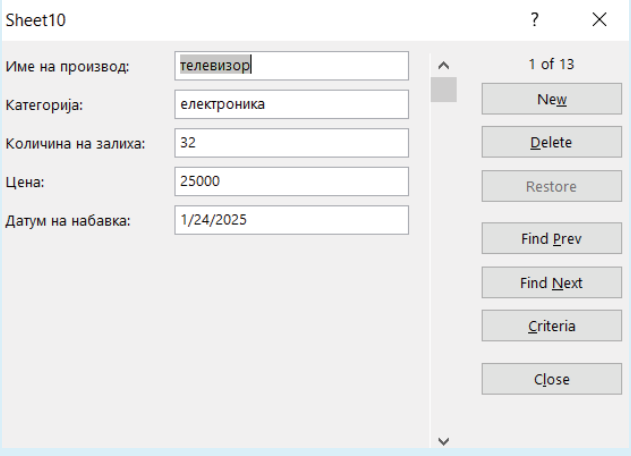
Слика 8 *Пример за Excel табела*

Чекори:

1. Кликни на која било клетка во твојата табела.

2. Во горната лента, кликни на новододаденото копче „Form“. Се отвора прозорец каде што можеш да внесуваш податоци во определените полиња.

3. Внеси ги податоците и кликни „New“ за да додадеш нов запис. Користи „Find Prev“ и „Find Next“ за да прелистуваш низ податоците. Ако сакаш да избришеш податок, притисни „Delete“.



Слика 9 *Образец за внесување податоци во пример табелата*

ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Што претставува работен лист во Excel и каде можеш да го најдеш?

2. Во програмата за табеларни пресметувања Сања внесува податок во една од клетките. Што од наведеното може да биде адреса на клетка во која Сања внесува податок?

а. AB1 б. 1ZA в. 1A1B г. A1B1

3. Кои операции со работни листови ги знаеш и објасни ја постапката за нивно изведување?

4. Зошто е корисно да се користат повеќе работни листови во еден Excel документ?

5. Што се подразбира под „ред“ во Excel? А што е „колона“?

6. Објасни ја постапката за истовремено менување на ширината на повеќе колони!

7. Кои типови податоци може да се внесат во клетка?

8. Зошто е важно правилно да ги уредиш клетките кога работиш со табеларни податоци?

9. Кои се чекорите за спојување на повеќе клетки во една?

10. Што претставува автоматското пополнување и како може да ти ја олесни работата при внесување податоци?

ПРАКТИЧНИ ВЕЖБИ:

1. Во програма за обработка на текст направи табела со наслов „Топ 5 омилени филмови /книги/ песни“ (избери една категорија која ти е блиска). Креирај табела во Excel со 5 реда кои ги претставуваат твоите топ 5 избори. Табелата треба да има 5 колони: | Ранг | Наслов | Жанр | Оценка (1–10) | Коментар |

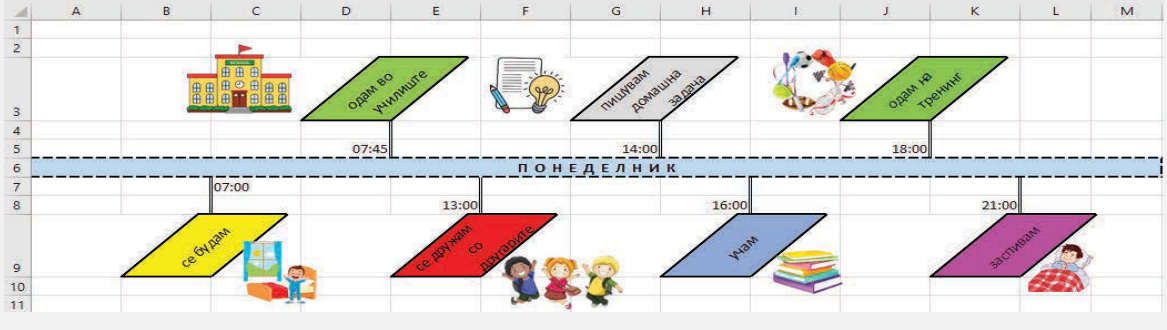
а) Наслов на табелата (на пример: „Моите топ 5 филмови“) да биде: Arial Black, големина 16, Центриран над табелата (споени клетки), во сина боја.

б) Насловите на колоните (Ранг, Наслов, Жанр, итн.): да бидат задебелени, центрирани, со фонт Calibri или Verdana, големина 12, Заднина: светлосива

в) Текстот во колоната Коментар да е накосен со големина 11.

г) Стави рамка на табелата.

2. Креирај хронолошка лента за тема по сопствен избор.



Слика 10 *Пример за хронолошка лента „Еден ден од моето секојдневие“*

5.2 Напредно користење формули и функции

Кога би имал табела со оценки – како најлесно би дознал кој ученик има највисок просек? Ако имаш табела со трошоци по месеци – како би ја нашол вкупната сума потрошена само за јануари? Кога имаш многу податоци, која програма може да ти овозможи нивна лесна организација во табели, брзо пресметување и прикажување на резултатите на разбирлив начин? Што користиш за да дојдеш до автоматски пресметки во табела, без да пресметуваш „рачно“?

5.2.1 Поими во програма за табеларно пресметување

ДА СЕ ПОТСЕТИМЕ:

- **Формула** е израз што се внесува во клетка и кој ѝ овозможува на програмата да направи пресметка врз податоците во табелата. Со формулата ѝ кажуваме на програмата кои вредности да ги земе, како да ги поврзе (на пример, да ги собере или одземе) и каде да го прикаже резултатот. Формулите може да содржат броеви, адреси на клетки, математички знаци и функции за добивање резултат.
- Примери за формули:
  - =(5+6)*9
  - =(B1+B2)-B12
  - =9+10+B6
  - =((B10)*10)/(B2+4)
- **Функциите** се опишуваат како однапред дефинирани формули кои се вградени во програмата.

Елементи на функцијата се **името на функцијата** и **аргументите.**

= ImeNaFunkcija (argumenti)

ImeNaFunkcija е збор што ја опишува задачата што треба да се изврши. Примери: SUM – за собирање, AVERAGE – за пресметка на просек, MIN – за пресметување минимална, најмала вредност, MAX – за пресметување максимална, најголема вредност.

Argumenti можат да бидат различни вредности зависно од тоа каква функција користиме и што сакаме да постигнеме. Можат да бидат текст, адреси на клетки, опсези на клетки, текст, логички вредности или други функции. Примери:

- SUM(A1:A10) аргумент е опсегот на клетки A1:A10, кој содржи броеви за собирање.
- AVERAGE(10, 20, 30) аргументите се константи, за кои ќе се пресмета просек.
- MAX (B2, B3, B4) аргументите се три одделни адреси на клетки B2, B3, B4 од кои ќе се прикаже најголемата внесена бројчена вредност.
- MAX (SUM(A1:A3), 100) аргументи се функцијата SUM(A1:A3) и константата 100.

. . .

Операторите се составен дел од формулите и функциите кои овозможуваат конкретни математички, логички или текстуални операции во рамките на формулата, односно функцијата. Постојат различни типови оператори, како што се:

- Аритметички оператори

+ (собирање), - (одземање), * (множење), / (делење) ^ (степенување).

Пример: Ако во клетката A1 имаш број 8, а во B1 број 4, формулата =A1 + B1 ќе врати 12 — бидејќи операторот + ги собира двата броја.

- Споредбени оператори > (поголемо), < (помало), = (еднакво), >= (поголемо или еднакво), <= (помало или еднакво), <> (не е еднакво).

Пример: Ако во клетката A1 е бројот 10, а во B1 е бројот 15, формулата =A1 < B1 ќе врати TRUE — бидејќи 10 е помало од 15.

- Текстуален оператор & (спојување на текст).

Пример: ако во клетката A1 се наоѓа зборот „Дигитален“, а во клетката B1 зборот „свет“, тогаш формулата =A1 & « « & B1 ќе прикаже: „Дигитален свет“ (овозможува додавање празно место помеѓу зборовите).

- Референтни оператори: (за опсег, се користи кога сакаме да избереме група клетки што се една до друга).

На пример: Ако имаш броеви во клетките од A1 до A6 и сакаш да ги собереш сите заедно, ќе напишеш =SUM(A1:A6), (за разделување, кога сакаме да избереме неколку клетки или опсези, но тие не се едни до други.).

Пример: со функцијата =SUM(A1, C1:C3) се собира вредноста од A1 и сите вредности од клетките од C1 до C3. (празно место, за пресек)

Пример: со функцијата =SUM(A1:A5 A3:C4) како резултат ќе се добие збирот на вредностите во A3 и A4, бидејќи тоа се клетките што се заеднички во двата опсега.

. . .

Веќе неколку пати го споменавме поимот опсег на клетки. Но, што всушност значи тој поим?

Опсег на клетки претставува група клетки што се избираат за работа со податоци. Притоа клетките може:

- да се наоѓаат во ист ред, на пример B2:D2 (во овој опсег се вклучени клетките B2, C2, D2),
- да се наоѓаат во иста колона, на пример A1:A3 (во овој опсег се вклучени клетките A1, A2, A3),
- да формираат правоаголна област, на пример A1:C3 (во оваа област се вклучени клетките A1, A2, A3, B1, B2, B3, C1, C2, C3).

. . .

Адресата на клетката е уникатен идентификатор што ја означува позицијата на клетката во работниот лист. Се состои од буквата на колоната (означена со букви: A, B, C...) и бројот на редот (означен со броеви: 1, 2, 3...)

Пример:

- A1 → Колона A, ред 1
- D5 → Колона D, ред 5

Има неколку видови адреси според тоа како се менува адресата кога се копира или се преместува формулата:

- Референтната адреса ја покажува позицијата на една клетка или опсегот на клетките во табелата, но не е фиксна — кога формулата со референтна адреса се копира или се преместува во друга клетка, адресата се менува автоматски според новата локација. Се пишува без никакви знаци пред буквата на колоната или бројот на редот.

На пример:

Во клетката D2 е формулата =B2*C2. Ако ја копираш формулата од D2 во D3, таа автоматски ќе се промени во =B3*C3

Тоа значи дека референтните адреси се прилагодуваат за да ја пресметаат содржината од „соседните" клетки, во зависност од тоа каде ќе ја ставиш формулата.

D2 | =B2*C2

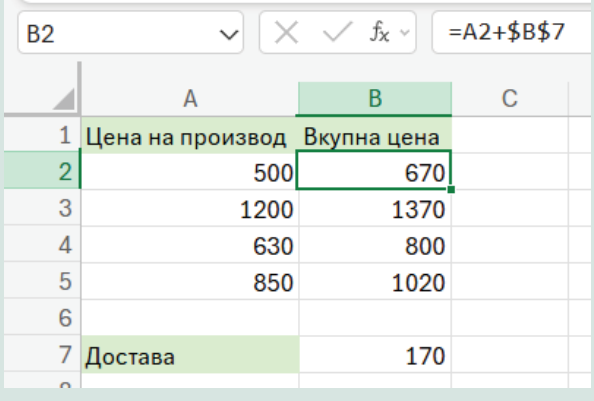
	A	B	C	D
1	Производ	Количина	Цена	Вкупно
2	Ранец	3	2,700.00 ден	8,100.00 ден
3	Шатор	1	9,300.00 ден	
4	Компас	1	4,000.00 ден	
5	Термос	2	1,200.00 ден	
6	Батериска лампа	5	950.00 ден	

Слика 11 *Пример за примена на референтна адреса*

- Апсолутна адреса е фиксирана адреса на клетка која останува иста кога ја копираме или ја преместуваме формулата во друга клетка. Таа секогаш покажува на точно одредена клетка. Се означува со додавање на знакот $ пред буквата на колоната и бројот на редот. Така $A$1 е апсолутна адреса која секогаш покажува на клетката во колона А, ред 1.

Пример: Во клетката B2 е формулата =A2+$B$7. Ако ја копираш формулата од B2 во B3, таа автоматски ќе се промени во =A3+$B$7. Значи, адресата $A$7 останува фиксирана (секогаш ја зема цената од A7), додека адресата A1 се менува на A2, A3 итн.

Има неколку видови адреси според тоа како се менува адресата на клетката кога формулата (функцијата) се копира или се премести:



Слика 12 *Пример за примена на апсолутна адреса*

- Мешана адреса е комбинација од апсолутна и релативна адреса што значи дека еден дел од адресата е фиксен, а другиот се менува кога ќе ја копираме формулата.

Пример: $A1 - Колоната „А“ е фиксирана, а редот „1“ се менува, A$1 → Редот „1“ е фиксиран, а колоната „А“ се менува.

5.2.2 Посложени функции

Досега научи да користиш основни функции како SUM, AVERAGE, MIN, MAX за да правиш едноставни пресметки во табелите. Но, што ако треба од табелата да добиеш попрецизни информации, како на пример:

- Колку ученици имаат оценка 5?
- Колку вкупно пари се потрошени само за храна?
- Како да провериме дали некој ученик има положено или не, врз основа на оценката?

Одговор на ваквите прашања даваат напредните функции како COUNT, COUNTIF, SUMIF и IF.

Со нив можеме:

- да броиме колку пати се појавува нешто (COUNTIF),
- да сумираме вредности според услов (SUMIF),
- да направиме избор или одлука во формула (IF),
- да пребројиме колку вредности има воопшто (COUNT).

ДА СЕ ПОТСЕТИМЕ:

Функцијата можеш да ја внесеш во клетка со директно запишување.

Чекори:

1. Кликни на клетката каде што сакаш да ја внесеш функцијата (селектирај ја).

2. Напиши = за да означиш дека внесуваш функција.

3. Внеси ја функцијата, на пример: =SUM(A1:A10).

4. Притисни Enter за да ја примениш.

5. Функцијата можеш да ја внесеш и преку менито Formulas.

Чекори:

1. Кликни на клетката каде што сакаш да ја внесеш функцијата.

2. Во лентата со менија кликни на табот Formulas.

3. Избери категорија на функции (од категориите: финансиски, логички, текстуални, функции за датум и време, функции за референци, за математички и тригонометриски функции).

4. Избери конкретна функција.

5. Внеси аргументи што ги користи функцијата.

6. Притисни Enter за да ја примениш.

. . .

Функција COUNT

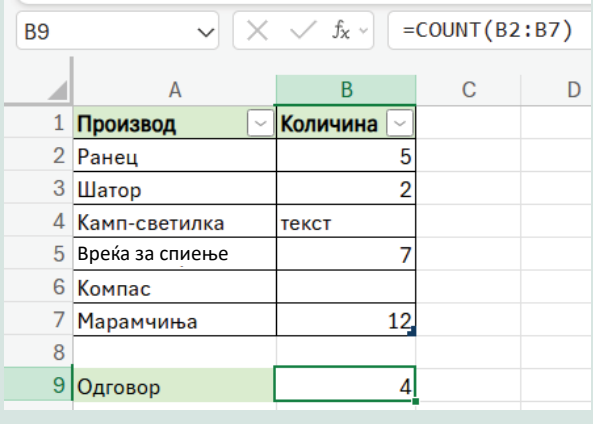
Функцијата COUNT брои колку клетки во избраниот опсег содржат бројчени вредности. Тоа значи дека брои само клетки што содржат броеви, игнорирајќи празни клетки, текст и други несоодветни вредности.

Форматот на функцијата е:

=COUNT (опсег1, опсег2, ...)

каде што аргумент може да биде еден или повеќе опсези во кои ќе се бројат само бројчени вредности.

Пример: Во продавница се води табела за дневна продажба на различни производи. Во колона А се внесени имињата на производите, а во колона В количината. Некои полиња се празни или имаат текст ако производот не бил продаден тој ден. Колку различни производи биле продадени тој ден?



Слика 13 *Пример табела за дневна продажба на различни производи*

Одговор: =COUNT(B2:B7). Продадени се 4 различни производи.

Забелешка: За разлика од функцијата COUNT, која брои само клетки со бројчени вредности, COUNTA ги зема предвид и клетките со текст, датуми, логички вредности и други содржини.

Функција COUNTIF

Функцијата COUNTIF брои колку пати се исполнува одреден услов во даден опсег на клетки. Тоа значи дека брои колку клетки имаат конкретен број, колку клетки имаат одреден текст или колку клетки се поголеми или помали од одредена вредност.

Форматот на функцијата е:

=COUNTIF(опсег, услов)

каде што опсег е делот од табелата каде што бараш податоци, а услов е критериумот што треба да го исполнуваат клетките за да бидат бројчени.

Пример:

- =COUNTIF (A1:A10, “>50”) – брои само клетки што имаат вредност поголема од 50 во опсегот A1:A10.

- =COUNTIF(B1:B20,"Завршено") – брои колку пати се појавува зборот «Завршено» во B1:B20.
- =COUNTIF(C1:C15, ">=10") – брои клетки со вредност еднаква или поголема во опсег C1:C15.
- =COUNTIF(D1:D30, " ") – брои празни клетки во опсегот D1:D30.

Пример: Во колоната овошје колку пати се појавува зборот „Јаболко".

	A
1	Овошје
2	Јаболко
3	Банана
4	Јаболко
5	Лимон
6	

Слика 14 *Пример за колона „овошје"*

Одговор: = COUNTIF(A2:A5, "Јаболко"). За примерот даден на слика 14, одговорот е 2 пати.

. . .

Функција SUMIF

Функцијата SUMIF ги собира (сумира) вредностите од клетките што се во одреден опсег и го исполнуваат зададениот услов. Тоа значи дека функцијата не ги сумира сите клетки, туку само оние што се совпаѓаат со критериумот што го задаваш.

Форматот на функцијата е:

=SUMIF(опсег, услов, опсег_за_сума])

каде опсег е опсегот на клетки каде што се проверува условот, услов е критериумот кој треба да го исполнува секоја клетка од опсегот, додека опсег_за_сума (опционално) е опсегот од каде што ќе се соберат вредностите. Ако не се наведе, се сумираат клетките од првиот опсег.

Пример:

- =SUMIF(A1:A10, ">50") – ги собира само броевите поголеми од 50 во A1:A10.
- = SUMIF(A1:A10, 50) → ги сумира броевите кои се еднакви на 50.
- = SUMIF(B1:B10, "Продадено") → сумира ако вредноста е "Продадено".

Пример: Од зададена табела (слика 15) сумирај го бројот на продадени ранци.

	A	B
1	**Производ**	**Продадена количина**
2	Ранец	5
3	Шатор	2
4	Ранец	3
5	Компас	4
6		
7	Продадени ранци	

Слика 15 *Пример за табела за сумирање на продажба за одреден производ*

Функција IF

Функцијата IF овозможува логичка проверка на услов и враќа различни вредности во зависност од тоа дали е исполнет условот (TRUE) или не (FALSE). Тоа значи дека функцијата проверува дали нешто е вистина или не, по што враќа една вредност ако е вистина и друга вредност ако не е.

Форматот на функцијата е:

=IF(логички_услов, вредност_ако_вистинито, вредност_ако_невистинито)

каде логички_услов е израз кој го проверуваме, вредност_ако_вистинито — резултат ако условот е исполнет, вредност_ако_невистинито — резултат ако условот не е исполнет.

Пример:

- =IF(A1>50, "голем број", "мал број") – ако вредноста која е во клетката A1 е поголема од 50 како резултат ќе се прикаже „голем број"; во спротивно, ќе се прикаже „мал број".
- =IF(A1>0, "Позитивен", "Негативен") – ако вредноста во клетката е поголема од нула како резултат ќе се прикаже текстот „позитивен"; во спротивно, ќе се прикаже „негативен".

Пример: На слика 16 е прикажана табела со производи и нивната моментална количина во магацин. Во колона „Порака" треба да пишува „Во залиха" ако има 1 или повеќе парчиња или „Нарачај" ако има 0 парчиња. Која функција ќе ја искористиш?

	A	B	C
1	Производ	Количина	Порака
2	Шатор	5	
3	Батериска лампа	0	
4	Ранец	2	
5	Вреќа за спиење	4	
6	Компас	12	

Слика 16 *Табела со производи и нивната моментална количина во магацин.*

C2 =IF(B2>0, "Во залиха", "Нарачај

	A	B	C	D	E	F
1	Производ	Количина	Порака			
2	Шатор	5	Во залиха			
3	Батериска лампа	0	Нарачај			
4	Ранец	2	Во залиха			
5	Вреќа за спиење	4	Во залиха			
6	Компас	12	Во залиха			

Слика 17 *Решение на задачата*

Одговор: =IF(B2>0, «Во залиха», «Нарачај»)

ЗАКЛУЧОК

Кога користиш функција во програма за табеларно пресметување, прво го наведуваш името на пресметката што сакаш да ја направиш (на пример: SUM, IF), а потоа, во загради, ги внесуваш аргументите — односно што точно треба да се пресмета. Аргументите ѝ даваат на функцијата флексибилност да работи со различни видови податоци. Во формулите и функциите, операторите играат клучна улога ▸▸▸

▸▸▸

— тие се користат за пресметки (на пример: +, -, *, /), споредби (на пример: =, >, <) и манипулација со текст (на пример: & за спојување).

Кога користиш функција, прво пишуваш кое пресметување сакаш да го направиш (преку името), а потоа што точно да пресмета (преку аргументи во загради). Аргументите им даваат на функциите флексибилност за работа со различни податоци.

Операторите играат клучна улога во пресметките, споредбите и управување со податоците.

Опсегот на клетки претставува група поврзани клетки и се користи во формули и функции за да се анализираат или да се пресметаат податоци од повеќе клетки одеднаш.

Кога работиш со формули, важно е да ја разликуваш релативната адреса (на пример: B2), која се менува при копирање, и апсолутната адреса (на пример: $B$2), која останува иста дури и ако формулата се копира на друго место. Ова ти овозможува поголема контрола врз пресметките.

Напредните функции како COUNT, COUNTIF, SUMIF и IF, кои овозможуваат броење, сумирање и логичка проверка според зададени услови, ги прави особено корисни за анализа на податоци.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Која е разликата помеѓу формула и функција во програма за табеларни пресметувања?

2. Како изгледа синтаксата (форматот) на функциите во програма за табеларни пресметувања?

3. Што е аргумент во функција и какви видови аргументи може да има една функција?

4. Дадена е формулата =AVERAGE(A1:A10). Што претставува A1:A10 во оваа формула?

5. Што е оператор и кои видови постојат?

6. Кои симболи се користат за дефинирање на континуиран и разделен опсег?

7. Како би се означил опсег од клетки од A1 до D10?

8. Објасни ги поимите апсолутна и релативна адреса! Кога треба да користиш апсолутни адреси, а кога релативни?

9. Што ќе се случи ако формулата =A1*2 (од B1) ја копираш во B2?

10. Како мешана адреса ($A1 или A$1) се разликува од апсолутна и релативна?

11. Наведи примери за користење на функциите COUNT, COUNTIF, SUMIF и IF!

ПРАКТИЧНИ ВЕЖБИ

1. Направи табела „Производи“ со колона производи и колона основна цена! Секој производ добива фиксна доплата од 20 денари за испорака, која се наоѓа во посебна клетка.

	A	B	C	D
1	**Производ**	**Основна цена**	**Доплата за испорака**	**Крајна цена**
2	Јаболко	50	20	
3	Банана	40		
4	Портокал	60		
5				

Слика 18 *Табела „Производи“*

2. Во колоната D пресметај ја крајната цена со додавање на доплатата! Искористи релативна адреса за основната цена (за да се менува при копирање) и апсолутна адреса за доплатата од C1 (за да не се менува при копирање)!

3. Во една табела внеси различни вредности (броеви, текст, празни клетки)! Со помош на функција определи колку бројчени вредности има.

4. Во табела се внесени имињата на учениците и нивните статуси за денешниот час по Информатика („Присутен“ или „Отсутен“). Изброј колку ученици се присутни на часот!

5. Креирај ја дадената табела и пресметај колку нарачки направил Марко и колкава сума потрошил на нарачки!

	A	B
1	**Клиент**	**Сума нарачка**
2	Марко	1000
3	Ана	850
4	Марко	750
5	Иван	600

Слика 19 *Табела „Нарачки“*

6. Направи табела со колони „Име на ученик“ (најмалку 10 ученици) и „оценка по Информатика“! Со помош на функција изброј колку ученици имаат оценка 5 по предметот Информатика!

7. Направи табела во која за една недела ќе евидентираш колку часа дневно учиш по секој предмет (колона А – предмети, колона Б – колку часа учиш)! Пресметај ја вкупната сума на часови посветени на Математика!

8. Во табела се впишани бројот на извршени задачи од вработените. Потребно е да се прикаже „Да“ ако вработениот има извршено 10 или повеќе задачи, и „Не“ во спротивно.

9. Во табела внеси вредности за температура. Ако е под 0 °C, до колоната „температура“ треба да пишува «Студено», ако е над 30 °C, «Жешко», инаку «Умерено».

5.3 Напредна работа со графикони

Работните листови во програмите за табеларни пресметки може да содржат многу податоци кои се честопати тешки за разбирање или објаснување. Работите може да станат многу појасни кога податоците визуелно ќе ги претставиш со помош на графикон. Сè што треба да направиш е да ги избереш податоците и да избереш вид графикон. Има различни типови графикони (столбест, линиски, кружен...), при што твој избор треба да биде оној што најефективно ќе ги претстави податоците. Ако не е јасен графиконот, тогаш тој нема да биде многу корисен.

ЗАДАЧИ ЗА ПОВТОРУВАЊЕ:

Број на посетители во парк во текот на неделата

Внеси ги податоците за посетеност:

Ден	Број на посетители
Понеделник	120
Вторник	150
Среда	200
Четврток	180
Петок	220

Креирај столбест дијаграм и анализирај кога има најмногу посетители.

Едно од можните решенија на задачата за повторување е графиконот прикажан на слика 20.

Слика 20 *Графикон за задачата „Број на посетители во парк во текот на неделата“*

5.3.1 Елементи на графикон

За да го разбереш графиконот, важно е да ги препознаеш неговите главни елементи – како што се насловот, оските, легендата и самите податоци што се прикажани визуелно.

1. Графиконот обично има две оски:

- **X-оска** (хоризонтална) на која се прикажани категории, датуми, денови итн.
- **Y-оска** (вертикална) на која се прикажани вредностите (броеви, проценти, мерки итн.).

1. **Наслов на графикон** (Chart Title) кој треба да даде јасен опис на податоците во графиконот.

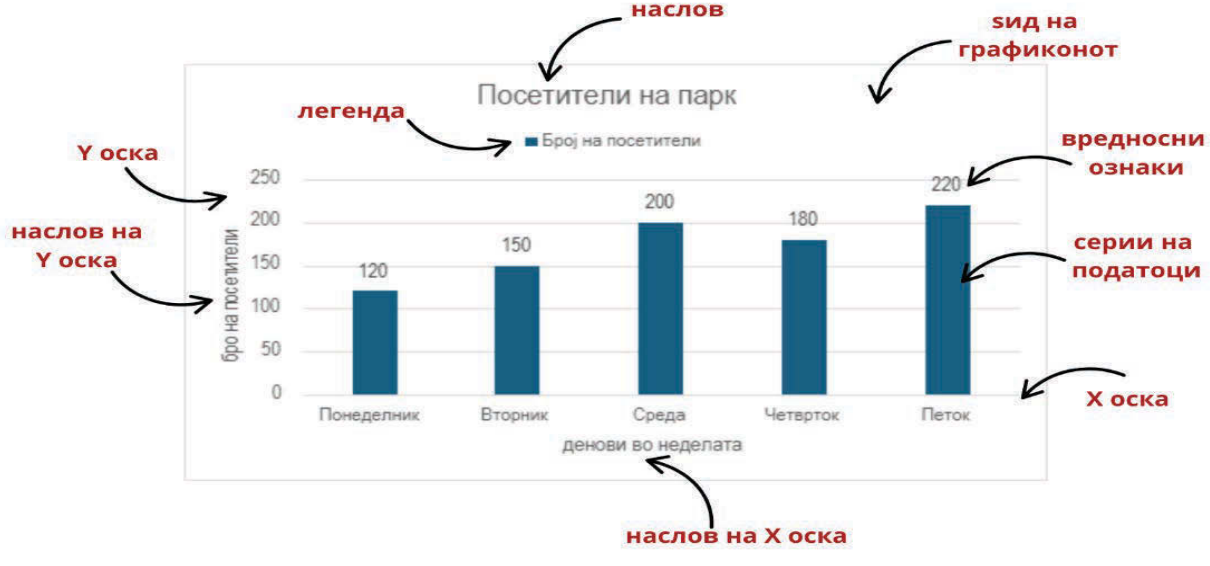
2. **Наслов на оски** (Axis Labels) кој е текст што ги означува категориите или вредностите на оските.

3. **Легенда (Legend)** чија задача е да објасни кои бои или знаци во графиконот претставуваат одредени податоци.

4. **Серии на податоци (Data Series)** кои се всушност група поврзани податоци прикажани како колони, линии, точки и слично.

5. **Мрежни линии (Gridlines)** или помошни линии кои може да бидат хоризонтални или вертикални, што го олеснуваат читањето на вредностите.

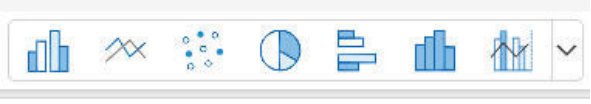
6. **Вредносни ознаки (Data Labels)**, броеви што стојат над или во самите елементи (столбови, линии) и ја покажуваат точната вредност.



Слика 21 *Елементи на графикон*

5.3.2 Дополнително уредување на графикон

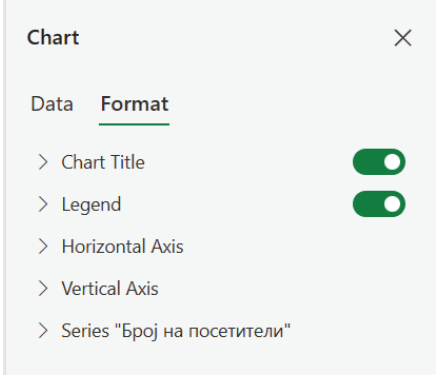
Откако ќе го направиш графиконот, можеш да се предомислиш и да заклучиш дека некој друг вид графикон подобро ги прикажува податоците. Во таков случај, можеш лесно да го промениш стилот или типот на графиконот без да ги внесуваш податоците повторно. Сè што треба да направиш е да кликнеш на графиконот, од менито да го избереш Chart, а потоа да одбереш од листата понудени типови.



Слика 22 *Алатки за избор за тип на графикон*

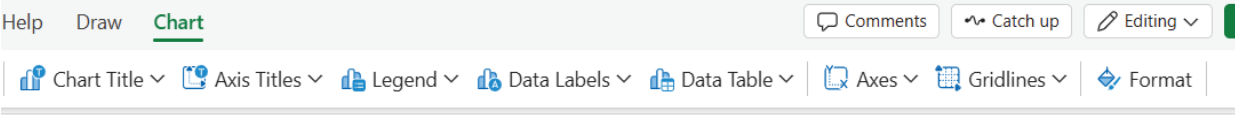
Наместо готов стил, секој дел од графиконот (столб, линија, оска, мрежна линија, легенда итн.) можеш поединечно да го уредиш. Во Excel 365 тоа можеш да го направиш на два начина:

- Двоен клик на елементот – се отвора панел за форматирање, каде што можеш да ги менуваш стилот, бојата, големината и други параметри.



Слика 23 *Алатки за уредување елементи на графикон*

2. Користење на менито „Chart“.

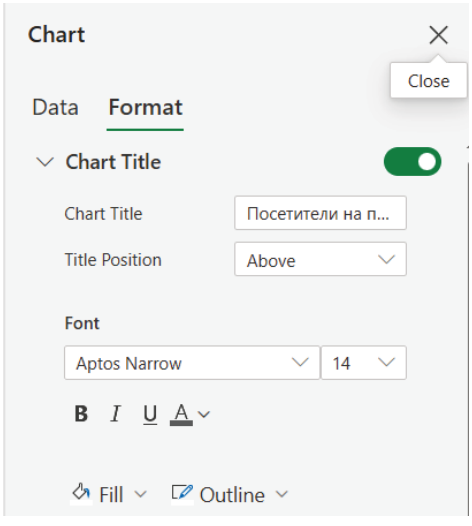


Слика 24 *Алатки за уредување – мени Chart*

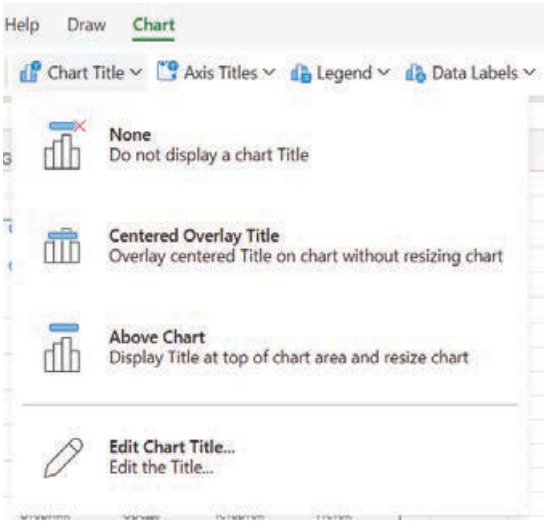
Уредување наслов на графикон

3. Двапати кликни на насловот и внеси нов текст. Во Format > Chart Title, можеш да промениш тип на фонт, големина, боја, порамнување (слика 25).

4. Ако користиш Chart > Chart Title, можеш да го поставиш на различни позиции (слика 26).



Слика 25 *Уредување наслов*



Слика 26 *Позиционирање на насловот*

Уредување на оските

- Двапати кликни на X-оската или на Y-оската за да се отвори опцијата за форматирање на оските.
- Можеш да го менуваш опсегот на вредности, името на оските, скалата...
- Во Chart > Axes, можеш да ги вклучиш или да ги исклучиш оските.

Уредување на легендата

- Преку Chart > Format > Legend легендата можеш да ја вклучиш/исклучиш, да ја преместиш, форматираш или да смениш фонт.
- Преку Chart > Legend , можеш да ја поставиш лево, десно, горе или долу.

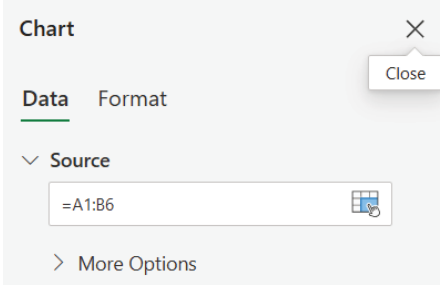
Прилагодување на сериите на податоци:

- Двапати кликни на столбовите/линиите > Chart > Format > Data Series и можеш да менуваш бои, стил и дебелина.

Повторно избирање на извор на податоци

Ако сакаш да го смениш опсегот на податоци на графиконот:

- Кликни на графиконот, оди во Chart > Select Data > Source



Слика 27 *Позиционирање на насловот*

Уредување на вредносните ознаки

- Двапати кликни на вредносните ознаки > Chart > Format > Data Labels и можеш да менуваш формат на бројот, фонт, содржина на ознаките.
- Во Chart > Data Labels можеш да одбереш дали да се прикажат вредносните ознаки на податоците.

ЗАКЛУЧОК

Графиконите се моќна алатка за визуелно прикажување на податоци, со што се овозможува полесно разбирање и анализа на информации. Програмите за табеларни пресметки овозможуваат и напредно уредување на графиконите –менување на нивниот вид, уредување на насловот, оските, легендата и сериите на податоци, како и прилагодување на вредносните ознаки. Дополнително, со повторен избор на извор на податоци, графиконите може да се ажурираат без потреба од ново креирање.

ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Што претставува графикон?

2. Што е важно при избор на вид графикон?

3. Кои се основните елементи на еден графикон?

4. Што е улогата на X-оската, а што на Y-оската?

5. Што претставува легендата на графиконот?

6. Што се вредносни ознаки (data labels)?

7. Како можеш да го смениш типот на веќе креиран графикон?

8. На кој начин може да се уреди насловот на графиконот?

9. Како можеш да ја преместиш или да ја форматираш легендата?

10. Објасни како можеш повторно да избереш извор на податоци за графикон!

11. Зошто е важно графиконот да биде визуелно јасен и читлив?

12. Како би одлучил/а кој тип графикон најдобро ги прикажува дадените податоци?

13. Доколку вредностите не се прикажани на столбовите, дали графиконот ќе биде разбирлив? Објасни!

ПРАКТИЧНА ВЕЖБА:

Следните податоци прикажуваат температури измерени во текот на една недела:

Ден	Температура (°C)
Понеделник	12
Вторник	15
Среда	17
Четврток	14
Петок	13
Сабота	16
Недела	18

1. Внеси ги податоците во табела.

2. Креирај столбест графикон.

3. Додај наслов на графиконот.

4. Додај наслови на оските („Денови“, „Температура“).

5. Прикажи ги вредносните ознаки (data labels) на врвот од секој столб.

6. Промени ја бојата на столбовите – столбот кој ја прикажува максималната температура обој го со црвена боја, за минималната температура со зелена боја, а останатите со сина боја.

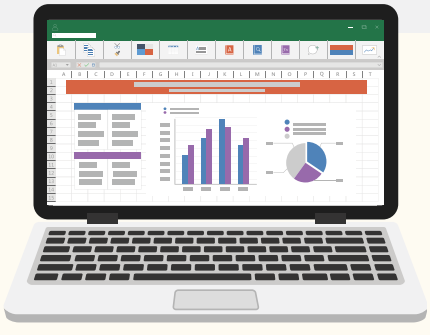
7. Додај уште еден ред податоци и прошири го графиконот.

8. Сними го документот.

9. Промени го типот на графикон во кружен (pie chart).

10. Прикажи ги процентите во ознаките на деловите од кругот.

11. Сподели заклучок: Кој тип графикон појасно ги прикажува податоците?



5.4 Сортирање

Сортирањето податоци значи нивно подредување по утврден редослед. Станува збор за една од најосновните и најкорисни функции. Сортирањето може да се изврши по азбучен редослед, нумерички (од најмалиот до најголемиот или од најголемиот до најмалиот број), хронолошки (од најстариот до најновиот и од најновиот до најстариот датум/време) по редослед дефиниран од страна на корисникот или по формат, вклучувајќи боја на клетка, боја на фонт или икони. Основна причина за сортирањето на податоците е да се овозможи подобро разбирање, организација и пронаоѓање на податоците – сортираните податоци многу побрзо и полесно се пребаруваат.

На што треба да внимаваш: При сортирање треба да ја избереш целата табела – ако сортираш само една колона без остатокот од табелата, редовите ќе се измешаат, што води до погрешни податоци (на пр., оценка на едно дете ќе се појави кај друго). Избегнувај празни редици/колони или споени клетки, затоа што може да предизвикаат проблеми при сортирањето. Препорака е да ставиш и наслови на колоните.

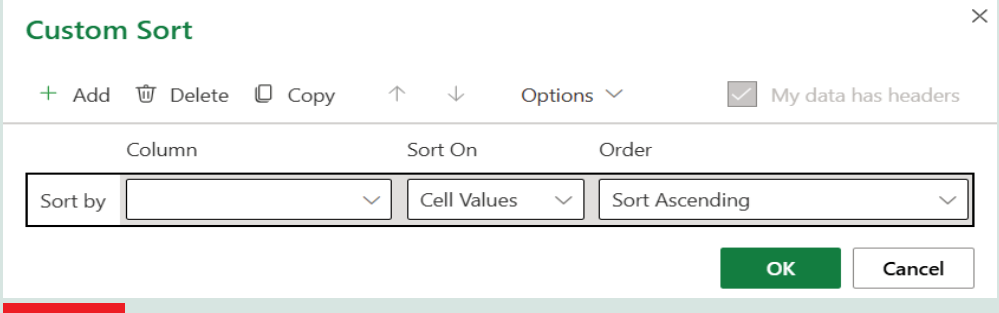
ДА СЕ ПОТСЕТИМЕ:

Сортирањето се активира со избор на Home > Sort & Filter. Притоа:

- Текстот можеш да го сортираш по азбучен редослед (Sort A to Z) или по обратен азбучен редослед (Sort Largest to Smallest).
- Броевите можеш да ги сортираш од најмалиот кон најголемиот (Sort Smallest to Largest) или од најголемиот кон најмалиот број (Sort Largest to Smallest).
- Датумот и времето можеш да ги сортираш хронолошки – од најстар кон најнов (Sort Oldest to Newest) или од најнов кон најстар (Sort Newest to Oldest).

...

Изборот на Custom Sort (прилагодено сортирање) ти овозможува да сортираш податоци по повеќе колони истовремено, или според специјални критериуми — како што се боја на клетка, боја на фонт, икона ако имаш поставено условно форматирање. Ова е особено корисно кога стандардното сортирање по азбучен или бројчен ред не е доволно.

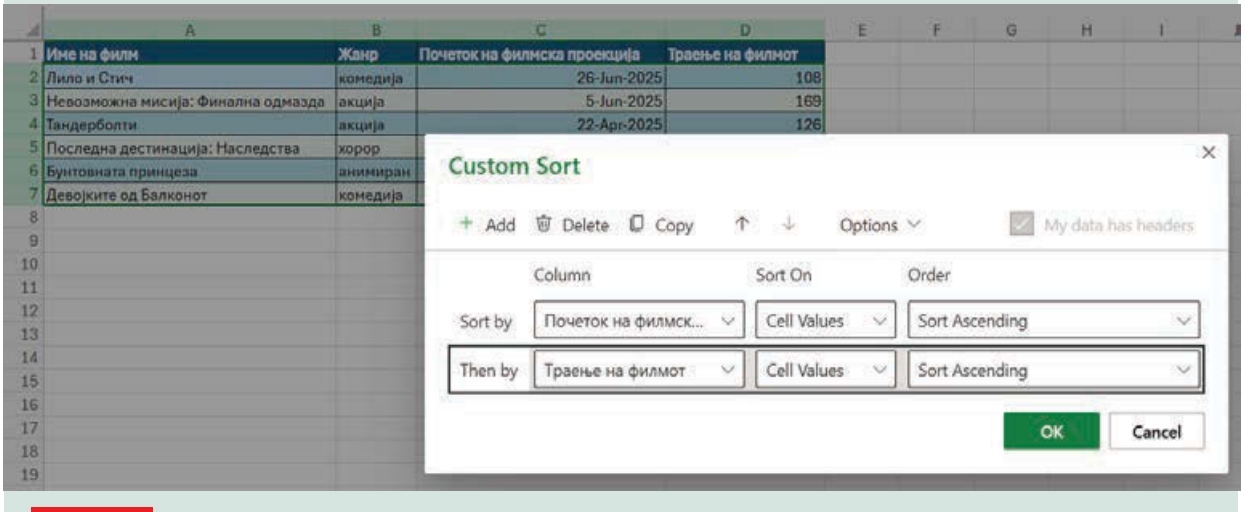


Слика 28 *Прозорец Custom Sort*

Во прозорецот Custom Sort треба да избереш:

- колона по која ќе сортираш (во областа Column),
- критериум по кој ќе сортираш (Cell Values – според содржината на клетката, Cell Color – според бојата на клетката, Font Color – според бојата на фонт или Conditional Formatting Icon – според иконата доколку си применил условно форматирање) и
- редослед по кој ќе сортираш (растечки/опаѓачки) по што се одбира OK.

Доколку сакаш да додадеш повеќе критериуми, тоа го правиш со клик на Add, кој се наоѓа во горниот лев агол од дијалог прозорецот Custom Sort. Во новиот критериум Then by на ист начин се одредува како да се изврши сортирањето. Не заборавај дека првиот наведен критериум има приоритет. За да се изврши сортирањето, на крај треба да ги потврдиш дотерувањата со избор на OK.



Слика 29 *Пример на сортирање по два критериума*

За да избришеш некој од критериумите, избери Delete.

Не заборавај да го одбереш (да го селектираш) полето My data has headers доколку имаш наслови на колоните за да спречиш мешање со останатите редови.

ЗАКЛУЧОК

Сортирањето податоци претставува една од најважните и најкорисни функции за ефикасно управување со информации. Без разлика дали станува збор за едноставно сортирање по една колона или за напредно прилагодено сортирање по повеќе критериуми, оваа алатка овозможува корисниците брзо да ги организираат и да ги анализираат податоците на логичен и разбирлив начин.

ПРАШАЊА И ЗАДАЧИ ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Што претставува сортирањето и зошто е важно?

2. Што се случува ако се сортираат податоците без да бидат избрани сите поврзани колони?

3. Што претставува Custom Sort во Excel? Објасни со свои зборови!

4. Како можеш да сортираш податоци по повеќе колони истовремено? Објасни ги чекорите!

5. Зошто е важно да имаш наслови (headers) во табелата пред да сортираш?

6. Кои се некои од најчестите проблеми што може да се појават при сортирање и како да се решат?

7. Објасни со свој пример кога и зошто би користел/а сортирање во секојдневна ситуација (на пр., распоред на часови, листа со оценки, итн.)!

ПРАКТИЧНА ВЕЖБА:

1. Направи табела со следните колони: Име на ученик, Презиме на ученик, Датум на раѓање, Време на пишување (време за кое ученикот ќе го напише зборот „информатика“ во програма за текст). Сортирај ја табелата по Време на пишување, а потоа по Датум на раѓање.

2. Замисли дека твоето училиште организира е-спорт турнир во кој тимови од ученици се натпреваруваат во популарни видеоигри. Секој играч има освоено одреден број поени според нивната изведба на тренинг натпреварите, а натпреварите се распоредени во различни денови од неделата.

По секоја од наведените активности сортираната табела ископирај ја во нов работен лист:

- Сортирај ги играчите според бројот на поени, од најмногу кон најмалку.
- Сортирај најпрво по Тим, а потоа по Поени (сортирање по две колони).
- Сортирај ја табелата по Ден за натпревар, така што деновите ќе бидат подредени по следниот редослед: Понеделник, Вторник, Среда, Четврток, Петок.
- Кој играч има најмногу поени од Тим Б?

	A	B	C	D
1	**Име на играч**	**Поени**	**Ден за натпревар**	**Тим**
2	Ана	78	Петок	Тим Б
3	Јован	85	Понеделник	Тим А
4	Кристина	85	Среда	Тим А
5	Александар	65	Петок	Тим Б
6	Елена	90	Вторник	Тим А
7	Стефан	72	Четврток	Тим Б

Слика 30 *Е-спорт турнир*

3. Во денешно време, YouTube претставува една од најпопуларните платформи за забава, едукација и креативно изразување, особено меѓу младите. Следната табела прикажува некои од познатите канали на YouTube. Во табелата се вклучени информации за името на каналот, неговата категорија, бројот на претплатници (во милиони) и датумот на започнување со работа на платформата.

	A	B	C	D
1	**Канал**	**Категорија**	**Претплатници (милиони)**	**Датум на започнување**
2	T-Series	Музика	266	13.03.2006
3	Dude Perfect	Спорт/Забава	61	16.03.2009
4	TED-Ed	Образование	22	22.03.2012
5	Lozano Music	Музика	0.29	08.02.2012
6	MKD Education	Едукација	0.04	25.10.2020
7	Slatkaristika Off.	Музика	0.31	20.09.2013
8	Fitnes so Aleks	Здравје/Фитнес	0.065	03.07.2019
9	Kids Learn MK	Едукација	0.08	18.03.2019

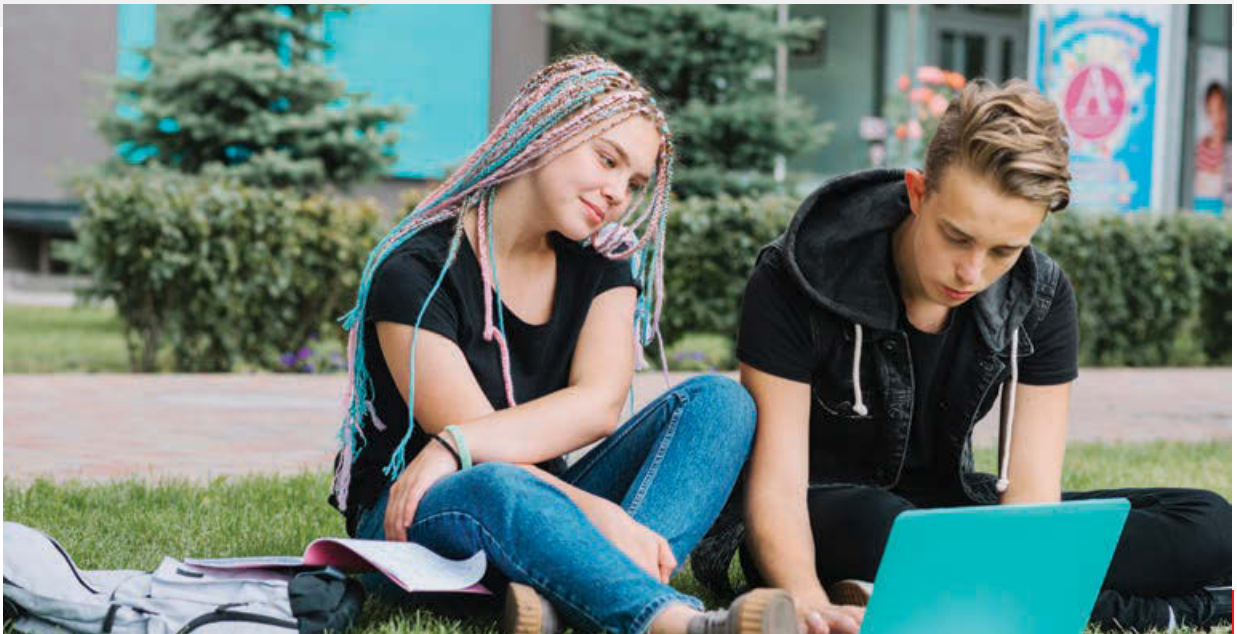
Слика 31 *Познати YouTube канали*

По секоја од наведените активности сортираната табела ископирај ја во нов работен лист:

1. Сортирај по колона „Претплатници (милиони)“ – од најмногу кон најмалку.

2. Сортирај по „Претплатници (милиони)“ – во растечки редослед.

3. Сортирај по „Датум на започнување“ – од најстар до најнов канал, а потоа по „Категорија“.



5.5 Филтрирање

Филтрирањето податоци во база овозможува да се прикажат (да се „пропуштат“) само оние податоци што задоволуваат одредени критериуми, додека останатите податоци ги кријат (не ги отстрануваат). Со користењето филтри може да се едитира, форматира и печати поттабела без преуредување. Главната намена на филтрирањето е да се олесни работата со податоци, да се пронајдат специфични информации и да се анализираат податоците на поедноставен начин, без да се менуваат или бришат оригиналните податоци. Во програмите за пресметување, може да се филтрираат различни типови податоци: текст (на пример, имиња, градови, категории), броеви (на пример, продажни суми, количини, проценти), датуми (на пример, датуми на нарачки, датуми на раѓање), логички вредности (TRUE/FALSE).

ДА СЕ ПОТСЕТИМЕ:

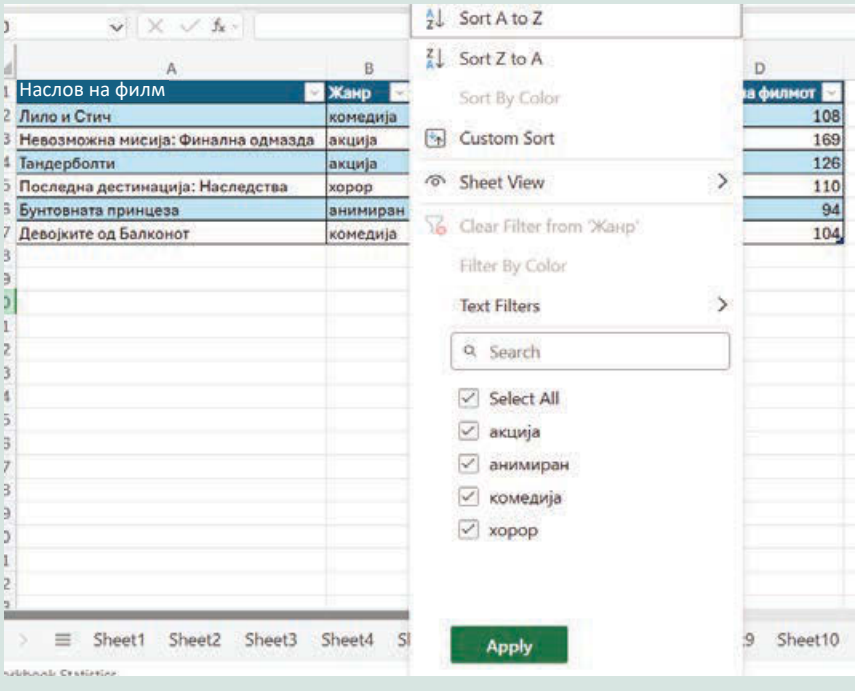
Филтрирање податоци со точно определени вредности

Чекори

1. Се кликнува на податоците што сакаме да се филтрираат.

2. Се избира Home → Sort & Filter → Filter.

3. Во првиот ред од табелата во секоја колона се појавува стрелка. Се кликнува на стрелката по што паѓа листа во која се прикажани сите податоци што ги има во колоната, а пред нив поле за потврда.



Слика 32 *Филтрирање податоци со точно определени вредности*

4. Во полето за потврда треба да останат избрани (обележани) само потребните податоци (со клик на полето за потврда се избираат податоците).

5. Се потврдува со избор на Apply.

Филтрирање податоци со поставување дополнителен услов

Чекори:

1. Се кликнува на податоците што сакаме да се филтрираат.

2. Се избира Home → Sort & Filter → Filter.

3. Во првиот ред од табелата во секоја колона се појавува стрелка. Се кликнува на стрелката по што паѓа листа од која се избира филтер за текст/број/датум (Text Filters/Number Filters/Date Filters).

4. Од дополнителната листа се избира опција според што ќе се филтрира (на пример: за текст – започнува со...., за број – помало од....., за датум – пред...).

5. Се отвора дијалог прозорец во кој се довршува поставувањето на условот.

6. Се кликнува на ОК.

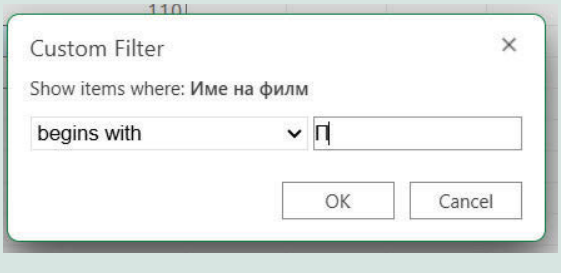
7. Со копчињата And (И) и Or (ИЛИ) се креираат сложени услови.

Пример:

Филтрирање податоци во табела: Дадена е табела со податоци: Наслов на филм, жанр, датум на проекција, траење на филмот. Прикажи ги имињата на филмовите што започнуваат на буквата „П“.

	A	B	C	D
1	Наслов на филм	Жанр	Почеток на филмска проекција	Траење на филмот
2	Тандерболти	акција	22-Apr-2025	126
3	Невозможна мисија: Финална одмазда	акција	5-Jun-2025	169
4	Последна дестинација: Наследства	хорор	6-Jun-2025	110
5	Бунтовната принцеза	анимиран	16-Jun-2025	94
6	Лило и Стич	комедија	26-Jun-2025	108
7	Девојките од балконот	комедија	27-Jun-2025	104

Слика 33 *Табела за преглед на филмови*



Слика 34 *Прозорец во кој се поставени барањата за филтер*

	A	B	C	D
1	Наслов на филм	Жанр	Почеток на филмска проекција	Траење на филмот
5	Последна дестинација: Наследства	хорор	6-Jun-2025	110
8				

Слика 35 *Филтрирана табела*

Отстранување филтер

1. Се кликнува на стрелката во колоната во која е поставен филтерот.

2. Од листата се избира Clear Filter From „име на колоната".

3. Се прикажуваат сите податоци.

Отстранување на сите филтри во работниот лист

4. Се избира Home > Sort & Filter > Clear.

ЗАКЛУЧОК

Со помош на филтри, корисниците можат да ја намалат сложеноста на големи табели и да се фокусираат само на релевантните информации. Ова ја подобрува ефикасноста, ја олеснува работата и ја зголемува јасноста при презентирањето и донесувањето одлуки. Сепак, при користење филтри треба да се внимава да не се изгубат важни податоци и да се осигури дека филтрираните податоци ја претставуваат целокупната слика.

ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Што претставува филтрирањето во Excel?

2. Како знаеме дека е применет филтер на колона?

3. Кои типови податоци може да се филтрираат во Excel? (наведи барем 3)?

4. Што се случува со податоците што не ги исполнуваат условите на филтерот – дали се бришат?

5. Како можеш да филтрираш само редови каде што вредноста е поголема од 5 000?

6. Како можеш да ги отстраниш сите филтри што се применети во табела?

7. Дали можеш да филтрираш по повеќе колони истовремено? Објасни како се прави тоа!

8. Кога ќе примениш филтер по текст, кои се опциите што можеш да ги користиш (на пример: „почнува со“, „содржи“...)?

9. Во табела со над 100 редови, имаш колона „Оценка“ (од 1 до 5). Како ќе филтрираш само редови со оценка 4 и 5?

10. Имаш колона „Цена“. Како ќе прикажеш само продукти со цена помеѓу 1 000 и 3 000 денари?

11. Кои проблеми може да се појават ако имаш празни редови или клетки кога применуваш филтер?

12. Како можеш да комбинираш филтер со сортирање за подобра анализа на податоците?

13. Зошто е важно да ги поставиш насловите на колоните пред да користиш филтер?

ПРАКТИЧНА ВЕЖБА:

1. Креирај табела „Потрошувачки на тинејџери за омилени продукти“ со име на колони Име, Категорија, Производ, Цена (ден), Купено месец и Оценка (1–5) и пополни ја според сликата.

	A	B	C	D	E	F
1	**Име**	**Категорија**	**Производ**	**Цена**	**Купено месец**	**Оценка (1–5)**
2	Мила	Слушалки	JBL Tune 510BT	3,200.00 ден	Јануари	5
3	Алекс	Гејминг опрема	Razer Mouse Viper	4,500.00 ден	Февруари	4
4	Сара	Смарт уред	Xiaomi Smart Band 8	2,800.00 ден	Јануари	5
5	Мелек	Гејминг опрема	Xbox контролер	4,200.00 ден	Март	3
6	Хана	Слушалки	Sony WH-CH510	3,900.00 ден	Февруари	4
7	Осман	Смарт уред	Apple AirTag	1,800.00 ден	Март	5
8	Ана	Гејминг опрема	RGB Gaming Keyboard	5,500.00 ден	Јануари	5
9						

Слика 36 *Табела за потрошувачката на тинејџери за омилени продукти*

По секоја од наведените активности филтрираната табела ископирај ја во нов работен лист:

1) Филтрирај ги учениците што купиле слушалки.

2) Прикажи ги само производите со оценка 5.

3) Филтрирај ги производите купени во февруари.

4) Прикажи ги сите продукти од категоријата „Гејминг опрема“.

5) Филтрирај ги производите што чинат повеќе од 4 000 денари.

6) Дополни ја табелата со свои податоци, примени различни филтри по име, категорија, месец или оценка и направи кратка анализа на навиките при купување.

2. Креирај табела „Активности на тинејџери на социјални медиуми“ со име на колони Име, Платформа, Тип на активност, Време дневно (мин), Омилена содржина и пополни ја според сликата.

	A	B	C	D	E
1	**Име**	**Платформа**	**Тип на активност**	**Време дневно (мин)**	**Омилена содржина**
2	Ива	Instagram	Објавување приказна	90	Мода
3	Марко	YouTube	Гледање видеа	120	Гејминг
4	Елена	TikTok	Креирање видеа	80	Танц
5	Марија	TikTok	Скролување	60	Комедија
6	Стефан	Instagram	Лајкување и коментари	45	Фитнес
7	Ана	YouTube	Гледање видеа	100	Наука
8	Емин	facebook	Лајкување и коментари	30	Мода
9	Ајше	Instagram	Објавување приказна	80	Гејминг

Слика 37 *Табела за активности на тинејџери на социјални медиуми*

По секоја од наведените активности филтрираната табела ископирај ја во нов работен лист:

1) Дополни ја табелата со свои податоци.

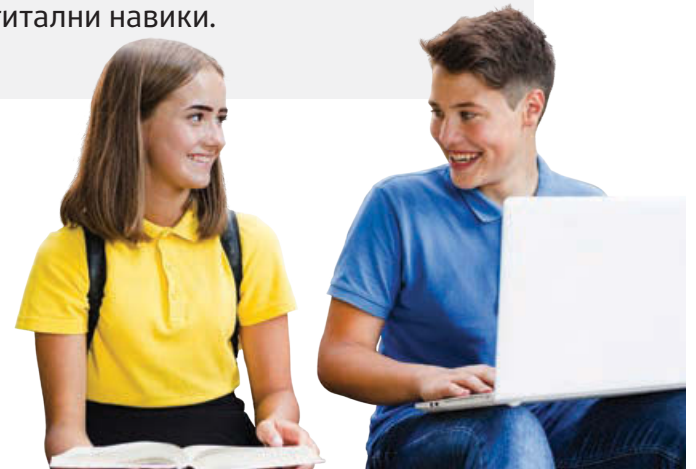
2) Филтрирај ги учениците што користат TikTok.

3) Прикажи ги само оние што поминуваат помеѓу 50 и 85 минути дневно на платформата.

4) Прикажи ги само оние што поминуваат повеќе од 100 минути дневно на платформата.

5) Филтрирај по платформа „Instagram“ и активност „Објавување приказна“.

6) Искористи ја оваа вежба за дискусија за дигитални навики.



5.6 Креирање извештаи – пивот-табела

Пивот-табела е напредна алатка која овозможува брзо сумирање, групирање и анализа на големи количини податоци. Наместо рачно броење или пресметување, пивот-табелата овозможува со неколку кликнувања да добиеш статистика – колку производи се продале, во кој месец имало најмногу приходи, колку гласови добил секој одговор во анкета, итн.

Важно: Најпрво треба да ги подготвиш податоците – во табелите не смее да има празни полиња и секоја колона треба да има име.

	A	B	C	D	E	F	G	H	I
1	ID	Производ	Категорија	Количина	Единична цена	Вкупна продажба	Месец	Град	Вработен
2	1	Маичка	Облека	15	800	12000	Јануари	Скопје	Марија Стојанова
3	2	Патики	Облека	10	2500	25000	Февруари	Битола	Александар Јанев
4	3	Телефон	Електроника	5	25000	125000	Јануари	Скопје	Иван Петров
5	4	Лаптоп	Електроника	2	45000	90000	Март	Охрид	Елена Наумовска
6	5	Чанта	Додаток	7	3000	21000	Февруари	Битола	Марија Стојанова
7	6	Слушалки	Електроника	12	1500	18000	Април	Тетово	Бојан Крстев
8	7	Јакна	Облека	8	4000	32000	Март	Куманово	Александар Јанев
9	8	Ракавици	Додаток	3	900	2700	Март	Скопје	Велес
10	9	Ранец	Додаток	6	2200	13200	Април	Скопје	Елена Наумовска
11	10	Смарт часовник	Електроника	3	7000	21000	Февруари	Тетово	Иван Петров
12	11	Очила	Додаток	4	2800	11200	Јануари	Битола	Марија Стојанова
13	12	Маичка	Облека	10	800	8000	Април	Куманово	Бојан Крстев
14	13	Патики	Облека	9	2500	22500	Март	Скопје	Александар Јанев
15	14	Телефон	Електроника	6	24000	144000	Април	Битола	Елена Наумовска
16	15	Лаптоп	Електроника	3	45000	135000	Јануари	Охрид	Иван Петров
17	16	Слушалки	Електроника	10	1500	15000	Февруари	Скопје	Бојан Крстев

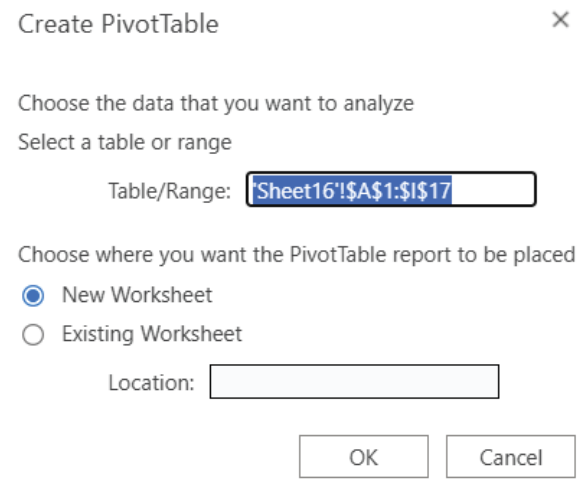
Слика 38 *Табела „Продажба“*

Чекори:

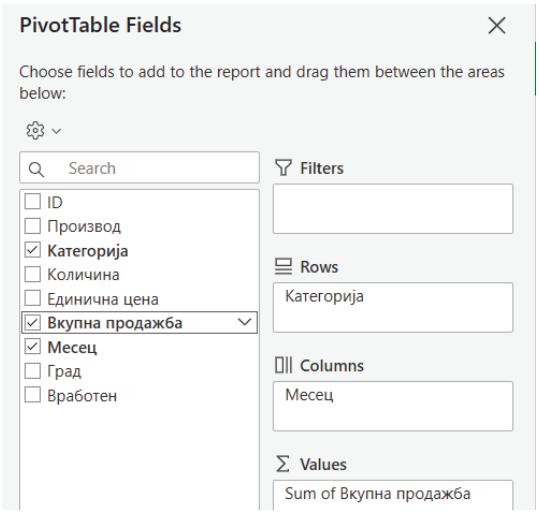
1. Се кликанува каде било во табелата.

2. Се избира Insert > PivotTable.

3. Се отвора прозорецот PivotTable каде што се избира опсегот на податоци (Table/Range), а потоа и локација за новата табела (New worksheet – нов работен лист (препорака), Existing Worksheet – постоечки работен лист). Се потврдува изборот со клик на OK.



Слика 39 *Create PivotTable*



Слика 40 *Полиња во пивот-табелата од Табела „Продажба“*

4. Пивот-табелата сè уште е празна. Од десната страна се прикажани имињата на колоните и панел во кој може да се влечат полиња во четири зони:

5. Rows (Редови): Категории по кои сакаш да групираш (на пример, категорија).

6. Columns (Колони): За разгранување по друга вредност (на пример, месец).

7. Values (Вредности): Што сакаш да пресметаш (сума, просек, број...).

8. Filters (Филтри): За филтрирање на табелата според одредени критериуми.

За овој пример, доколку крајниот резултат треба да биде пивот-табела за вкупна продажба по категории и месеци, десниот панел ќе изгледа како на следната слика:

Добиената пивот-табела го има следниот облик:

3	Sum of Вкупна продажба	Месец				
4	Категорија	Април	Јануари	Март	Февруари	Grand Total
5	Додаток	13200	11200	2700	21000	48100
6	Електроника	162000	260000	90000	36000	548000
7	Облека	8000	12000	54500	25000	99500
8	**Grand Total**	**183200**	**283200**	**147200**	**82000**	**695600**

Слика 41 *Пивот-табела од табела „Продажба“*

За да ја ажурираш (освежиш) пивот-табелата по промена на податоците во изворната табела, еден од начините е: кликни со десен клик каде било во пивот-табелата и од прикажаното мени избери Refresh (Освежи), по што табелата автоматски ќе ги преземе најновите податоци од изворот.

ЗАКЛУЧОК

Употребата на пивот-табели е од суштинско значење во процесот на анализа на податоци, особено поради нивната флексибилност, брзина и способност да обработуваат големи количини информации без потреба од напредни формули. Сепак, нивната ефикасност зависи од квалитетот на почетните податоци и од способноста на корисникот да ги постави вистинските параметри. Се препорачува секоја анализа со пивот-табела да започне со добро подготвени и чисти податоци, користење смислени категории за редови и колони, како и јасна цел за анализа.

Пивот-табелите значително ја подобруваат способноста за донесување одлуки во бизнис-контексти преку претставување информации на концизен и аналитички начин. Без разлика дали се користат во финансиска анализа, следење трошоци, анализа на задоволството на клиентите или оперативно планирање, тие придонесуваат за информирано одлучување базирано на податоци.

ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. За што најчесто се користат пивот-табелите?

2. Кои типови податоци се погодни за анализа со пивот-табели?

3. Зошто е важно податоците да имаат наслови на колоните пред да се креира пивот-табела?

4. Кои функции за пресметки може да се користат во пивот-табела (на пр. сума, просек)?

5. Што ќе се случи со пивот-табелата ако се променат податоците во изворната табела?

6. Како се ажурира (refresh) пивот-табела по промена на податоците?

7. Кои се некои од најчестите грешки при работа со пивот-табели?

ПРАКТИЧНА ВЕЖБА:

- Во твоето училиште се изведуваат многу активности – секој месец се случуваат работилници, натпревари, спортски денови и креативни проекти. Некои ученици се натпреваруваат, други учат нови вештини, а трети помагаат во организацијата. Наставничкиот совет сака подобро да ја разбере вклученоста на учениците и ефектот од овие активности. Добиваш задача да направиш анализа на настаните со помош на пивот-табела во Excel.

Изворната табела што ќе ја користиш, ги содржи следниве колони:

- Име на ученик
- Одделение
- Тип на настан – Работилница, натпревар, спортски ден, презентација итн.
- Датум на одржување – Кога се случил настанот
- Времетраење (во минути) – Колку време траела активноста

Со пивот-табелата, треба да се одговори на прашања како:

а) Кој тип настани е најчест?
б) Кои одделенија се најактивни?
в) Колку просечно траат настаните?
г) Кои ученици имаат најмногу учества?

Предлог за изворна табела:

	A	B	C	D	E
1	**Име на ученик**	**Одделение**	**Тип на настан**	**Датум на одржување**	**Времетраење (во минути)**
2	Ана Петрова	VIII-1	Натпревар	12.03.2024	90
3	Јован Стојков	IX-2	Работилница	15.03.2024	120
4	Ева Милошевска	VIII-2	Спортски ден	20.03.2024	60
5	Петар Јованов	IX-2	Работилница	25.03.2024	90
6	Ана Петрова	VIII-1	Натпревар	05.04.2024	95
7	Ева Милошевска	VIII-2	Презентација	10.04.2024	45
8	Јован Стојков	IX-2	Натпревар	12.04.2024	100

Слика 42 *Табела „Ученички активности“*

Решенија за самопроверка

а) **Rows:** Тип на настан
Values: Count of Тип на настан
б) **Rows:** Одделение
Values: Count of Име на ученик
в) **Rows:** Тип на настан (или Одделение)
Values: Average of Времетраење (во минути)
г) **Rows:** Име на ученик
Values: Count of Тип на настан

ЗАБЕЛЕШКА:

За да ја смениш функцијата во полето Values (на пример: Sum, Count, Average) кликни на стрелката десно од нејзиното име, избери Value Field Settings... и од тука избери ја саканата функција.

5.7 Заштита на податоци

Во реалната работа со табели често се внесуваат важни податоци – било да се тоа оценки, финансиски извештаи, распореди или планови. Но, што ако некој по грешка избрише формула? Или ако некој внесе невалидна вредност, како текст наместо број? За да се спречат ваквите грешки, постојат алатки за заштита и контрола на податоците.

Во Excel 365, функцијата Protect се користи за заштита на листови, клетки или на целокупниот документ од неовластени измени.

Заштита на целиот лист

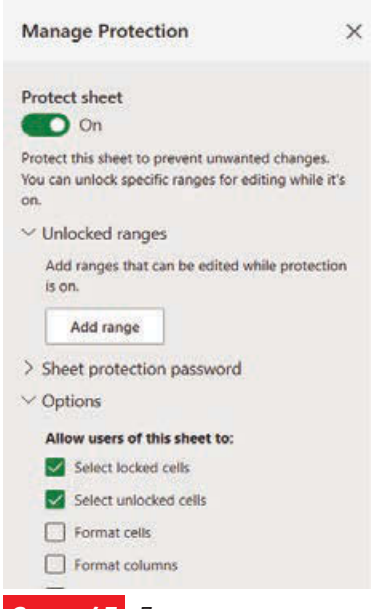
Чекори:

1. Избери Review.

2. Кликни на Protection -> Manage Protection.

3. Внеси лозинка (опционално).

4. Избери што може да менуваат корисниците.
(на пример форматирање, селектирање на клетки).



Слика 43 *Дел од паното Manage Protection*

Важно: Кога Excel документот ќе се извезе (експортира) во друг формат, како PDF, CSV или HTML, заштитата што си ја поставил во Excel, не се пренесува. Ако сакаш документот да има заштита, треба да го зачуваш документот како Excel (.xlsx). File → Save a Copy ако документот е на OneDrive.

Забелешка: Доколку работиш во онлајн верзијата на Excel 365 опциите за заштита се малку поразлични од десктоп верзијата. Тука можеш да заштитиш цел лист или можеш да ограничиш уредување преку Manage Protection – има опција Add range, каде што можеш да дефинираш кои делови од листот може да се менуваат додека е листот заклучен.

Во Excel сите клетки се „заклучени“, но тоа нема ефект додека не се вклучи заштита на листот. Затоа, ако сакаш само некои клетки да се заклучат, прво треба да ги „отклучиш“ сите, па потоа да ги „заклучиш“ само оние што ти се потребни.

Отклучување на сите клетки

Чекори:

1. Селектирај ги сите клетки (Ctrl+A).

2. Десен клик → Format Cells → Protection таб.

3. Исклучи на опцијата Locked.

Заклучување на клетки

1. Селектирај ги клетките што сакаш да не може да се менуваат.
2. Десен клик → Format Cells → Protection таб.
3. Одбери го полето Locked. (обележи го).
4. Оди на Review → Protect Sheet и активирај заштита.

Сега сите клетки што ги означи како „Locked" не може да се менуваат, додека останатите се слободни за уредување!

Забелешка: Доколку сакаш да сокриеш формули и функции, постапката е иста со заклучување на клетки, но наместо Locked треба да одбереш Hide.

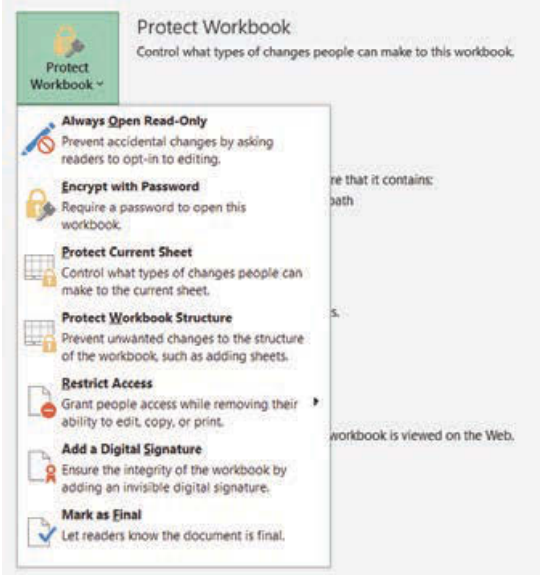
Заштита на цела работна книга

Чекори:

1. Оди на File → Info.

2. Кликни на Protect Workbook → Encrypt with Password.

3. Внеси лозинка и потврди.



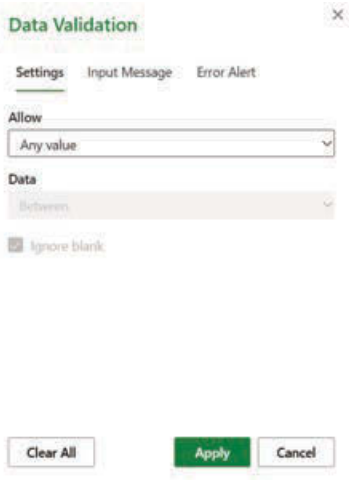
Слика 44 *Прозорец protect workbook*

Валидација на податоци

Замисли дека креираш табела во Excel во која учениците треба да внесат оцени од 1 до 5. Но, еден ученик по грешка внел 7, а друг внел текст наместо број. Таквите грешки можат да доведат до погрешни резултати во пресметките. Токму тука валидацијата на податоци доаѓа како решение.

Валидација на податоци овозможува да се постават правила за тоа кои вредности смее да се внесат во одредени клетки. На пример, можеш да дозволиш само броеви помеѓу 1 и 10, само датуми по денешниот датум, или да направиш паѓачка листа со однапред дефинирани вредности. Со ова се избегнуваат грешки, се контролира точноста на податоците и се олеснува обработката на информациите.

Валидацијата на податоци се активира со избор на Data > Data validation.



Слика 45 *Прозорец Data validation*

Во Allow одредуваш каков тип на податок е дозволено да се внесе во избраната клетка или опсег од клетки. Дозволени вредности се:

Опција	Објаснување
Any value	Дозволува што било (нема ограничување).
Whole number	Дозволува само цели броеви (без децимали).
Decimal	Дозволува броеви со децимали.
List	Дозволува избор од однапред дефинирана листа.
Date	Дозволува внес само на датуми.
Time	Дозволува внес само на време.
Text length	Ограничува колку карактери може да се внесат.
Custom	Дозволува поставување на сопствено правило со формула.

Поставување валидација (пример со цел број)

Чекори:

1. Означи ги клетките каде што сакаш да се внесе цел број помеѓу 1 и 5.

2. Оди во Data > Data Validation.

3. Под „Allow" избери Whole number.

4. Во прозорецот Data validation во Data одбери between, во Minimum напиши 1, а во полето Maximum напиши 5.

5. Потврди го изборот со клик на Apply.

Сега, доколку во овие клетки се обидеш да внесеш број различен од дефинираниот опсег (од 1 до 5), ќе се појави порака за грешка.

Слика 46 *Прозорец Data validation за цел број*

Поставување на валидација (пример со листа, со директно внесување вредности)

Замисли дека креираш табела во која учениците треба да одберат еден од неколку однапред дефинирани одговори, како што се „Да", „Не" или „Можеби". Наместо секој да пишува по свое и да се случат грешки при внесувањето, можеш однапред да им понудиш избор од паѓачко мени.

Чекори:

1. Избери ја клетката или опсегот каде што сакаш да додадеш паѓачка листа.

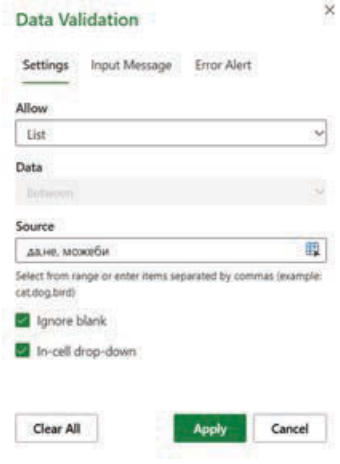
2. Избери Data > Data Validation.

3. Во прозорчето што ќе се отвори:

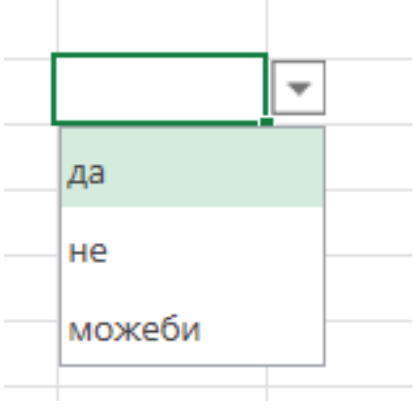
- Во полето Allow (Дозволи) избери: List (Листа).
- Во полето Source (Извор), директно внеси ги опциите, разделени со запирки.

4. Потврди го изборот (Apply).

Сега во селектираното поле (или опсег) на десниот крај од клетките ќе се појави стрелка. Со клик на неа се отвора паѓачка листа од која треба да избереш една од понудените опции.



Слика 47 *Прозорец Data validation за паѓачка листа*



Слика 48 *Клетка со паѓачко мени*

ЗАБЕЛЕШКА:

Во прозорецот data Validation има уште две јазичиња:

- Input Message служи за пишување информациска порака што се појавува **кога корисникот ќе кликне на клетка** во која има поставена валидација на податоци.
- Error Alert каде што може да напишеш каква порака сакаш да се појави **кога некој ќе внесе вредност што не е дозволена** според правилата на валидацијата.

ЗАКЛУЧОК

Валидацијата на податоци е практична и моќна алатка која овозможува контрола на внесот на податоци во табелите. Со неа може да се ограничи што може да се внесе во одредени клетки – на пример, само броеви, датуми, текстови или вредности од паѓачка листа. Ова е особено важно кога се работи со поголеми табели или кога се споделуваат документи со други лица, бидејќи помага да се избегнат грешки и да се зачува точноста на податоците.

ПРАШАЊА ЗА ПРОВЕРКА НА ЗНАЕЊЕТО

1. Што значи заклучување на клетки во Excel и дали тие веднаш стануваат заштитени?

2. Кој е првиот чекор што треба да го направиш ако сакаш да дозволиш уредување само на некои клетки?

3. Кој е вториот чекор по заклучување на клетките за да се активира заштитата?

4. Како се заштитува еден работен лист во Excel 365?

5. Што е разликата меѓу „заштита на лист“ и „заштита на работна книга“?

6. Кои опции можеш да ги овозможиш или оневозможиш при заштита на лист?

7. Може ли да им се дозволи на други корисници да внесуваат податоци само во одредени клетки?

8. Дали е задолжително да се постави лозинка при заштита на лист или од работна книга?

9. Што се случува ако се обидеш да уредуваш заклучена клетка во заштитен лист?

10. Како се отстранува (исклучува) заштитата од лист или од работна книга?

11. Што претставува валидација на податоци?

12. Која е целта на користење валидација на податоци?

13. Кои се можностите во паѓачкото мени „Allow“ во прозорецот за валидација?

14. Што треба да содржи полето „Source“ при внесување листа на вредности?

15. Што се случува ако некој внесе вредност што не е дозволена според валидацијата?

16. Дали можеш да ја примениш истата валидација на повеќе клетки одеднаш? Како?

ПРАКТИЧНА ЗАДАЧА

1. Имаш Excel табела со следниве податоци:

Име	Презиме	Оценка 1	Оценка 2	Вкупно
Марко	Јанев	5	4	9
Елвир	Баку	4	5	9
Ивица	Стојанов	3	4	7

- Заклучи ги клетките со исклучок на колоните „Оценка 1“ и „Оценка 2“!
- **2.** Во табелата со производи пополни ги соодветно празните полиња користејќи формули кои ќе ги скриеш!

Производ	Цена по парче	Количина	Вкупно
Леб	40	2	
Млеко	60	3	
Сок	90	1	
Шеќер	50	2	
ВКУПНО			

3. Во колона B (клетки B2 до B10), направи паѓачка листа со следниве опции: I година, II година, III година, IV година!

4. Во колона C внеси возраст на ученици! Дозволи внес само на цели броеви од 14 до 19.

5. Во колона D внеси датум на упис. Дозволи внес само на датуми од 01.09.2024 до 30.09.2025.

ПОВТОРУВАЊЕ ЗА ТЕМА 5

1. Наведи три основни формули или функции што се користат во Excel за пресметување податоци (на пример: SUM, AVERAGE, IF)!

2. Објасни што значи поимот „релативно адресирање“ и како се разликува од „апсолутно адресирање“ во Excel!

3. Креирај табела со имиња на ученици и нивните оценки! Примени формула за пресметка на просек за секој ученик!

4. Во истата табела, додај условно форматирање за да ги обележиш со црвена боја сите оценки под 2,50!

5. Имаш база на податоци со колони: Име, Предмет, Оценка, Датум. Филтрирај ги само учениците со оценка ≥ 4 и објасни како ги доби резултатите!

6. Анализирај кој тип графикон (линиски, колони или пита-дијаграм) е најсоодветен за да се прикаже промената на оценките во текот на месецот. Образложи го изборот!

7. Дадена е табела со трошоци и приходи. Процени дали функцијата IF е најдобриот избор за проверка и дали е буџетот позитивен или негативен! Објасни ја твојата одлука!

8. Имаш табела со важни финансиски податоци. Одлучи кој тип на заштита е најпотребен: заклучување на клетки, заштита на лист или заштита на цела работна книга! Образложи!

9. Креирај пивот-табела од база на податоци со најмалку 20 записи (на пример: продажба, оценки, трошоци)! Организирај ја така што ќе прикажува вкупни суми по категории!

10. Направи сопствен извештај кој содржи:

- табела како база на податоци,
- пресметки со минимум 3 напредни функции (IF, VLOOKUP/XLOOKUP, COUNTIF),
- графикон по избор,
- филтри за брзо пребарување,
- пивот-табела,
- основна заштита на најважните клетки.