Тодор Димов

ИНФОРМАТИКА ЗА II (ДРУГУ) ГОДИНУ ГИМНАЗИЈСКОГ ОБРАЗОВАЊА

Одлуком о одобравању и употреби уџбеника за наставни предмет Информатика за II (други) разред средњег гимназијског образовања број 26-843/1 од 22. 7. 2026. године, који је донела Национална комисија за уџбенике.

Скопље, 2026.

Информатика

II (другу) годину гимназијског образовања

Назив оригинала:
Информатика
за II (втора) година гимназиско образование

Издавач:
ДПТУ „СТУДЕНТСКИ СЕРВИС“ ДОО – Скопље

Аутор:
Тодор Димов

Рецензенти:
проф. др. Марија Календар
проф. др. Горјан Јакимовски
Жаклина Милошева

Уредник:
Симона Ристовска
Лектор:
Нена Ристић Костовска
Превод са македонског на српски језик:
Нена Ристић Костовска
Дизајн и техничка припрема:
ДПТУ „СТУДЕНТСКИ СЕРВИС“ ДОО – Скопље

Штампа:
ДПТУ „СТУДЕНТСКИ СЕРВИС“ ДОО
ул. „Перо Наков“ бр. 112 – Скопље

Тираж:
50 примерака

CIP - Каталогизација во публикација
Национална и универзитетска библиотека "Св. Климент Охридски", Скопје

004(075.3)=163.41

ДИМОВ, Тодор
Информатика за II (другу) годину гимназијског образовања / Тодор Димов ; [превод са македонског на српски језик Нена Ристић Костовска]. - Скопје : Студентски сервис, 2026. - 330 стр. : илустр. ; 28 см

Превод на делото: Информатика за II (втора) година гимназиско образование

ISBN 978-608-4777-53-3

COBISS.MK-ID 69472261

САДРЖАЈ

Живимо у времену у којем је технологија неодвојиви део свакодневног живота. Компјутери, интернет, мобилни уређаји и дигиталне услуге омогућавају нам да брже комуницирамо, учимо, радимо и стварамо. Иза свих ових технологија стоје знања и вештине из области информатике. Информатика данас не подразумева само коришћење рачунара. Она обухвата разумевање начина на који функционишу савремене технологије, обраду и организацију података, развој програма, стварање дигиталних садржаја и решавање проблема помоћу компјутерских система.

Овај уџбеник је израђен у складу са наставним програмом Информатике за други разред гимназијског образовања и има за циљ да помогне ученицима да стекну савремена знања и практичне вештине потребне за успешно учешће у дигиталном друштву. Садржаји уџбеника обухватају савремене технологије у информатици, програмирање, базе података, мултимедију, компјутерску графику и веб-развој. Посебна пажња посвећена је практичној примени знања путем примера, задатака и активности који подстичу ученике да размишљају, истражују и стварају сопствена решења.

Циљ уџбеника није само да пружи нове информације већ и да развија логичко размишљање, креативност, дигиталну писменост и спремност за целоживотно учење. У времену када се технологија непрестано мења, најважнија постаје способност учења, прилагођавања и примене знања у новим ситуацијама.

Надамо се да ће овај уџбеник бити користан водич у откривању света информатике и да ће мотивисати ученике да наставе да истражују, стварају и развијају своје идеје.

Информатика није само предмет који се изучава у учионици. Она је језик савременог света и алатка за стварање будућности.

TEMA 1

1.1

САВРЕМЕНЕ ТЕХНОЛОГИЈЕ У ИНФОРМАТИЦИ И ДИГИТАЛНА ТРАНСФОРМАЦИЈА

Живимо у ери у којој се промене одвијају огромном брзином, а њихов главни покретач је информатика. Данас се појам "технологија" више не односи само на обичне рачунаре и основне програме, већ на сложен систем иновација које из темеља мењају начин на који радимо, комуницирамо и живимо.

Под појмом савремених технологија у информатици подразумевају се најновији напредни алати, хардверски уређаји, софтверска решења и мрежни системи који омогућавају брзу обраду, складиштење и анализу огромних количина података. Ове технологије представљају прелазак са некадашњег једноставног и статичког управљања базама података на активан, интелигентан екосистем. Оне омогућавају анализирање огромних количина података и доношење паметних одлука у реалном времену, повезујући дигитални са физичким светом.

Савремене технологије непрестано се развијају и интегришу у све поре људског живота, стварајући феномен назван дигитална трансформација – процес интегрисања дигиталне технологије у све области једног друштва, као што су образовање, економија, производња и пословање, чиме се суштински мења начин њиховог функционисања.

Свет информатике данас почива на неколико кључних стубова:

- Вештачка интелигенција (AI) и велики језички модели (LLM), који представљају напредне системе способне да обрађују и генеришу текст, програмски код и идеје на начин сличан човеку (на пример, модели GPT, Gemini и Claude), анализирајући милијарде података за свега неколико секунди.
- Напредна роботика, која обједињује информатику, вештачку интелигенцију и машинство. То су аутономне или полуаутономне машине способне да извршавају сложене физичке задатке, уче из окружења и реагују на промене у реалном времену.
- Интернет ствари (IoT) представља мрежу физичких уређаја (паметних телефона, кућних апарата и индустријских сензора) који су повезани са интернетом и међусобно комуницирају.
- Технологије у облаку (Cloud Computing), које омогућавају складиштење и обраду података на удаљеним серверима уместо на локалном компјутеру. (на пр. Google Drive, Netflix).

Примена ових технологија у нашем свакодневном животу веома је распрострањена. Могућности њихове примене практично су неограничене. Ове технологије најчешће се примењују у следећим областима:

- **Креативан рад и лични асистенти.** Помоћу великих језичких модела сваки човек данас има приступ својеврсном „дигиталном мозгу". Ови модели помажу при писању есеја, превођењу језика у реалном времену, стицању нових вештина и генерисању рачунарског кода.

- **Паметна аутоматизација у дому и у индустрији.** Захваљујући роботици и интернету ствари (IoT), у свакодневном животу роботске руке обављају прецизне операције, паковање и опасне физичке послове, чиме се елиминише ризик по људе.
- **Образовање и медицина**. Велики језички модели служе као лични ментори који прилагођавају објашњења нивоу знања ученика. У медицини хируршки роботи, попут система Da Vinci, омогућавају лекарима да изводе изузетно прецизне операције на даљину.
- **Електронски услуги (Е-услуги).** Плаћање рачуна једним кликом, онлајн банкарство и корисничка подршка путем паметних „чет-ботова", који раде 24/7 захваљујући ВИ.
- **Саобраћај,** у којем се користе навигациони системи који помажу при избору најпогоднијe руте.
- **Пољопривреда,** у којој се примењују паметни системи за праћење временских услова и наводњавање.

1.1.1 Предности, изазови и утицај на друштво

Као и свака велика револуција, дигитална трансформација доноси значајне користи, али и изазове које друштво треба да превазиђе.

Позитивни аспекти	Негативни аспекти
Ефикасност и продуктивност: Велики језички модели и роботи преузимају ручне и понављајуће задатке, остављајући човеку више времена за креативно размишљање.	Губитак радних места: Аутоматизација, роботика и вештачка интелигенција могу да замене одређена занимања (од фабричких радника до административних службеника), што намеће потребу за брзом преквалификацијом.
Безбедност на радном месту: Роботи могу да раде у опасним окружењима (рудницима, великим морским дубинама и нуклеарним електранама), у којима је људски живот угрожен.	Зависност и когнитивна лењост: Претерано ослањање на језичке моделе при писању и размишљању може умањити способности за критичко мишљење и креативност, нарочито код младих.
Приступ знању и инклузивност: Језичке баријере нестају захваљујући напредним преводиоцима заснованим на великим језичким моделима (LLM), чиме информације постају доступне свима.	Ширење дезинформација: Велики језички модели могу ненамерно да генеришу нетачне информације које делују уверљиво (тзв. "халуцинације") или да буду злоупотребљени у пропагандне сврхе.
Боља дијагностика и нега: Спој вештачке интелигенције и медицинске роботике доприноси дужем и квалитетнијем животу људи.	Сајбер-безбедност и етичка питања: Ко је одговоран ако робот направи грешку? Како заштитити личне податке на којима се језички модели обучавају?

Савремене технологије треба промишљено примењивати у свакодневном животу. Следећи примери описују начине на које се технологије могу корисно употребљавати.

- Адаптивно учење за доживотно образовање. Уместо класичних курсева, практичније је користити отворене образовне платформе засноване на технологији облака за преквалификацију или стицање нових вештина (језици, програмирање), које се прилагођавају индивидуалном темпу учења.

- Паметно менаџирање здравља (IoT и Телемедицина). Коришћење паметних сатова и апликација за праћење сна, пулса и физичке активности ради дељења тих података са изабраним лекаром путем дигиталних платформи (уместо личних посета ради рутинских контрола).
- Аутоматизација рутинских задатака са сарадницима са ВИ. Коришћење алата заснованих на вештачкој интелигенцији као "когнитивних асистената" за брзо састављање пословних порука електронске поште, структурирање бележака са састанака или претраживање обимних текстуалних архива, како би се ослободило време за креативно решавање проблема.
- Свесно ограничавање коришћења дигиталних уређаја. Као одговор на смањен распон пажње, боље је користити апликације које ограничавају приступ друштвеним мрежама током радног времена, ради очувања способности усредсређеног рада.

ЗАКЉУЧАК:

Савремене технологије у информатици, од роботике која покреће физички свет до великих језичких модела који преображавају интелектуални рад, представљају моћне алате. Њихов утицај на друштво зависи искључиво од начина на који их користимо. Њихов огроман потенцијал треба искористити за олакшавање живота, уз истовремено развијање критичког мишљења и свести о етичким нормама ради заштите од негативних последица.

ДА ЛИ ЗНАШ?

Да ли знаш да један паметни телефон данас има већу рачунарску моћ од рачунара који су коришћени током првих свемирских мисија на Месец?

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА

1. Како се дефинише појам "савремене технологије" у информатици?
2. Наведи три примера савремених технологија које су данас најзаступљеније у свету.
3. Објасни својим речима везу између савремених технологија и процеса дигиталне трансформације.
4. Како функционише интернет ствари (IoT) и како повезује наш свакодневни живот са мрежом?
5. Зашто велики језички модели понекад генеришу "халуцинације" (нетачне информације) и шта то значи?
6. Опиши једну ситуацију у којој ти лично или твој паметни дом користите IoT технологију током дана.
7. Упореди позитивне и негативне стране масовног увођења робота у фабрике. Ко добија, а ко губи?
8. Анализирај утицај технологија у облаку на начин на који се данас чувају и деле подаци у поређењу са прошлошћу (хард-дискови, УСБ меморије).
9. Ко треба да сноси етичку и правну одговорност ако медицински робот направи грешку током операције – програмер, инжењер или лекар? Образложи свој став.
10. Процени да ли брзи развој вештачке интелигенције води ка друштвеној изолацији

и смањењу креативности код људи или, напротив, повећава њихову ефикасност.

11. Предложи идеју за нови „паметни уређај“ (заснован на IoT-у или вештачкој интелигенцији) који би могао да реши неки стварни проблем у твојој школи или граду.
12. Напиши кратак акциони план (3–4 корака) како би једно предузеће које традиционално ради на папиру могло безбедно да спроведе дигиталну трансформацију.

ИСТРАЖИВАЧКИ И ПРАКТИЧНИ ЗАДАЦИ

1. Направи списак десет савремених технологија које користиш током једног дана. За сваку технологију напиши у којој области се примењује и како ти олакшава свакодневни живот.
2. Изабери једну јавну установу или институцију (школу, библиотеку, банку, болницу или општину) и истражи које дигиталне услуге нуди грађанима. Припреми кратак извештај.
3. Размисли како је изгледао један школски дан пре двадесет година и упореди га са данашњим. Наведи најмање пет промена које су резултат савремених технологија.
4. У групи са друговима из одељења разговарај о предностима и могућим недостацима употребе паметних телефона у свакодневном животу. Запиши закључке.
5. Истражи једну савремену технологију за коју сматраш да ће имати велики утицај на будућност. Припреми кратку презентацију у којој ћеш објаснити како функционише и где може да се примени.

ЗАКЉУЧАК:

Савремене технологије представљају савремена техничка и информатичка решења која олакшавају свакодневни живот.

Дигитална трансформација представља процес увођења дигиталних технологија у различите области друштва.

Савремене технологије примењују се у образовању, здравству, индустрији, саобраћају, трговини и комуникацији.

Примена савремених технологија доноси бројне користи, али и одређене изазове.

Одговорно и безбедно коришћење савремених технологија доприноси заштити личних података и дигиталне приватности.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Истражи како би пронашао одговор на питање: "Како одређена апликација зна шта желим да видим у следећем тренутку?" Проналажење одговора отвара врата твојој будућности у свету напредне информатике.

1.2

САЈБЕР БЕЗБЕДНОСТ И ЗАШТИТА ЛИЧНИХ ПОДАТАКА

Сајбер-безбедност представља кључну област савременог дигиталног света, у којем заштита информација постаје све важнија. У условима све веће употребе интернета и дигиталних услуга, лични подаци изложени су различитим ризицима и злоупотребама. Зато су очување приватности и обезбеђивање одговарајуће заштите података од суштинског значаја за сваког појединца и организацију.

Дигиталне претње, као што су хакерски напади, вируси и фишинг, представљају озбиљну опасност по безбедност информација. Разумевање ових изазова и примена основних начела заштите омогућавају безбедније коришћење дигиталних технологија. Тако сајбер-безбедност постаје неопходна вештина за функционисање у савременом друштву.

ВЕЋ ЗНАШ ДА...

- користиш интернет за претраживање информација;
- комуницираш преко дигиталних платформи;
- користиш различите корисничке налоге;
- делиш дигиталне садржаје;
- препознаш да лични подаци треба да буду заштићени.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Колико корисничких налога користиш током седмице?
2. Да ли си некада размишљао/ла које личне податке остављаш при коришћењу интернета?
3. Како би поступио/ла ако непознато лице затражи твоју лозинку или твоје личне податке?
4. Да ли сви подаци који се објављују на интернету треба да буду подељени са другим људима?
5. Зашто је важно да се буде пажљив при коришћењу дигиталних услуга?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш зашто је важно да се заштите лични и дигитални подаци.
- описујеш основне мере заштите (јаке лозинке, антивирус, пажљиво коришћење интернета).
- примењујеш правила за безбедно коришћење интернета у датој ситуацији (да заштитиш своје податке при коришћењу социјалних мрежа или е-поште).
- разликујеш безбедан од небезбедног онлајн понашања и анализираш могуће последице лоше заштите података.
- предлажеш начине за побољшање личне дигиталне безбедности.

Савремене информационе технологије омогућавају брзу комуникацију, приступ великом броју информација и коришћење различитих дигиталних услуга. Свакодневно се користе електронска пошта, друштвене мреже, платформе за учење, електронско банкарство и различите мобилне апликације. При њиховом коришћењу често се уносе лични подаци, као што су име и презиме, број телефона, адреса електронске поште, лозинке и друге информације. Због тога безбедно коришћење дигиталних технологија постаје све важније за сваког корисника.

Развојем савремених технологија појављују се и различити ризици који могу да утичу на безбедност података и приватност корисника. Постоје стални покушаји крађе личних информација, неовлашћеног приступа корисничким налозима и ширења штетних програма. Да би се ови ризици смањили, потребно је познавати основна правила сајбер-безбедности и примењивати мере заштите личних података приликом свакодневног коришћења дигиталних технологија.

1.2.1 ШТА ПРЕДСТАВЉА САЈБЕР БЕЗБЕДНОСТ И ЗАШТО ЈЕ ВАЖНА?

Сваки пут када се повежеш на интернет, без обзира да ли прегледаш друштвене мреже, шаљеш е-пошту или играш онлајн игре, заправо отвараш врата дигиталног света. Али, као што закључаваш дом да би се заштитио од провалника, тако треба да бринеш и о безбедности свог дигиталног простора.

Сајбер-безбедност представља скуп мера, правила и поступака који омогућава заштиту рачунарских система, мрежа и дигиталних података од различитих претњи и злоупотреба. У савременом друштву велики број активности зависи од информационих технологија и интернета. Коришћење електронске поште, друштвених мрежа, дигиталних платформи за учење, електронског банкарства и различитих мобилних апликација подразумева сталну размену информација путем дигиталних мрежа. Због тога безбедност ових система има значајну улогу у заштити корисника и њихових података.

Сајбер-безбедност се не односи само на заштиту великих компанија и институција већ и на сваког појединачног корисника. Свакодневно коришћење паметних телефона, таблета и рачунара подразумева одређени ниво одговорности при чувању личних информација. Применом одговарајућих мера могу се смањити ризици од неовлашћеног приступа подацима, крађе идентитета или злоупотребе корисничких налога. Познавање основних начела сајбер-безбедности доприноси безбеднијем коришћењу савремених технологија.

1.2.2 ЛИЧНИ ПОДАЦИ И ДИГИТАЛНА ПРИВАТНОСТ

Лични подаци представљају информације на основу којих се одређена особа може идентификовати. У ову групу спадају име и презиме, адреса становања, број телефона, адреса електронске поште, матични број, фотографије и многе друге информације које се могу повезати са одређеним корисником. У дигиталном друштву свакодневно се остављају различити лични подаци приликом коришћења интернет услуга, мобилних апликација и електронских платформи. Због тога њихова заштита представља важан део дигиталне безбедности.

Дигитална приватност представља право сваког корисника да самостално одлучује које ће информације делити и са ким ће их делити. При коришћењу интернета потребно је пажљиво бирати податке који се објављују на друштвеним мрежама или другим дигиталним платформама. Прекомерно дељење личних информација може довести до различитих злоупотреба и угрозити безбедност корисника. Одговорно коришћење дигиталних услуга доприноси бољој заштити приватности и смањењу могућности злоупотребе личних података.

1.2.3 ПРАВИЛА ЗА БЕЗБЕДНО КОРИШЋЕЊЕ ИНТЕРНЕТА

Да би био безбедан на интернету, није ти потребно напредно информатичко знање. Довољно је да примењујеш три златна правила дигиталне хигијене:

- **Креирање јаких лозинки:** Лозинка је прва линија одбране. Слабу лозинку (као 123456 или ime123) хакери могу открити за неколико секунди. Јака лозинка треба да има најмање 12 знакова, да садржи комбинацију великих и малих слова, бројева и специјалних знакова (нпр. @, #, $, %). Поред тога, никада не треба користити исту лозинку за више различитих налога.
- **Коришћење и ажурирање антивирусног софтвера:** Антивирусни програми делују као дигитални чувари који скенирају уређај, откривају сумњив софтвер (вирусе, тројанце и шпијунске програме) и блокирају га пре него што нанесе штету. Важно је редовно ажурирати овај софтвер како би могао да препозна нове претње.
- **Пажљиво коришћење интернета:** Ово подразумева свест о томе на шта кликаш. Избегавај преузимање датотека са непознатих веб-сајтова, не кликај на сумњиве огласе (који обећавају брзу зараду или бесплатне награде) и не повезуј се на јавне, незаштићене Wi Fi мреже (као што су оне у кафићима или парковима) када уносиш осетљиве податке.

1.2.4 НАЈЧЕШЋЕ ДИГИТАЛНЕ ПРЕТЊЕ И РИЗИЦИ

Развојем интернета и савремених технологија појављују се различите дигиталне претње које могу да угрозе безбедност корисника. Међу најчешћим претњама су лажне поруке и интернет преваре, штетни програми, неовлашћени приступ корисничким налозима и крађа личних података. У одређеним ситуацијама нападачи покушавају да стекну поверење корисника и наведу их да добровољно открију своје лозинке или друге осетљиве информације. Такви покушаји најчешће се спроводе путем електронске поште, порука или лажних интернет страница.

Поред директних напада, постоје и ризици који произлазе из непажљивог коришћења дигиталних технологија. Коришћење слабих лозинки, преузимање датотека из непроверених извора или отварање сумњивих порука може довести до угрожавања безбедности рачунарских система и личних података. Препознавање могућих претњи и пажљиво коришћење интернет услуга представљају важан корак у заштити дигиталног идентитета и смањењу потенцијалних ризика.

1.2.5 БЕЗБЕДНО И ОДГОВОРНО КОРИШЋЕЊЕ ИНТЕРНЕТА

Безбедно коришћење интернета подразумева примену различитих правила и навика које доприносе заштити корисника и њихових података. Једна од основних мера јесте коришћење сигурних лозинки које садрже комбинацију слова, бројева и специјалних знакова. Корисно је не употребљавати исту лозинку за више различитих услуга и редовно је мењати. Додатну заштиту пружа и вишефакторска провера идентитета, при којој је за приступ корисничком налогу потребна додатна потврда.

Одговорно коришћење интернета подразумева и пажљиво поступање приликом дељења информација и коришћења дигиталних услуга. Потребно је проверавати веродостојност интернет страница, опрезно отварати поруке електронске поште и не делити личне податке са непознатим особама. Редовно ажурирање оперативних система и програма доприноси бољој заштити од нових безбедносних претњи. Развијањем добрих дигиталних навика и критичког мишљења корисници могу значајно да побољшају своју безбедност приликом коришћења савремених информационих технологија.

Најбољи начин да се разуме сајбер-безбедност јесте кроз њену примену у свакодневним ситуацијама са којима се суочавамо.

Пример 1: Коришћење социјалних мрежа (TikTok, Instagram, Snapchat...)

Социјалне мреже су место где се лако прослеђује превише информација. За заштиту података у овој ситуацији, треба да се примене следећа правила:

- **Лични профил (Private Account):** Промени подешавања приватности тако да само прихваћени пријатељи могу да виде твоје објаве, фотографије и локацију.
- **Пажљиво са пријављивањем "check-in" (Локација):** Не прослеђуј своју тачну локацију у реалном времену. То потврђује злонамерним особама где се тачно налазиш (или да у том тренутку ниси код куће).
- **Размисли пре него што објавиш:** Свака фотографија или информација (на пример, фотографија личне карте, кућна адреса или фотографија школе) остаје на интернету заувек, чак и ако је избришеш.

Пример 2: Коришћење електронске поште (е-пошта)

Електронска пошта је најчешћа мета такозваних **фишинг (phishing) напада** – превара помоћу којих нападачи покушавају да украду твоје лозинке или податке са банковних картица, лажно се представљајући као легитимне компаније или институције (банка, друштвена мрежа, пошта). Ради заштите примењују се следећа правила:

1. **Игнорисање непознатих пошиљалаца:** Ако добијеш е-пошту са непознате адресе са које се тражи хитан одговор (на пр. "Твој профил ће бити избрисан, кликни овде да потврдиш лозинку"), одмах је обриши.
2. **Провера линкова и приложених датотека:** Никада не отварај приложене документе са наставцима као што су .exe или .zip од непознатих људи, јер могу аутоматски да инсталирају вирус на твој рачунар.

Пример 3: Укључивање двофакторне аутентикације (2FA): Ово представља додатни заштитни слој. При пријављивању на електронску пошту, поред лозинке, систем тражи и код који ти притиже као СМС на твом телефону. На тај начин, чак и да неко сазна лозинку, неће моћи да користи твој профил без твог телефона.

1.2.6 ДИГИТАЛНИ УЧЕНИК И САЈБЕР БЕЗБЕДНОСТ

Током једног обичног школског дана ученик користи велики број дигиталних услуга. Пријављује се на електронске платформе за учење, претражује информације на интернету, комуницира путем различитих апликација и користи електронску пошту. Током свих ових активности размењују се различити подаци, а за приступ услугама најчешће се користе кориснички налози и лозинке. Безбедно коришћење ових услуга доприноси заштити личних података и смањењу могућности злоупотребе.

У једној ситуацији ученик добија поруку да је освојио награду и да је потребно да на непознатој интернет страници унесе своје корисничко име, лозинку и податке о банковном рачуну. Ако се не провере пошиљалац и адреса странице, може доћи до злоупотребе корисничког налога. Пажљиво поступање, провера извора информације и избегавање дељења личних података са непознатим особама представљају једноставне, али важне мере за побољшање дигиталне безбедности.

Понашање на интернету у великој мери одређује са каквим ћеш се последицама суочити у небезбедном интернет простору. Сваки корак који направиш у дигиталном свету оставља траг. Често, због убрзаног темпа свакодневног живота, заборављамо да интернет није само игралиште већ и простор пун скривених ризика. Разлика између безбедног сурфовања и потенцијалне дигиталне катастрофе најчешће зависи од наших одлука и навика.

1.2.7 БЕЗБЕДНО НАСУПРОТ НЕБЕЗБЕДНОМ ОНЛАЈН ПОНАШАЊУ

Да би се заштитио, најпре мораш да научиш да препознајеш ризичне ситуације. Понашање на интернету уопштено се може поделити у две категорије:

Небезбедно понашање (Ризик)	Безбедно понашање (Заштита)
Коришћење отворених јавних Wi-Fi мрежа (без лозинке) за пријављивање на друштвене мреже или е-банкарство.	Коришћење мобилног интернета (4G/5G) или заштићене кућне мреже приликом приступа осетљивим налозима.
Прихватање захтева за пријатељство од непознатих лица и прослеђивање приватних детаља са њима.	Комуникација само са особама које познајеш у стварном животу и заштита приватности.
Кликање на привлачне линкове који обећавају бесплатне награде, телефоне или "наградне игре" (Phishing).	Игнорисање сумњивих порука и провера званичних веб-страница за било какве награде.
Преузимање и инсталирање пиратских игара, филмова или програма из непоузданих извора и са торрент-страница.	Преузимање софтвера искључиво са званичних веб-страница и онлајн продавница (Google Play, App Store, званични сајтови).

Последице непажљивог понашања и недовољне заштите могу бити озбиљне и дуготрајне. Ако друге особе дођу до твојих личних података (лозинки, фотографија, матичног броја), могући су следећи сценарији:

- **Крађа идентитета:** Хакери могу да искористе твоје податке и фотографије како би направили лажне профиле, вршили преваре у твоје име или уцењивали твоје пријатеље и чланове породице тражећи новац.
- **Финансијска штета:** Преко фишинг порука или небезбедног плаћања, нападачи могу да добију приступ трансакцијским рачунима или кредитним картицама твојих родитеља, и испразне рачун за неколико минута.
- **Зараза уцењивачким софтвером (Ransomware):** Уместо обичног вируса на уређају може да се инсталира софтвер који ће закључати све твоје документе, слике и пројекте, а хакери ће тражити да им платиш откуп да би ти их вратили.
- **Нарушавање емоционалног здравља и угледа:** Нарушавање приватности и јавно објављивање приватних разговора или слика (сајбер-булинг) изазивају велики стрес, стид и дугорочне последице по друштвени живот особе.

Побољшање заштите не захтева сложене техничке вештине, већ усвајање добрих дигиталних навика. Ево неколико конкретних предлога које можеш одмах да примениш:

3. 1. Активирај двофакторску аутентификацију (2FA): Ово је најважнији корак. Поред лозинке, при пријављивању на профил (Instagram, TikTok, Google итд.) увек ће бити потребан и додатни код који стиже на твој телефон. Чак и ако неко зна лозинку, неће моћи да приступи твом профилу.
4. Користи менаџер лозинки (password manager): Уместо да свуда користиш исту једноставну лозинку, што је велика грешка, користи апликацију која ће генерисати и безбедно чувати различите, сложене лозинке за сваки твој налог.
5. Редовно прави резервну копију (Backup): Једном недељно или месечно пребаци важне документе и слике на спољни хард-диск, USB меморију или у безбедан облак (Cloud). На тај

начин, чак и ако украду или се поквари уређај, подаци ће остати безбедни и моћи ће поново да се користе.

6. Редовно ажурирање система: Не одлажи ажурирање оперативног система на телефону или рачунару. Ова ажурирања често садрже "закрпе" за новооткривене безбедносне пропусте које хакери користе за нападе.

ДА ЛИ ЗНАШ?

Да ли знаш да јака лозинка треба да буде довољно дуга и да садржи комбинацију великих и малих слова, бројева и специјалних знакова? Коришћење различитих лозинки за различите дигиталне услуге значајно побољшава безбедност корисничких налога.

ИСТРАЖИ

Спроведи истраживање како би сазнао која правила безбедног коришћења интернета примењују твоја школа, другари или породица. На основу резултата направи списак од најмање десет правила која доприносе бољој заштити личних података и дигиталне приватности.

ЗАКЉУЧАК:

Сајбер безбедност не зависи од сложености рачунара, већ од нашег понашања. Коришћењем јаких лозинки, антивируса и мало опреза при сваком клику постајемо заштићени и одговорни корисници дигиталног света. Интернет-простор је безбедан онолико колико смо и ми опрезни. Развијањем свести о опасностима и применом једноставних механизама заштите, сами постајемо творци сопствене безбедности и преузимамо контролу над својим дигиталним животом.

ПИТАЊА ЗА ПОНАВЉАЊЕ ЗНАЊА

1. Зашто сајбер безбедност има значајну улогу у савременом друштву?
2. Које информације се сматрају личним подацима?
3. Како може да буде угрожена дигитална приватност корисника?
4. Који су најчешћи ризици при коришћењу интернета и дигиталних услуга?
5. На који начин добре дигиталне навике доприносе бољој заштити личних података?

ИСТРАЖИВАЧКИ И ПРАКТИЧНИ ЗАДАЦИ

1. Направи списак личних података који се најчешће користе приликом отварања корисничког налога на интернету. Објасни зашто је важно да ти подаци буду заштићени.
2. Истражи која правила примењују најпознате интернет-платформе за заштиту корисничких налога и које личне податке сакупљају и обрађују. Припреми кратак извештај.
3. Размисли и опиши три ситуације у којима непажљиво коришћење интернета може да доведе до злоупотребе личних података.

4. У групи са осталим ученицима разговарај о предностима и ризицима дељења фотографија и личних информација на друштвеним мрежама. Запиши закључке.
5. Припреми кратак водич са десет савета за безбедно и одговорно коришћење интернета који би могао да буде користан ученицима твоје школе.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Сајбер безбедност је једна од области информатике које се најбрже развијају. Велики број државних институција, компанија, банака, болница и образовних установа свакодневно примењује савремене безбедносне системе за заштиту својих података и услуга. Постоје различита занимања повезана са овом облашћу, као што су аналитичар за сајбер безбедност, стручњак за заштиту мрежа, истраживач дигиталних претњи и специјалиста за дигиталну форензику. Развојем савремених технологија непрестано расте потреба за стручњацима који ће допринети заштити дигиталног простора.

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА

Заокружи тачан одговор.

1. Сајбер безбедност односи се на:

а) израду рачунара;

б) заштиту дигиталних система и података;

в) поправку мобилних телефона;

г) производњу рачунарских игара.

2. Објасни појмове.

Појам	Објашњење
Лични подаци	_____
Дигитална приватност	_____
Сајбер безбедност	_____

3. Допуни.

Безбедна лозинка треба да буде _____.

4. Које су три кључне карактеристике које садржи једна јака и безбедна лозинка?

5. Објасни својим речима како функционише "фишинг" (phishing) напад и шта је његов главни циљ.

6. Ако добијеш е-пошту од непознатог пошиљаоца са прикаченим фајлом са наставцима .exe у коме тврди да је видеоигра бесплатна, које конкретне кораке ћеш предузети да се заштитиш?

7. Упореди коришћење јавне Wi Fi мреже у кафићу са коришћењем сопственог мобилног интернета (4G/5G). Како избор мреже утиче на безбедност твојих личних података?

8. Процени могуће последице крађе идентитета на друштвеним мрежама. Шта мислиш, да ли су за једног тинејџера последице опасније у финансијском или у емоционалном и друштвеном смислу? Образложи свој став.

9. Предложи три основна правила која треба да садржи школски „Кодекс безбедног онлајн понашања", како би се заштитили подаци ученика у твојој школи.

1.3

КРИПТОГРАФИЈА

Свакодневно шаљемо милионе порука, купујемо онлајн и пријављујемо се на различите платформе. Али, да ли си се икада запитао/запитала како ови подаци „путују" светом, а да их неко не украде или прочита? Одговор лежи у једној древној, али данас ултрамодерној науци – криптографији.

Криптографија је наука о безбедној комуникацији која омогућава заштиту информација од неовлашћеног приступа у дигиталном свету. У њеној основи налази се процес шифровања, којим се обичан и читљив текст помоћу математичких алгоритама претвара у неразумљив код. Да би порука поново постала разумљива примаоцу, користи се обрнути процес, назван дешифровање.

Криптографија је наука о безбедној комуникацији која омогућава заштиту информација од неовлашћеног приступа у дигиталном свету. У њеној основи налази се процес шифровања, којим се обичан и читљив текст помоћу математичких алгоритама претвара у неразумљив код. Да би порука поново постала разумљива примаоцу, користи се обрнути процес, назван дешифровање.

ВЕЋ ЗНАШ ДА...

- објашњаваш зашто треба да буду заштићени лични подаци;
- препознајеш безбедно и небезбедно коришћење интернета;
- објашњаваш шта је дигитална приватност;
- наводиш примере за безбедно коришћење дигиталних технологија;
- анализираш могуће ризике при размени информација.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Како би испратио/ла тајну поруку пријатељи без да могу други да ја прочитају?
2. Да ли су људи у прошлости имали потребу да штите своје информације?
3. Које информације би желео/ла да данас остану поверљиве?
4. Шта мислиш, како банке и интернет-продавнице штите податке својих корисника?
5. Да ли је могуће да само овлашћено лице може да прочита послату поруку?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- дефинишеш појам криптографије са кључним основним појмовима: шифрирање, кључ, тајна порука;
- препознајеш примену криптографије у конкретној свакодневној ситуацији;
- криптографија представља науку о заштити информација. Шифровањем се
- разликујеш заштићене од незаштићених информација;
- објасниш значење криптографије за приватност и безбедност.

Комуникација и размена информација одувек су представљале важан део свакодневног живота људи. У различитим историјским раздобљима постојала је потреба за слањем порука које је требало да остану тајне и буду разумљиве само особама којима су намењене. Војни планови, дипломатски споразуми и важне државне информације често су заштићивани различитим методама прикривања њиховог садржаја. Потреба за заштитом информација довела је до развоја различитих техника шифровања порука.

Развојем савремених технологија криптографија постаје важан део свакодневног живота. Данас се она не користи само у војним и државним системима већ и приликом коришћења банкарских услуга, електронске поште, интернет продавница и мобилних апликација. Захваљујући криптографским методама, осетљиве информације могу се безбедно размењивати и заштитити од неовлашћеног приступа.

1.3.1 ШТА ПРЕДСТАВЉА КРИПТОГРАФИЈА?

Криптографија представља научну област која се бави заштитом информација од неовлашћеног приступа. Основни циљ криптографије јесте да омогући да одређену поруку или податак могу прочитати само особе којима су намењени. У ту сврху примењују се различити поступци којима се разумљива порука претвара у облик који друге особе не могу лако да разумеју. Особа која шаље поруку и особа која је прима треба да располажу потребним информацијама како би правилно прочитале њен садржај. На тај начин смањује се могућност злоупотребе поверљивих информација.

Криптографија данас има примену у великом броју области свакодневног живота. При коришћењу електронског банкарства, интернет-продавница, мобилних апликација и различитих дигиталних услуга, лични подаци и финансијске информације заштићују се криптографским методама. Такође, електронска пошта, дигитални сертификати и бројне интернет-услуге користе различите технике за заштиту информација. Захваљујући криптографији, корисници могу безбедније да размењују податке и користе савремене дигиталне услуге. Да бисмо је разумели, морамо познавати четири кључна појма: шифровање, дешифровање, криптографски кључ и тајна порука.

1.3.2 ШИФРИРАЊЕ, ДЕШИФРИРАЊЕ И КРИПТОГРАФСКИ КЉУЧ

Шифровање представља поступак у којем се разумљива порука претвара у неразумљив запис како би се заштитила од неовлашћеног приступа. Порука која се може прочитати назива се изворна порука, а након шифровања добија се шифрована порука. Она изгледа као низ насумичних слова и знакова. Чак и ако је неко пресретне док путује интернетом, за њега је потпуно бескорисна јер не може да је прочита. На пример, реч "Здраво!" може се претворити у неразумљив код: "Ху9#z!". То се постиже помоћу сложених математичких правила (алгоритама).

Особа која прима поруку треба да располаже одговарајућим информацијама како би могла да је врати у првобитни облик. Поступак враћања поруке у њен разумљив облик назива се дешифровање.

За успешно шифровање и дешифровање користи се криптографски кључ. Кључ представља скуп правила или података који омогућавају да се порука правилно заштити и поново прочита. У једноставним системима исти кључ може да се користи и за шифровање и за дешифровање, док се у савременијим системима за ова два поступка користе различити кључеви. Без познавања одговарајућег кључа, неовлашћеним особама много је теже да разумеју шифровану поруку.

1.3.3 САВРЕМЕНА ТЕХНОЛОГИЈА У ПРАКСИ

Цезарова шифра

Један од најпознатијих примера шифровања јесте Цезарова шифра. Код овог начина заштите свако слово поруке замењује се другим словом које се налази неколико места даље у азбуци. Ако се слова померају за три места, слово А замењује се словом Г, Б словом Д, В словом Ђ и тако даље. Особа која зна правило померања може лако да прочита поруку, док онима који нису упућени она изгледа као насумичан низ слова.

Цезарова шифра данас нема практичну примену у заштити важних информација јер је њено правило једноставно и лако се може открити. Ипак, она има велики образовни значај јер на једноставан начин објашњава основне идеје криптографије. Помоћу ње лако се може разумети како функционишу шифровање, дешифровање и употреба криптографског кључа.

Пример

Претпоставимо да је померање за три слова.

Изворна порука	Шифрована порука
А	Г
Б	Д
В	Ђ
Г	Е

Ако је порука ИНТЕРНЕТ, после шифрирања добиће се нови низ слова према изабраном правилу за померање.

Процес шифрирања (слово по слово):

И (+3 места даље) = **Л**
Н (+3 места даље) = **П**
Т (+3 места даље) = **Ф**
Е (+3 места даље) = **И**
Р (+3 места даље) = **Ћ**
Н (+3 места даље) = **П**
Е (+3 места даље) = **И**
Т (+3 места даље) = Ф

Тајна порука (кључ 3) биће следећи низ слова ЛПФИЋПИФ

Савремена криптографија

Развојем рачунарских система и интернета криптографија добија знатно ширу примену него раније. Данас бројне дигиталне услуге користе савремене криптографске методе за заштиту информација које се размењују између корисника и информационих система. Приликом пријављивања на кориснички налог, слања електронске поште, коришћења интернет продавница или електронског банкарства, осетљиве информације штите се различитим криптографским поступцима. На тај начин смањује се могућност да неовлашћене особе прочитају или измене податке.

У савременој криптографији примењују се различите методе за заштиту нформација. У неким системима исти криптографски кључ користи се за шифровање и дешифровање порука, док се у другим користе два различита кључа. Избор методе зависи од потребног нивоа безбедности и области примене. Савремени криптографски алгоритми осмишљени су тако да обезбеђују висок ниво заштите и омогућавају безбедну комуникацију између корисника и информационих система.

Примена криптографије у свакодневном животу

Криптографија је саставни део многих дигиталних услуга које људи свакодневно користе. Приликом коришћења електронског банкарства, плаћања путем интернета или приступања различитим веб-сајтовима, лични и финансијски подаци штите се криптографским механизмима. Мобилне апликације за комуникацију користе криптографију како би поруке могле да прочитају само особе које учествују у разговору. Апликације WhatsApp, Viber и Signal користе такозвано end-to-end шифровање (шифровање од краја до краја). То значи да се порука шифрује на твом телефону, а дешифрује тек када стигне до телефона твог пријатеља. Чак ни компаније које пружају ове услуге, попут компаније Meta, не могу да прочитају твоје разговоре.

Такође, бројне електронске услуге користе дигиталне сертификате који омогућавају проверу идентитета корисника и интернет страница. Криптографију не користе само тајне службе – она је свеприсутна.

Примена криптографије доприноси побољшању безбедности и поверења у савремене информационе системе. Захваљујући криптографским методама, корисници могу безбедније да користе различите дигиталне услуге. Истовремено, важно је разумети да криптографија представља само један део укупне дигиталне безбедности. За постизање вишег нивоа заштите потребно је примењивати и друге мере, као што су коришћење сигурних лозинки, пажљиво дељење информација и редовно ажурирање информационих система.

Развојем савремених технологија криптографија постаје важан део свакодневног живота. Данас се она не користи само у војним и државним системима већ и приликом коришћења банкарских услуга, електронске поште, интернет продавница и мобилних апликација. Захваљујући криптографским методама, осетљиве информације могу се безбедно размењивати и заштитити од неовлашћеног приступа.

Када корисник приступи интернет продавници и унесе податке за плаћање, криптографски системи омогућавају безбедан пренос тих информација између корисника и сервера. Захваљујући томе, осетљиви подаци не шаљу се као обичан текст, већ у заштићеном облику који неовлашћене особе много теже могу да прочитају.

Како разликовати заштићене од незаштићених информација?

Као ученик, веома је важно да одмах направиш разлику да ли су информације са којима радиш безбедне или нису безбедне:

- Заштићене информације. То су криптирани подаци приступ њима имају само овлашћена лица. Најлакше ћеш их препознати по икони са катанцем у прелистувачу и протоколу HTTPS (слово S значи Secure / Безбедно). Уношење лозинки или личних података на оваквим страницама је заштићено.
- Незаштићене информације. То су јавне информације које путују као обичан текст преко протокола HTTP (без катанца). Исто тако, незаштићене су све информације које прослеђујеш на

јавне профиле на социјалним мрежама, на отвореним форумима или када користиш јавни Wi-Fi у кафићу без лозинке. Сваки онај који има мало хакерског знања може да их пресретне.

Без криптографије савремени дигитални свет престао би да функционише. Њен значај је огроман из два разлога:

- **Заштита приватности;** она нам омогућава да имамо лични простор у дигиталном свету. Пружа нам слободу да комуницирамо са пријатељима, породицом или лекарима, знајући да наше приватне мисли, фотографије и тајне остају само наше.
- **Обезбеђивање дигиталне безбедности;** она штити државне институције, болнице, школе и банке од сајбер-напада. Криптографија спречава злонамерне особе да мењају туђе оцене у електронском дневнику, краду идентитете или онемогуће рад система читаве државе.

ЗАКЉУЧАК:

Криптографија је невидљиви чувар наших дигиталних живота. Разумевање њених основа помаже нам да будемо промишљенији корисници који знају како да заштите сопствени дигитални отисак.

ДА ЛИ ЗНАШ?

Да ли знаш да је сваки пут када на интернет-страници приметиш симбол катанца поред адресе, комуникација између твог уређаја и странице заштићена савременим криптографским технологијама?

ИСТРАЖИ

Истражи у којим свакодневним активностима користиш криптографију а да тога ниси свестан/свесна. Направи листу од најмање пет примера и објасни какву улогу криптографија има у сваком од њих.

ПИТАЊА ЗА ПОНАВЉАЊЕ ЗНАЊА

1. Питања за понављање
2. Шта представља криптографија?
3. У чему је разлика између шифрирања и дешифрирања?
4. Какву улогу има криптографски кључ?
5. Зашто Цезарова шифра има образовни значај и данас?
6. У којим свакодневним дигиталним услугама се примењује криптографија?
7. Зашто људи имају потребу да штите своје информације?
8. Како се разликују шифрирање и дешифрирање?
9. Какву улогу има криптографски кључ у заштити информација?
10. Зашто Цезарова шифра данас има претежно образовно значење?
11. У којим савременим дигиталним услугама се свакодневно примењује криптографија?

ИСТРАЖИВАЧКИ И ПРАКТИЧНИ ЗАДАЦИ

1. Помоћу правила Цезарове шифре састави тајну поруку за свог друга из одељења. Затим размените поруке и покушајте да их дешифрујете.
2. Истражи у којим свакодневним активностима користиш криптографију, а да тога ниси ни свестан/свесна. Наведи најмање пет примера.
3. Упореди Цезарову шифру са савременим методама заштите информација. Наведи најмање три разлике.
4. Размисли зашто банке, интернет продавнице и мобилне апликације користе криптографију. Припреми кратак текст са својим запажањима.
5. Истражи шта је дигитални сертификат и објасни зашто је важан за безбедну комуникацију путем интернета.

ЗАПАМТИ ШТА СИ НАУЧИО

Криптографија представља науку о заштити информација.

Шифровањем се разумљива порука претвара у заштићени запис

Дешифровање омогућава поновно читање поруке.

Криптографски кључ има важну улогу у заштити информација.

Криптографија има широку примену у савременим информационим системима и свакодневним дигиталним услугама.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Савремена криптографија једна је од најважнијих области информатике и има значајну улогу у функционисању интернета. Осим за заштиту порука и финансијских података, криптографске методе примењују се при дигиталном потписивању докумената, изради дигиталних сертификата, заштити рачунарских мрежа и функционисању блокчејн (blockchain) технологије. Развојем савремених информационих система криптографија се непрестано усавршава како би обезбедила виши ниво безбедности и поверљивости информација.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

1. Заокружи тачан одговор.

Криптографија се користи за:

а) убрзање компјутера;

б) заштиту информација;

в) поправку компјутерских мрежа;

г) израду мобилних апликација.

2. Дефиниши појам криптографије, обавезно користеће три кључна појма у дефиницији: шифровање, кључ и тајна порука

3. Тачно или нетачно.

Шифровање и дешифровање представљају исти поступак.

4. Објасни појмове.

Појам	Објашњење
Шифровање	_____
Дешифровање	_____
Криптографски кључ	_____

5. Допуни.

Један од најпознатијих једноставних начина шифровања је .

6. Како криптографија практично помаже у заштити података док они путују интернетом? Објасни поступак претварања обичног текста у шифровани текст и обрнуто.

7. Наведи једну конкретну ситуацију из свог свакодневног живота (на пример, приликом коришћења апликација за комуникацију или онлајн куповине) у којој несвесно користиш криптографију и опиши шта се у том тренутку дешава са твојим подацима!

8. Направи поређење између заштићених и незаштићених информација на интернету. По којим дигиталним знацима (симболима или протоколима у веб-прегледачу) ученици могу одмах да препознају да ли веб-страница штити њихове податке?

9. Процени значај криптографије за друштво: шта би се догодило са приватношћу грађана и безбедношћу институција (као што су банке и болнице) када би криптографија изненада нестала из дигиталног света? Образложи последице.

10. Замисли да треба да објасниш ученицима млађег узраста зашто је важно да штите своје податке. Осмисли три једноставна „златна правила“ за безбедно претраживање која се заснивају на разликовању заштићених и незаштићених информација на интернету.

ПРОБЛЕМСКИ ЗАДАТАК

Замисли да треба да пошаљеш поверљиву поруку пријатељу/пријатељици преко интернета. Објасни зашто је важно да порука буде шифрована и које предности пружа криптографија.

1.4

BLOCKCHAIN ТЕХНОЛОГИЈА – ПОВЕРЉИВИ ДИГИТАЛНИ ЗАПИСИ У САВРЕМЕНОМ СВЕТУ

Блокчејн (blockchain) технологија представља савремено информационо решење које омогућава безбедно чување и праћење дигиталних записа. Захваљујући криптографским методама и начину организације података, свака промена остаје забележена и може се лако проверити. Због ових својстава, блокчејн технологија примењује се у различитим областима савременог друштва, а њена употреба непрестано се шири.

ВЕЋ ЗНАШ ДА...

- објашњаваш шта представља криптографија;
- разликујеш шифровање и дешифровање;
- објашњаваш улогу криптографског кључа;
- наводиш примери заштите информација;
- препознајеш примену криптографије у савременим информатичким системима.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Како се важне информације могу чувати тако да свака промена остане забележена?
2. Да ли је могуће да базу података користи велики број корисника без једног централног управљача?
3. Зашто је важно знати ко је и када извршио одређену промену у неком запису?
4. Да ли технологија која се користи за криптовалуте може да се примењује и у другим областима?
5. Како савремене информационе технологије могу да допринесу већем поверењу у дигиталне услуге?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш шта представља blockchain технологија;
- објашњаваш структуру блока и повезивање блокова;
- објасниш принцип децентрализације;
- наводиш примере примене blockchain технологије.

У савременом друштву свакодневно настаје велики број дигиталних записа које треба безбедно чувати и заштитити од неовлашћених измена. Банкарске трансакције, подаци о власништву, здравствени картони, уговори и различити електронски документи представљају информације које треба

да буду тачне и поуздане. Уколико се ови записи могу лако изменити или избрисати, могу настати озбиљни проблеми и неспоразуми.

1.4.1 ШТА ПРЕДСТАВЉА BLOCKCHAIN ТЕХНОЛОГИЈА?

У савременом друштву постоји потреба да се важне информације безбедно чувају и буду доступне за проверу. Приликом куповине непокретности, преноса новчаних средстава, издавања диплома или склапања уговора, неопходно је да постоји тачна евиденција о обављеним активностима. Ако се записи могу лако изменити или избрисати, могу настати велики проблеми и неспоразуми. Због тога савремене информационе технологије нуде решења која омогућавају безбедно чување и праћење дигиталних записа.

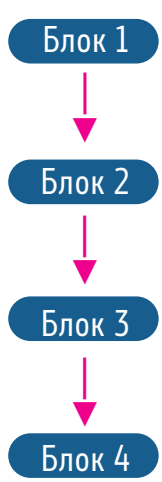
Блокчејн (blockchain) технологија представља посебан начин организовања и чувања података. Уместо да информације буду сачуване у једном централном запису, оне се организују у више међусобно повезаних блокова. Сваки нови блок садржи нове информације и повезан је са претходним блоком у ланцу. Захваљујући таквој организацији и примени криптографских метода, подаци су заштићени, а свака промена остаје забележена. Због тога блокчејн технологија представља сигуран начин чувања различитих врста дигиталних записа.

Блокови, трансакције и децентрализација

Основни елемент блокчејн (blockchain) технологије јесте блок. Блок представља дигиталну целину у којој се чувају одређене информације. У зависности од примене система, блок може да садржи податке о финансијским трансакцијама, промени власништва, закљученим уговорима или другим важним записима. Када се креира нови запис, формира се нови блок који се повезује са претходним и постаје део ланца блокова. На тај начин ствара се узастопна историја свих догађаја у систему.

Блокчејн (blockchain) технологија често се повезује са појмом децентрализације. У традиционалним системима најчешће постоји један централни управљач који чува и контролише податке. У блокчејн системима записи могу бити доступни већем броју повезаних учесника, при чему се свака нова промена бележи и може се проверити. Овакав начин организације доприноси већој транспарентности и поверењу у систем, јер историја записа остаје сачувана и може се лако пратити.

Приказ повезивања блокова



Сваки нови блок се повезује са претходним и постаје део заједничког ланца записа.

Где се чувају блокови?

Једна од најважнијих карактеристика блокчејн (blockchain) технологије јесте начин чувања података. Уместо да сви записи буду смештени на једном централном рачунару или серверу, копије блокчејн система могу се чувати на више повезаних рачунара који учествују у његовом функционисању. Ови рачунари називају се чворови (енгл. nodes) и сваки од њих садржи копију ланца блокова. Када се у систем дода нови блок, информација се бележи и ажурира код свих учесника у мрежи. Захваљујући оваквом начину организације, подаци не зависе од једног места за чување, а њихова безбедност и доступност значајно су побољшане.

Чување блокова на више повезаних рачунара доприноси већој поузданости и отпорности система. Ако један рачунар престане да ради или се појави технички проблем, остали рачунари и даље располажу сачуваним подацима. Истовремено, свака нова промена треба да буде забележена у систему, што омогућава лакшу проверу тачности записа. На тај начин блокчејн (blockchain) технологија обезбеђује сигурно чување информација и ствара поверење међу учесницима који користе мрежу.

1.4.2 BLOCKCHAIN ТЕХНОЛОГИЈА У САВРЕМЕНОМ СВЕТУ

Једна од најпознатијих примена blockchain технологије јесу криптовалуте. Криптовалуте представљају дигитална средства размене која користе blockchain за безбедно евидентирање трансакција. Свака трансакција записује се у блок и постаје део ланца записа. Захваљујући томе, може се пратити историја трансакција и смањити могућност неовлашћених измена. Bitcoin је једна од најпознатијих криптовалута, али данас постоје и многи други системи који користе ову технологију. Blockchain технологија не користи се само за криптовалуте. Она налази примену у здравству, логистици, образовању, јавној управи и при управљању различитим дигиталним записима.

Предности и могућности Blockchain технологије

Блокчејн (blockchain) технологија пружа више предности при чувању и обради дигиталних записа. Захваљујући повезаности блокова, свака промена остаје евидентирана и може се проверити. То доприноси већој транспарентности и поверењу у систем. Поред тога, примена криптографских метода обезбеђује бољу заштиту информација и смањује могућност неовлашћених измена података.

Поред бројних предности, примена блокчејн (blockchain) технологије доноси и одређене изазове. За њено успешно коришћење потребни су одговарајућа техничка инфраструктура и добра организација система који је примењују. Такође, различите области примене имају различите потребе и захтеве, па избор технологије зависи од конкретне намене. Са развојем информатике и дигиталне трансформације очекује се да ће се блокчејн технологија примењивати у све већем броју савремених информационих система и услуга.

1.4.3 САВРЕМЕНА ТЕХНОЛОГИЈА У ПРАКСИ

Blockchain технологија и катастарске евиденције

У многим државама катастарске службе воде евиденцију о власништву над земљиштем и непокретностима. Приликом куповине, продаје, наслеђивања или поклањања имовине, неопходно је да свака промена буде тачно евидентирана. Ако записи нису ажурирани или се могу лако изменити, могу настати спорови око власништва и озбиљни правни проблеми. Због тога различите државе истражују савремена информациона решења која ће омогућити већу сигурност и транспарентност катастарских података.

Блокчејн (blockchain) технологија представља начин безбедног организовања и чувања дигиталних записа. Основни елементи блокчејн система јесу блокови и њихово повезивање у ланац. Трансакција представља запис који се уноси у систем. Децентрализација омогућава да записи буду доступни већем броју учесника и да их они могу проверити. Криптовалуте представљају једну од најпознатијих примена блокчејн технологије. Она се примењује и у многим другим областима савременог друштва.

ДА ЛИ ЗНАШ?

Да ли знаш да се блокчејн (blockchain) технологија данас не користи само за криптовалуте? Осим у финансијском сектору, она се примењује или тестира у здравству, образовању, саобраћају, логистици и приликом издавања дигиталних докумената.

ИСТРАЖИ

Истражи три области у којима се примењује blockchain технологија. За сваку област наведи:

- шта се евидентира;
- какве повољности обезбеђује blockchain;
- на који начин технологија доприноси бољој сигурности података.

ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представља blockchain технологија?
2. Шта представља блок у blockchain систему?
3. Шта је трансакција?
4. Шта значи децентрализација?
5. Зошто криптовалуте претстављају једну од најпознатијих примена blockchain технологије?
6. Наведи најмање три области у којима може да се примени blockchain технологија.

ИСТРАЖИВАЧКИ И ПРАКТИЧНИ ЗАДАЦИ

1. Замисли да у твојој школи треба да се чувају електронске дипломе. Објасни на који начин би блокчејн (blockchain) технологија могла да помогне у њиховом евидентирању.
2. Истражи једну криптовалуту по свом избору и припреми кратак извештај о њеној примени и основним карактеристикама.
3. Направи табелу у којој ћеш упоредити класичну базу података и блокчејн (blockchain) систем према начину чувања информација.
4. Размисли у којим областима у Републици Северној Македонији би могла да се примени блокчејн (blockchain) технологија. Наведи најмање три примера и образложи свој избор.
5. Истражи на који начин блокчејн (blockchain) технологија може да допринесе унапређивању катастарских евиденција и управљању власништвом над непокретностима.

ЗАКЉУЧАК:

Блокчејн (blockchain) технологија пружа више предности при чувању и обради дигиталних записа. Захваљујући повезаности блокова, свака промена остаје евидентирана и може се проверити. То доприноси већој транспарентности и поверењу у систем. Поред тога, примена криптографских метода обезбеђује бољу заштиту информација и смањује могућност неовлашћених измена података.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Блокчејн (blockchain) технологија непрестано се развија и шири на нове области примене. Осим у финансијском сектору, користи се за праћење производа у ланцима снабдевања, издавање дигиталних сертификата, управљање медицинским подацима и заштиту интелектуалне својине. У многим државама и организацијама истражују се нови начини њене примене како би се побољшале сигурност, транспарентност и поузданост различитих информационих система.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

1. Заокружи тачан одговор.

Blockchain технологија се најчешће користи за:

а) обраду фотографија;

б) сигурно чување и праћење дигиталних записа;

в) израду оперативних система;

г) поправку компјутерских мрежа.

2. Тачно или нетачно.

Криптовалуте представљају јединствену примену blockchain технологије.

3. Објасни појмове.

Појам	Објашњење
Блок	_____
Трансакција	_____
Децентрализација	_____

4. Допуни.

Најпознатија примена blockchain технологије су _____.

5. Како се примењује принцип децентрализације при извршавању неке дигиталне трансакције у blockchain мрежи у поређењу са традиционалним банкарским плаћањем?

6. Анализирај улогу децентрализације у безбедности података. Зашто је готово немогуће избрисати или изменити већ забележену трансакцију након што се блок повеже у ланац?

7. Процени предности и недостатке криптовалута као најраспрострањеније примене блокчејн (blockchain) технологије. Да ли сматраш да оне у будућности могу у потпуности да замене традиционални новац? Образложи свој став.

8. Осим за криптовалуте, предложи идеју како би блокчејн (blockchain) технологија, са својим карактеристикама непроменљивости и децентрализације, могла да се примени за решавање неког стварног друштвеног проблема, као што су гласање на изборима или заштита ауторских права.

СИТУАЦИОНИ ЗАДАТАК

Замисли да треба да се води евиденција о власништву над кућама у једном насељу. Објасни на који начин би блокчејн (blockchain) технологија могла да допринесе већој сигурности и поузданости тих података.

1.5

РОБОТИКА И АУТОМАТИЗАЦИЈА

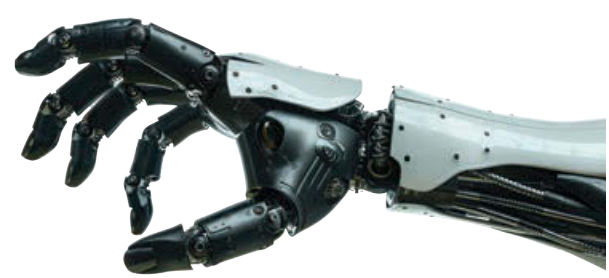
Савремене информационе технологије омогућавају стварање машина које могу самостално да извршавају различите задатке. Захваљујући развоју рачунарских система, сензора и аутоматског управљања, роботи се данас примењују у бројним областима свакодневног живота и индустрије. Они могу да извршавају прецизне, понављајуће или опасне задатке, чиме помажу људима и доприносе унапређењу радних процеса.

ВЕЋ ЗНАШ ДА...

- наводиш примере савремених технологија;
- објашњаваш као информатички системи обрађују информације;
- препознајеш примену дигиталних технологија у свакодневном животу;
- наводиш примере аутоматизације различитих процеса;
- објашњаваш као савремене технологије доприносе развоју друштва.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Да ли роботи могу да извршавају задатке које су раније људи извршавали?
2. У којим професијама си видео употребу робота?
3. Како би робот могао да помогне људима у свакодневном животу?
4. Да ли роботи увек имају људски изглед?
5. Које задатке би поверио неком роботу?



У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш појам роботике.
- наводиш примере примене робота.
- разумеш улогу аутоматизације.
- повезујеш коришћење робота у савременим технологијама.

Роботика представља савремену научну и техничку област која се бави пројектирањем, израдом и применом роботских система. Њен развој је повезан са информатиком, електроником, машинством и аутоматиком. Данас се роботи могу срести у фабрикама, болницама, пољопривреди, саобраћају, домаћинствима и у многим другим областима. Захваљујући њиховој примени, многи задаци се извршавају брже, прецизније и безбедније.

1.5.1 ПОЈАМ РОБОТИКЕ

Роботика представља научну и техничку област која се бави стварањем и применом робота. Робот је машина која може да извршава унапред одређене задатке помоћу програма и информација које добија из свог окружења. За свој рад роботи користе различите сензоре, управљачке системе и механичке делове који им омогућавају да се крећу, препознају предмете и обављају одређене активности. У зависности од њихове намене, роботи могу имати различит изглед и различите могућности.

Роботи не мора обавезно да имају људски изглед. Већина роботских система који се данас користе има облик прилагођен задацима које извршавају. У индустрији се најчешће користе роботске руке за склапање производа, у домаћинствима роботи за чишћење, а у пољопривреди аутоматизоване машине за праћење и обраду усева. Различите врсте робота створене су да помажу људима и омогуће брже и прецизније извршавање задатака.

1.5.2 АУТОМАТИЗАЦИЈА И ПРИМЕНА РОБОТА

Аутоматизација представља процес у којем се одређени задаци извршавају аутоматски, уз минимално или без непосредног учешћа човека. Роботски системи су важан део аутоматизације јер могу великом брзином и прецизношћу да обављају бројне понављајуће активности. У производњи роботи склапају производе, обављају заваривање, бојење и паковање, док у складиштима помажу при сортирању и транспорту робе.

Примена робота није ограничена само на индустрију. У медицини се користе роботски системи који помажу лекарима приликом сложених операција. У пољопривреди роботи могу да прате стање усева и допринесу прецизнијем наводњавању. У саобраћају се развијају аутономна возила и дронови, док се у домаћинствима користе роботи за чишћење и одржавање простора. Захваљујући развоју информатике, роботи постају део све већег броја активности у савременом друштву.

1.5.3 МЕДИЦИНСКИ РОБОТИ, ДРОНОВИ И САВРЕМЕНИ РОБОТСКИ СИСТЕМИ

Развој роботике омогућава стварање специјализованих робота који помажу људима у различитим областима живота и рада. У медицини се користе роботски системи који лекарима омогућавају да изводе сложене и прецизне хируршке захвате. Они помажу и при рехабилитацији пацијената, транспорту медицинске опреме и достављању лекова у болницама. Захваљујући великој прецизности, медицински роботи доприносе унапређењу здравствених услуга и смањењу могућности настанка грешака током одређених поступака.

Посебну групу савремених роботских система представљају дронови. То су беспилотне летелице којима се може управљати на даљину или које могу аутоматски да извршавају унапред задате задатке. Примењују се за фотографисање и снимање, надгледање пољопривредних површина, праћење саобраћаја, потрагу и спасавање, као и за доставу различитих производа. У неким државама дронови се користе и за транспорт медицинског материјала до тешко доступних места. Развој оваквих система показује да роботика има широку примену у савременом друштву.

1.5.4 РОБОТИ У АУТОМОБИЛСКОЈ ИНДУСТРИЈИ

У савременим фабрикама аутомобила велики број радних процеса обавља се помоћу индустријских робота. Роботске руке обављају заваривање, фарбање, склапање и премештање тешких делова великом брзином и прецизношћу. Они могу непрекидно да раде и извршавају исте задатке са једнаким квалитетом, што доприноси повећању производње. Иако роботи обављају велики део

посла, они не замењују људе у потпуности. За њихово програмирање, одржавање и контролу потребна су стручна лица са различитим знањима и вештинама. На овај начин роботика ствара нове могућности за развој савремених занимања повезаних са информатиком, електроником и инжењерством.

1.5.5 ПРЕДНОСТИ И ОГРАНИЧЕЊА РОБОТИКЕ

Роботика доноси бројне предности савременом друштву. Роботи могу да извршавају задатке који захтевају велику прецизност, брзину или издржљивост. У индустрији помажу у производњи различитих производа, у медицини доприносе унапређењу здравствених услуга, а у пољопривреди омогућавају ефикасније праћење и обраду усева. Роботи могу да раде у опасним срединама, као што су рудници, подручја погођена природним катастрофама или свемирске мисије, чиме се смањује ризик за људе. Њихова примена доприноси повећању продуктивности и побољшању квалитета различитих радних процеса.

Упркос бројним предностима, роботски системи имају и одређена ограничења. Они раде према програмима и правилима које је осмислио човек и не могу самостално да замене многе људске способности, као што су креативност, емпатија и морално расуђивање. За њихов рад потребни су редовно одржавање, техничка подршка и стручна лица која ће их програмирати и контролисати. Развој роботике отвара и нова питања о безбедности, одговорности и утицају аутоматизације на различита занимања. Због тога роботске системе треба развијати и примењивати тако да доприносе унапређењу живота и рада људи.

1.5.6 РОБОТИКА И БУДУЋНОСТ

Развој роботике наставља се великом брзином и очекује се да ће роботски системи у будућности постати још заступљенији у свакодневном животу. У паметним градовима развијају се аутономна возила, роботи за одржавање јавних простора и аутоматизовани системи за управљање саобраћајем. У медицини се стварају роботи који ће помагати лекарима при дијагностиковању и лечењу пацијената, док се у пољопривреди развијају системи који ће пратити стање усева и аутоматски обављати различите активности.

Развој роботике уско је повезан са развојем информатике и вештачке интелигенције. Савремени роботи користе различите сензоре за прикупљање информација из окружења и рачунарске системе за обраду тих информација. Захваљујући томе, могу да препознају предмете, избегавају препреке и доносе једноставне одлуке при извршавању својих задатака. У будућности се очекује још већа сарадња робота и вештачке интелигенције, која ће створити нове могућности за развој савременог друштва.

1.5.7 РОБОТИ У СПАСИЛАЧКИМ МИСИЈАМА

Након земљотреса, пожара и других природних катастрофа, спасилачке екипе се често суочавају са опасним условима рада. У таквим ситуацијама користе се специјални роботи и дронови који могу да приступе местима на којима је кретање људи отежано или опасно. Они су опремљени камерама и различитим сензорима који омогућавају претраживање терена и проналажење особа којима је потребна помоћ.

Примена роботских система у спасилачким мисијама доприноси бржем прикупљању информација и бољој организацији спасилачких активности. Истовремено се смањује ризик за људе који учествују у овим операцијама. Овај пример показује да роботика не служи само за аутоматизацију производње већ има важну улогу и у заштити људског живота.

ДА ЛИ ЗНАШ?

Да ли знаш да на дну океана и на површини Марса раде специјални роботи који прикупљају податке и обављају истраживања на местима до којих човек тешко може да стигне?

ИСТРАЖИ

Изабери једну област у којој се користе роботи (медицина, индустрија, пољопривреда, транспорт, свемир или спасилачке мисије). Припреми кратак извештај у коме ћеш објаснити:

- какви роботи се користе;
- које задатке извршавају;
- какве повољности обезбеђују;
- са каквим изазовима се суочавају.

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА:

1. Шта представља роботика?
2. Шта представља аутоматизација?
3. Да ли сви роботи имају људски изглед? Образложи одговор.
4. У којим областима из свакодневног живота и рада се примењују роботи?
5. Како роботи доприносе побољшању производње и услуга?
6. Које су најважније предности примене роботских система?
7. Која ограничења и изазови постоје при њиховој примени?

ИСТРАЖИВАЧКИ И ПРАКТИЧНИ ЗАДАЦИ

1. Направи табелу у којој ћеш упоредити четири различите врсте робота (индустријске, медицинске, кућне и дронове). За сваку врсту наведи:

- област примене;
- задаци које извршава;
- користи од његове употребе.

2. Истражи како се користе роботи у савременој пољопривреди. Припреми кратак извештај и наведи најмање три активности које могу да извршавају.

3. Замисли да твоја школа набави робота који ће помагати ученицима и наставницима. Опиши:

- какав задатак би извршавао;
- које технологије би користио;
- како би побољшао рад школе.

4. Направи кратку презентацију о примени робота у медицини или спасилачким мисијама. Укључи примере и објасни које користи обезбеђују.

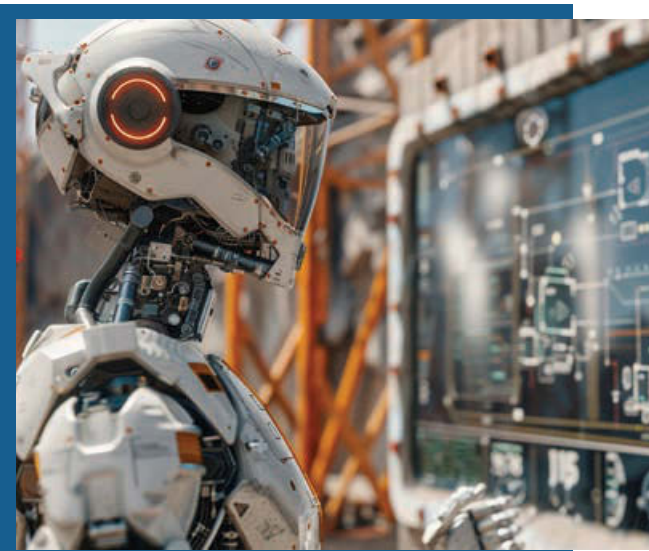
5. Размисли које професије би могли да користе роботи у будућности. Наведи најмање пет примера и објасни на који начин би помагао људима.

ЗАКЉУЧАК:

Роботика представља научну и техничку област која се бави стварањем и применом робота. Роботи не морају да имају људски изглед и могу бити прилагођени различитим задацима. Аутоматизација омогућава да се одређене активности извршавају аутоматски. Роботи се примењују у индустрији, медицини, пољопривреди, саобраћају, домаћинствима и многим другим областима. Роботски системи могу да извршавају опасне, тешке и понављајуће задатке. Развој роботике уско је повезан са информатиком и савременим технологијама.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Последњих година развија се нова генерација робота који могу да сарађују са људима при извршавању различитих задатака. Ови системи користе сензоре, камере и савремена информациона решења како би боље препознали своје окружење. Поред индустријских робота, развијају се роботи за помоћ старијим особама, аутономна возила, роботи за истраживање океана и свемира, као и роботи који учествују у научним експериментима. Очекује се да ће роботика и у будућности имати значајну улогу у развоју савременог друштва.



ПРОВЕРИ СВОЈЕ ЗНАЊЕ

1. Заокружи тачан одговор.

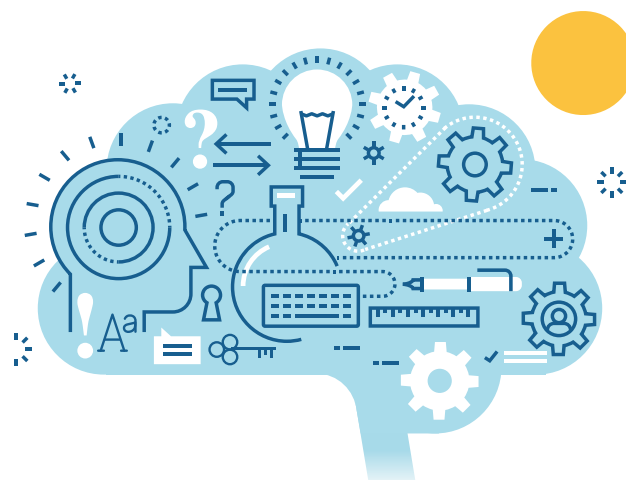
Роботика представља област која се бави:

а) производњом електричне енергије;

б) креирањем и применом роботских система;

в) проучавањем планета;

г) поправком компјутера.



2. Тачно или нетачно.

Сви роботи имају људски изглед.

3. Допуни.

Процес при којем се одређени задаци извршавају аутоматски назива се _____.

4. Објасни појмове.

Појам	Објашњење
Робот	_____
Роботика	_____
Аутоматизација	_____

СИТУАЦИОНИ ЗАДАТАК

Замисли да у једном складишту сваког дана треба премештати тешке терете. Објасни како би роботски системи могли да помогну у овом послу и које би предности обезбедили.

1.6

ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА И МАШИНСКО УЧЕЊЕ

Људи свакодневно користе различите дигиталне услуге које могу да препознају говор, анализирају фотографије, предлажу садржаје и помажу при доношењу одлука. Мобилни телефони могу да препознају лице корисника, интернет платформе предлажу филмове и музику на основу интересовања корисника, а навигациони системи бирају најпогоднију путању до одређене локације. Ове могућности резултат су развоја савремених информационих технологија и примене вештачке интелигенције.

ВЕЋ ЗНАШ ДА...

- наводиш примере савремених технологија;
- објашњаваш роботи користе информације из околине;
- препознајеш примену аутоматизације;
- наводиш примере дигиталних услуга;
- објашњаваш како информатички системи обрађују податке.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Како мобилни телефон препознаје лице свог власника?
2. Како интернет-продавнице предлажу производе који би могли да те интересују?
3. Како апликације за навигацију бирају најбољу маршруту?
4. Да ли компјутерски системи могу да уче од података које обрађују?
5. У којим областима свакодневног живота си срео примену вештачке интелигенције?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш шта представља вештачку интелигенцију;
- објашњаваш шта представља машинско учење;
- наводиш примере примене вештачке интелигенције;
- анализираш како функционише ВИ;
- препознајеш предности и ризике датог примера.

Вештачка интелигенција представља област информатике која се бави стварањем рачунарских система способних да обрађују информације и решавају задатке који обично захтевају људску интелигенцију. Ови системи за свој рад користе велике количине података и различите алгоритме за анализу и обраду информација. Захваљујући томе, могу да препознају обрасце, предвиђају могуће резултате и помажу људима при обављању различитих активности.

1.6.1 ШТА ПРЕДСТАВЉА ВЕШТАЧКУ ИНТЕЛИГЕНЦИЈУ?

Вештачка интелигенција представља област информатике која се бави развојем рачунарских система способних да извршавају задатке који обично захтевају људско размишљање. Такви системи могу да препознају слике, обрађују говор, анализирају податке и помажу при доношењу одређених одлука. За свој рад користе информације које добијају из различитих извора и примењују посебне алгоритме за њихову обраду. Основни циљ вештачке интелигенције јесте стварање система који ће моћи ефикасније да решавају различите практичне задатке.

Примена вештачке интелигенције данас је присутна у многим областима свакодневног живота. При коришћењу интернет претраживача, друштвених мрежа, платформи за музику и филмове или мобилних телефона, корисници свакодневно користе услуге које примењују алгоритме вештачке интелигенције. У медицини ови системи помажу при анализи медицинских снимака, у саобраћају доприносе развоју аутономних возила, а у образовању омогућавају стварање интерактивних система за учење. Захваљујући оваквој примени, вештачка интелигенција постаје важан део савременог друштва.

1.6.2 ШТА ПРЕДСТАВЉА МАШИНСКО УЧЕЊЕ?

Машинско учење представља област вештачке интелигенције која омогућава рачунарским системима да уче из података које обрађују. Уместо да се за сваку ситуацију пишу посебна правила, систем анализира велики број примера и постепено открива одређене законитости и обрасце. Што више одговарајућих података систем обрађује, то успешније може да извршава задатке за које је намењен.

Машинско учење има широку примену у савременим информационим системима. Користи се за препознавање говора, аутоматско превођење текстова, препознавање лица и предмета на фотографијама, откривање нежељених електронских порука и препоручивање различитих садржаја на интернету. На пример, када корисник гледа видео-снимке на интернету, систем анализира које садржаје најчешће бира и предлаже му нове видео-снимке са сличном тематиком. На овај начин машинско учење помаже рачунарским системима да боље разумеју потребе корисника и пруже корисније услуге.

1.6.3 ПРИМЕНА ВЕШТАЧКЕ ИНТЕЛИГЕНЦИЈЕ

Вештачка интелигенција данас се примењује у великом броју области савременог друштва. У здравству помаже при анализи медицинских података и снимака, у саобраћају доприноси развоју система за управљање саобраћајем и аутономних возила, а у образовању се користи за стварање интерактивних образовних садржаја и система за учење. У пољопривреди се примењује за праћење стања усева, док у индустрији помаже при аутоматизацији производних процеса и контроли квалитета производа.

Велики број дигиталних услуга које људи свакодневно користе примењује различите методе вештачке интелигенције. Интернет претраживачи помажу у проналажењу информација, навигациони системи предлажу одговарајуће путање, а различите платформе препоручују музику, филмове или друге садржаје у складу са интересовањима корисника. При обради фотографија и видео-записа, савремени системи могу да препознају лица и предмете, док при коришћењу мобилних телефона могу да помогну у препознавању говора и његовом претварању у текст. Ови примери показују да је вештачка интелигенција већ постала део свакодневног живота.

1.6.4 КАКО ФУНКЦИОНИШЕ ВИ?

Сваки пут када откључаш телефон помоћу препознавања гласа или када ти YouTube предложи следећи омиљени видео-снимак, користиш моћ вештачке интелигенције (ВИ). Иако на први поглед делује као магија, иза ових процеса стоје прецизни математички поступци, анализа података и алгоритми машинског учења. Да бисмо разумели како ВИ заиста функционише, анализираћемо два примера из стварног живота које свакодневно користимо.

ПРИМЕР 1: Системи за препоруке (Како TikTok, Netflix или Spotify знају шта волиш?)

Да ли си се некада запитао/ла како нека апликација успева да задржи твоју пажњу часовима? Користи ВИ систем за препоруке. Овај систем није погодан случајно, већ ради кроз четири кључне фазе:

1. **Сабирање података (твој дигитални отисак):** Док користиш апликацију, алгоритам будно прати сваки твој потез. Он прикупља следеће податке: на којим објавама се дуже задржаваш, које видео-снимке прескачеш након прве секунде, шта ти се допада, шта делиш или коментаришеш.
2. **Препознавање образаца (проналажење шема):** ВИ анализира ове податке и сврстава те у групу са хиљадама других корисника који имају сличне навике. Ако корисник Марко и ти гледате иста три видео-снимка о кувању, ВИ вас означава као "дигиталне близанце" са сличним интересовањима.
3. **Филтрирање и предвиђање:** Ако Марко погледа четврти видео-снимак (на пример, нови рецепт за пицу) и одгледа га до краја, алгоритам математички израчунава да постоји вероватноћа од 95% да ће се тај видео-снимак допасти и теби.
4. **Резултат:** Видео-снимак о пици појављује се на твојој почетној страници. Ако га погледаш, систем добија потврду да је његова процена била тачна, а алгоритам постаје још паметнији за следећу препоруку.

ПРИМЕР 2: Препознавање говора (Како функционишу Siri, Google Assistant или диктирање порука?)

Када кажеш "Hey Siri" или користиш опцију за претварање говора у текст (Voice-to-Text), вештачка интелигенција треба да премости јаз између људског (биолошког) и рачунарског (дигиталног) света. Тај сложени процес одвија се на следећи начин:

1. **Акустична анализа:** Твој глас је звучни талас. Микрофон уређаја снима тај талас и претвара га у дигитални сигнал (низ бројева). ВИ пречишћава овај сигнал од околне буке (на пример, од буке аутомобила на улици).
2. **Разбијање фонема:** ВИ дели снимљени говор на најмање могуће звучне јединице које се називају фонеме. На пример, ако изговориш реч "компјутер", алгоритам је препознаје као спој гласова к-о-м-п-ј-у-т-е-р.
3. **Језичко моделирање:** Ово је кључни део ВИ. Алгоритам упоређује те гласове са огромном базом података (милионима сати снимљеног људског говора) како би утврдио која реч највише одговара том звуку. Он такође анализира и контекст реченице. Ако кажеш "Желим јабуку", ВИ зна да мислиш на воће, а не на технолошку компанију Apple, захваљујући претходној речи "желим".
4. **Извршавање наредби:** Након што ВИ разуме текст, она активира одговарајућу функцију на телефону – исписује реч на екрану, пушта песму коју си затражио/ла или ти одговара на питање генерисаним гласом.

Из ова два примера видимо да ВИ не "размишља" као човек са емоцијама и свешћу. Она функционише као моћан статистички процесор. Принцип је увек исти: Улазни подаци (твоји "лајкови" или глас) → Математичка обрада и поређење са претходним знањем → Излазни резултат (нова препорука или написан текст).

Што више података ВИ има на располагању, то ће прецизније функционисати у свакодневним ситуацијама.

1.6.5 ПРЕДНОСТИ И РИЗИЦИ ВЕШТАЧКЕ ИНТЕЛИГЕНЦИЈЕ

Вештачка интелигенција (ВИ) несумњиво је једна од најважнијих технолошких иновација у историји човечанства. Она више није само идеја из научне фантастике, већ алат који активно преобликује медицину, образовање, пословање и наш свакодневни живот. Ипак, као и свака моћна технологија, ВИ доноси огроман потенцијал за напредак, али и озбиљне изазове. Да бисмо били спремни за будућност, морамо детаљно анализирати обе стране "медаље": предности и могуће ризике њене употребе.

ПРЕДНОСТИ КОРИШЋЕЊА ВИ

Примена ВИ у друштву доноси револуционарне предности које живот чине лакшим, безбеднијим и ефикаснијим.

- Автоматизација и огромна ефикасност: • ВИ може да обради милијарде података за неколико секунди и да обавља репетитивне (понављајуће/монотоне) задатке без одмора. То омогућава људима да се ослободе административних послова и усредсреде на креативно размишљање и решавање сложених проблема.
- Напредак у медицини и спашавање живота: Алгоритми ВИ могу да анализирају медицинске снимке (као што су рендгенски снимци или магнетна резонанца) са прецизношћу већом од људског ока, откривајући болести у најранијој фази. Такође, ВИ драматично убрзава процес откривања и тестирања нових лекова.
- Персонализовано учење: У образовању паметни системи могу да прилагоде наставни материјал брзини и стилу учења сваког ученика појединачно. Ако ученик "запне" на одређеној теми, ВИ асистент може да понуди другачија објашњења и додатне вежбе.
- Повећана безбедност у опасним срединама: Роботи којима управља ВИ могу да извршавају задатке опасне по живот људи, као што су деминирање терена, истраживање океанских дубина, гашење великих пожара или рад у нуклеарним електранама.

МОГУЋИ РИЗИЦИ КОРИШЋЕЊА ВИ

Упркос свим предностима, неконтролисани развој вештачке интелигенције отвара озбиљна етичка, друштвена и безбедносна питања.

- **Утицај на тржиште рада (губљење радних места):** Брзи развој ВИ изазива страх од масовне незапослености. Системи засновани на ВИ могу да замене

људе у занимањима из области као што су корисничка подршка, превођење и вожња (аутономна возила), па чак и млађе програмере или писце, због чега се јавља хитна потреба за преквалификацијом људи.

- **Пристрасност и дискриминација (algorithmic bias):** ВИ учи из података које су створили људи. Ако ти подаци садрже историјске предрасуде или дискриминацију (на расној, родној или националној основи), ВИ ће усвојити те предрасуде и наставити да доноси неправедне одлуке – на пример, при избору кандидата за посао или одобравању банкарских кредита.
- **Губитак приватности:** Да би функционисала прецизно, ВИ су потребне огромне количине личних података. Озбиљан ризик представља стални дигитални надзор, у којем се сваки наш корак, разговор, претрага и емоција прате, анализирају и потенцијално злоупотребљавају у комерцијалне или политичке сврхе.
- **Ширење дезинформација и "Deepfakes":** Напредни алати ВИ омогућавају лако стварање лажних вести, уверљивих текстова, па чак и лажних видео и аудио-снимака на којима јавне личности или обични људи изгледају као да говоре нешто што никада нису изговорили. То угрожава поверење у информације и може да изазове политичку нестабилност.
- **Когнитивна лењост код људи:** Претерано ослањање на ВИ при писању домаћих задатака, решавању проблема и доношењу одлука може дугорочно да умањи критичко мишљење, логичко расуђивање и креативне способности младих генерација.

Упркос бројним могућностима, рад са вештачком интелигенцијом носи и одређене ризике. Њен рад зависи од квалитета и количине података које обрађује, као и од правила према којима је створена. Савремени системи могу да погреше или да пруже нетачне резултате, па су зато неопходни људска провера и надзор. Одговорна примена вештачке интелигенције подразумева њено коришћење као помоћне технологије која доприноси унапређењу различитих активности и услуга.

Вештачку интелигенцију треба користити креативно, етички и одговорно. Кључ њене успешне примене лежи у доношењу строгих етичких закона и прописа који ће штитити људска права и приватност, а истовремено подстицати технолошки напредак. Циљ човечанства не треба да буде замена човека вештачком интелигенцијом, већ стварање синергије у којој ће ВИ помагати човеку да постигне више.

1.6.6 ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА И САВРЕМЕНО ДРУШТВО

Вештачка интелигенција има све већи значај у развоју савременог друштва. Њена примена доприноси унапређењу различитих услуга и олакшавању свакодневних активности људи. У здравству помаже при анализи медицинских података, у образовању омогућава стварање савремених дигиталних алата за учење, а у саобраћају доприноси бољем управљању саобраћајним системима. У индустрији се користи за аутоматизацију производње и контролу квалитета, док у пољопривреди помаже при праћењу стања усева и ефикаснијем коришћењу ресурса.

Развој вештачке интелигенције отвара нове могућности за научна истраживања, економски развој и побољшање квалитета живота. Истовремено, неопходно је да се савремене технологије користе одговорно и пажљиво. Људи имају важну улогу у развоју, контроли и примени ових система, јер они стварају правила према којима системи функционишу. Зато је, поред техничких знања, важно развијати и критичко мишљење, дигиталну писменост и одговоран однос према коришћењу информационих технологија.

Вештачката интелигенција и будућност

Са развојем информатике очекује се да ће вештачка интелигенција налазити примену у све већем броју области. Савремени системи помагаће у научним истраживањима, развоју нових лекова, унапређењу образовања, заштити животне средине и управљању сложеним техничким системима. У многим занимањима вештачка интелигенција биће алат који ће помагати људима да брже обрађују информације и лакше решавају различите задатке.

Упркос брзом развоју технологије, човек остаје најважнији чинилац у доношењу одлука и примени савремених система. Вештачка интелигенција представља средство које може да допринесе унапређењу различитих активности, али не може у потпуности да замени људско знање, креативност, искуство и одговорност. Због тога ће будући развој ове области зависити од успешне сарадње људи и савремених информационих технологија.

Вештачка интелигенција у образовању

Савремене образовне платформе користе различите технологије вештачке интелигенције како би помогле ученицима и наставницима. Такви системи могу да предлажу додатне материјале за учење, помажу при проналажењу информација и обезбеђују интерактивне активности за савладавање наставних садржаја. На овај начин ученици могу лакше да приступе различитим изворима знања и организују процес учења.

Вештачка интелигенција не замењује наставника, већ представља додатни алат који може да допринесе унапређењу наставног процеса. Наставник има важну улогу у избору наставних садржаја, усмеравању ученика и провери тачности информација. Одговорно коришћење савремених технологија доприноси развоју дигиталних компетенција и критичког мишљења код ученика.

ДА ЛИ ЗНАШ?

Да ли знаш да свакодневно користиш системе који примењују вештачку интелигенцију? Интернет претраживачи, апликације за навигацију, системи за препознавање говора и многе дигиталне услуге користе различите методе за обраду података и помажу при решавању задатака.

ИСТРАЖИ

Истражи пет примера примене вештачке интелигенције у свакодневном животу. За сваки пример објасни:

- где се користи;
- какав задатак извршава;
- на који начин помаже људима.

ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представља вештачку интелигенцију?
2. Шта представља машинско учење?
3. У којим областима савременог друштва се примењује вештачка интелигенција?
4. На који начин вештачка интелигенција помаже у свакодневном животу?
5. Које су најважније предности примене вештачке интелигенције?
6. Која ограничења постоје при коришћењу система са вештачком интелигенцијом?
7. Зашто је важно да се вештачка интелигенција користи одговорно?

ИСТРАЖИВАЧКИ И ПРАКТИЧНИ ЗАДАЦИ

1. Направи листу десет примера примене вештачке интелигенције у свакодневном животу. За сваки пример наведи где се користи и какав задатак извршава.
2. Истражи како се користи вештачка интелигенција у медицини, у образовању или пољопривреди. Припреми кратак извештај.
3. Замисли да твоја школа жели да уведе систем вештачке интелигенције. Предложи три активности у којима би могао да помогне и објасни повољности.
4. Направи табелу у којој ћеш да упоредиш човека и систем са вештачком интелигенцијом према:
   - брзина обраде података;
   - учењу;
   - креативности;
   - доношење одлука;
   - потреба за људским надзором.
5. Процени предности увођења ВИ у медицину и образовање у односу на потенцијални ризик од губитка радних места у будућности. Да ли сматраш да предности оправдавају развој ове технологије? Образложи свој став.
6. Имајући у виду ризике повезане са приватношћу података и појавом "deepfake" (лажних) видео и аудио-снимака, донеси закључак: Који су етички изазови коришћења ВИ најопаснији за младе генерације и како утичу на њихову способност критичког мишљења?
7. Размисли и напиши кратак текст на тему:

„Како може вештачката интелигенција може да побољша свакодневни живот људи?""

ЗАПАМТИ ШТА СИ НАУЧИО

Вештачка интелигенција представља област информатике која се бави креирањем интелигентних компјутерских система.

Машинско учење представља део вештачке интелигенције који омогућава учење из података.

Вештачка интелигенција примењује се у здравству, образовању, индустрији, саобраћају, пољопривреди и многим другим областима.

Савремене дигиталне услуге свакодневно користе различите методе вештачке интелигенције.

Вештачка интелигенција доноси бројне предности, али има и одређена ограничења.

Одговорна примена вештачке интелигенције има значајну улогу у развоју савременог друштва.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Развој вештачке интелигенције представља једну од најдинамичнијих области савремене информатике. Данас се стварају системи који могу да обрађују велике количине података, помажу у научним истраживањима и доприносе решавању сложених проблема. Вештачка интелигенција користи се при истраживању свемира, развоју нових лекова, заштити животне средине и у многим другим активностима. Са даљим развојем технологије очекује се да ће њена примена наставити да расте, при чему ће човек имати важну улогу у контроли и одговорном коришћењу ових система.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

1. Заокружи тачан одговор.

Вештачка интелигенција представља:

а) програм за обраду текста;
б) област информатике који прави интелигентне компјутерске системе;
в) рачунарску игру;
г) оперативни систем.

2. Тачно или нетачно.

Машинско учење представља део вештачке интелигенције.

3. Допуни.

Област која омогућава рачунарским системима да уче из података назива се _____.

4. Објасни појмове.

Појам	Објашњење
Вештачка интелигенција	_____
Машинско учење	_____
Алгоритам	_____

СИТУАЦИОНИ ЗАДАТАК

Замисли да једна школа жели да користи систем вештачке интелигенције за унапређивање наставе. Наведи две активности у којима би систем могао да помогне и објасни зашто је при његовом коришћењу неопходан људски надзор.

1.7

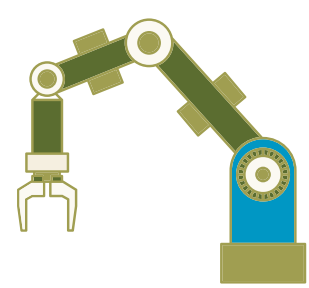
ВЕЛИКИ ЈЕЗИЧКИ МОДЕЛИ И ГЕНЕРАТИВНА ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА

Велики језички модели, познатији по енглеској скраћеници LLM (Large Language Models), представљају напредне системе вештачке интелигенције који су обучени на огромним количинама текстуалних података како би разумели, обрадили и тумачили људски језик. Њихова основна и најфасцинантнија способност јесте револуционарно генерисање текста. На основу задатог питања или контекста, модел може самостално да пише есеје, чланке, песме или детаљна објашњења која звуче природно, као да их је написао човек.

Ова технолошка основа омогућила је стварање паметних четбот система (као што је ChatGPT), који омогућавају интерактивну комуникацију у реалном времену и пружају тренутне одговоре на најразличитија питања корисника. Захваљујући својој флексибилности, ови модели имају широку примену у образовању, где служе као персонализовани дигитални тутори који помажу ученицима да савладају тешке лекције или преводе текстове на различите језике.

Истовремено, донели су велику трансформацију и у свет технологије, омогућавајући моћну примену у програмирању, где их програмери користе за аутоматско писање, прегледање, проналажење грешака (отклањање грешака, енгл. debugging) и објашњавање компјутерског кода.

ВЕЋ ЗНАШ ДА...

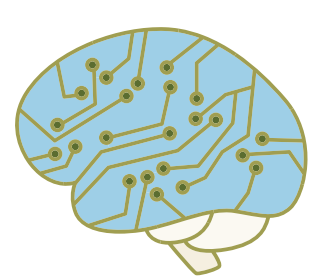


- објашњаваш шта представља вештачку интелигенцију;
- објашњаваш шта представља машинско учење;
- наводиш примере примене вештачке интелигенције;
- анализираш предности и ограничења ВИ;
- препознајеш примену савремених информатичких технологија.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Како може компјутерски систем да одговори на питање постављено природним језиком?
2. Како је могуће компјутер да направи текст, слику или музику?
3. Да ли компјутерски систем увек даје тачне информације?
4. Како може генеративна вештачка интелигенција да помогне у образовању?
5. Зашто је важно критички да се проверавају информације добијене савременим дигиталним системима?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:



- објашњаваш шта представља велике јазичке моделе;
- наводиш примере њихове примене;
- разумеш како се користе за генерисање текста;
- идентификујеш могућности за примену у учењу и раду.

Последњих година вештачка интелигенција значајно се развија и стиче нове могућности за обраду и стварање различитих врста садржаја. Савремени рачунарски системи могу да разумеју питања постављена природним језиком, стварају текстове, помажу при превођењу, израђују сажетке и генеришу различите идеје. Захваљујући овим способностима, примењују се у образовању, науци, пословању и многим другим областима.

Велики језички модели представљају савремене системе вештачке интелигенције који се обучавају обрадом великих количина текстуалних података. Захваљујући томе, могу да препознају језичке обрасце и стварају нове садржаје у складу са захтевима корисника. Генеративна вештачка интелигенција не само да обрађује информације већ може и да ствара текстове, слике, музику и друге дигиталне садржаје. Због широке примене ових технологија, важно је да корисници разумеју њихове могућности, али и ограничења.

1.7.1 ШТА ПРЕДСТАВЉАЈУ ВЕЛИКИ ЈЕЗИЧКИ МОДЕЛИ?

Велики језички модели представљају савремене рачунарске системе који могу да обрађују и стварају текст на природном језику. За свој рад користе велике количине података и различите алгоритме вештачке интелигенције, помоћу којих препознају везе између речи и реченица. Захваљујући томе, могу да одговарају на питања, објашњавају појмове, преводе текстове и помажу при стварању различитих садржаја.

Велики језички модели (LLM) представљају изузетно напредне системе вештачке интелигенције који су осмишљени да разумеју и генеришу људски језик. Реч „велики" није употребљена случајно.

Она означава две ствари:

1. Огромна база података. Ови модели су прочитали милијарде страница текста са интернета – књиге, статије, Википедија, научни радови и компјутерски код.
2. Милијарде параметара. Они имају сложену унутрашњу структуру (дигитална неуронска мрежа) која омогућава да повежу речи и схвате контекст реченице такорећи једнако добро као и људи.

Најједноставније речено, LLM је као суперпаметан дигитални читалац који је прочитао такорећи све књиге на свету и сада може да разговара са тобом на било коју тему.

Велики језички модели примењују се у образовању, науци, пословању и различитим дигиталним услугама. Они могу да помогну при претраживању информација, припреми текстова, учењу страних језика и организовању различитих активности. Ипак, важно је знати да ови системи не размишљају на исти начин као људи, већ стварају садржаје на основу обрађених података и алгоритама према којима функционишу.

Данас постоји неколико водећих технолошких гиганата који развијају најмоћније LLM алате на свету. Сигурно већ користиш неке од њих:

- **ChatGPT (OpenAI):** Први алат који је изазвао револуцију у јавности. Одличан је за свакодневни разговор, креативно писање и објашњавање сложених појмова.
- **Gemini (Google):** Модел који је директно повезан са Google претраживачем, што му омогућава брз приступ најновијим и најтачнијим информацијама на интернету у реалном времену.
- **Claude (Anthropic):** Алат познат по изузетно природном стилу писања, логичком расуђивању и способности обраде веома обимних и дугачких докумената.
- Copilot (Microsoft): Асистент који је уграђен у програме које користиш за школу (Word, PowerPoint) и који помаже при писању и дизајнирању у њима.

1.7.2 КАКО ФУНКЦИОНИШЕ ГЕНЕРИСАЊЕ ТЕКСТА?

Многи мисле да LLM модели "размишљају" или да знају истину. Реалност је другачија – то су моћне статистичке машине за предвиђање.

Када моделу поставиш питање (то питање се назива инструкција, енгл. prompt), он не претражује базу података да би копирао и "налепио" готов одговор. Уместо тога, ради следеће:

- Анализира твоје питање реч по реч.
- На основу свега што је прочитао у прошлости, рачуна која реч логички треба да буде следећа.
- На пример, ако реченица почиње речима: „Главни град Македоније је...", модел математички предвиђа да ће следећа реч бити „Скопље", са вероватноћом од 99,9%.

Процес се одвија невероватном брзином, реч по реч, стварајући текст који изгледа као да га у реалном времену пише прави човек.

LLM моделе можеш да користиш као твој лични асистент који је доступан 24/7 у учењу као твој лични супертутор:

- Објашњавање сложених лекција. Ако не разумеш лекцију из физике или хемије, можеш да кажеш моделу: „Објасни ми Други Њутнов закон као да имам 10 година и дај ми пример из свакодневног живота".
- Учење страних језика. Можеш да разговараш са ChatGPT-ом на енглеском или немачком језику и затражиш да у одговору исправља твоје граматичке грешке.
- Припрема за тестове. Можеш да приложиш своје белешке и кажеш моделу: „Направи ми квиз од пет питања са вишеструким избором како бих проверио/ла своје знање."

У раду и програмирању:

- Писање и дебаговање кода. Ако у школи учиш програмирање (на пример Python или C++), LLM може да ти помогне да пронађеш где недостаје тачка са запетом (;) или зашто код не ради.
- Генерисање идеја. Ако треба да израдиш пројекат или есеј, а немаш идеју како да почнеш, алатка може да ти предложи пет различитих структура и тема за писање.
- Аутоматизација монотоних задатака. У пословању се ови модели користе за брзо одговарање на електронску пошту, писање извештаја и превођење обимних уговора.

Важно правило: LLM алати су одлични сарадници, али лоша замена за твоје когнитивне способности. Понекад могу да "халуцинирају" (да изнесу нетачне чињенице са великом сигурношћу). Зато увек проверавај важне информације и користи ове алате да унапредиш своје знање, а не да избегнеш самостално размишљање!

Велики језички модели и генеративна вештачка интелигенција примењују се у различитим областима савременог друштва. У пословању се користе за припрему докумената и анализу информација, док у науци могу да помогну при обради великих количина података и организовању истраживања. Захваљујући њиховој примени, многе активности могу да се обаве брже и ефикасније.

Генеративна вештачка интелигенција може да ствара различите врсте дигиталних садржаја. Поред текстова, савремени системи могу да помогну при стварању слика, музике, презентација и других мултимедијалних материјала. Ове технологије примењују се у образовању, дизајну, уметности, медијима и многим другим областима. Ипак, створене садржаје треба пажљиво анализирати и проверити њихову тачност и примереност конкретној намени.

1.7.3 ОДГОВОРНО КОРИШЋЕЊЕ ГЕНЕРАТИВНЕ ВЕШТАЧКЕ ИНТЕЛИГЕНЦИЈЕ

Савремени системи генеративне вештачке интелигенције представљају корисне алате који могу да помогну људима у различитим активностима. Ипак, њихово коришћење захтева одговорност и критичко мишљење. Информације које ови системи стварају не треба аутоматски прихватати као потпуно тачне, већ их треба упоређивати са поузданим изворима и пажљиво проверавати. Посебно је важно примењивати ово правило при коришћењу информација из области образовања, здравства и других значајних области.

При коришћењу генеративне вештачке интелигенције потребно је поштовати етичка начела и правила безбедног коришћења дигиталних технологија. Треба водити рачуна о заштити личних података и приватности, као и о поштовању ауторских права и интелектуалне својине. Генеративну вештачку интелигенцију треба користити као помоћни алат који доприноси учењу, истраживању и стварању нових идеја, а не као замену за људско знање, креативност и одговорност.

1.7.4 САВРЕМЕНА ТЕХНОЛОГИЈА У ПРАКСИ

Генеративна вештачка интелигенција у образовању

Ученик/ца припрема презентацију о климатским променама. Помоћу генеративне вештачке интелигенције може да добије идеје за структуру презентације, објашњења појмова и предлоге за додатне теме за истраживање. Затим проверава информације у уџбеницима, енциклопедијама и другим поузданим изворима и допуњује их сопственим закључцима.

На овај начин генеративна вештачка интелигенција представља алат који помаже у процесу учења, али не замењује самостално размишљање и истраживање. Одговорно коришћење оваквих система доприноси развоју дигиталних компетенција, критичког мишљења и способности анализирања различитих извора информација.

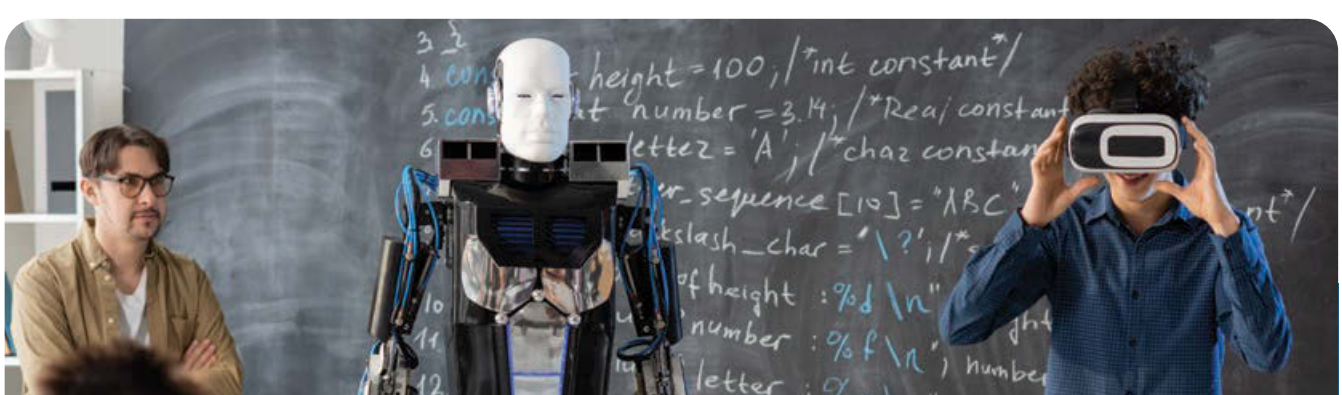
ДА ЛИ ЗНАШ?

Да ли знаш да генеративна вештачка интелигенција може да ствара различите врсте садржаја, као што су текстови, слике, музика и видео-записи, али да човек има важну улогу у провери и одговорном коришћењу тих садржаја?

ИСТРАЖИ

Истражи три области у којима се примењује генеративна вештачка интелигенција. За сваку област наведи:

- какве садржаје може да ствара;
- како помаже људима;
- зашто је важно да се користи одговорно.



ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представљају велики језички модели?
2. Шта представља генеративна вештачка интелигенција?
3. Какве врсте садржаја може да прави генеративна вештачка интелигенција?
4. У којим областима се примењују велики језички модели?
5. Зашто је потребно да се критички проверавају информације добијене од генеративне вештачке интелигенције?
6. Зашто је важно да се поштују ауторска права и приватност при коришћењу ових технологија?
7. Како може генеративна вештачка интелигенција да помогне у образовању?

ИСТРАЖУВАЧКИ И ПРАКТИЧНИ ЗАДАЦИ

1. Истражи три различите примене генеративне вештачке интелигенције у свакодневном животу и припреми кратак извештај.
2. Замисли да треба да припремиш презентацију о заштити животне средине. Опиши на који начин генеративна вештачка интелигенција може да ти помогне у припреми и које информације обавезно треба самостално да провериш.
3. Направи табелу у којој ћеш да упоредиш класичан интернет-претраживач и генеративну вештачку интелигенцију према:

- начину рада;
- врсти информација;
- примени;
- потреби за провером информација.

4. Размисли и наведи пет правила за безбедно и одговорно коришћење генеративне вештачке интелигенције.
5. Напиши краток есеј на тему:

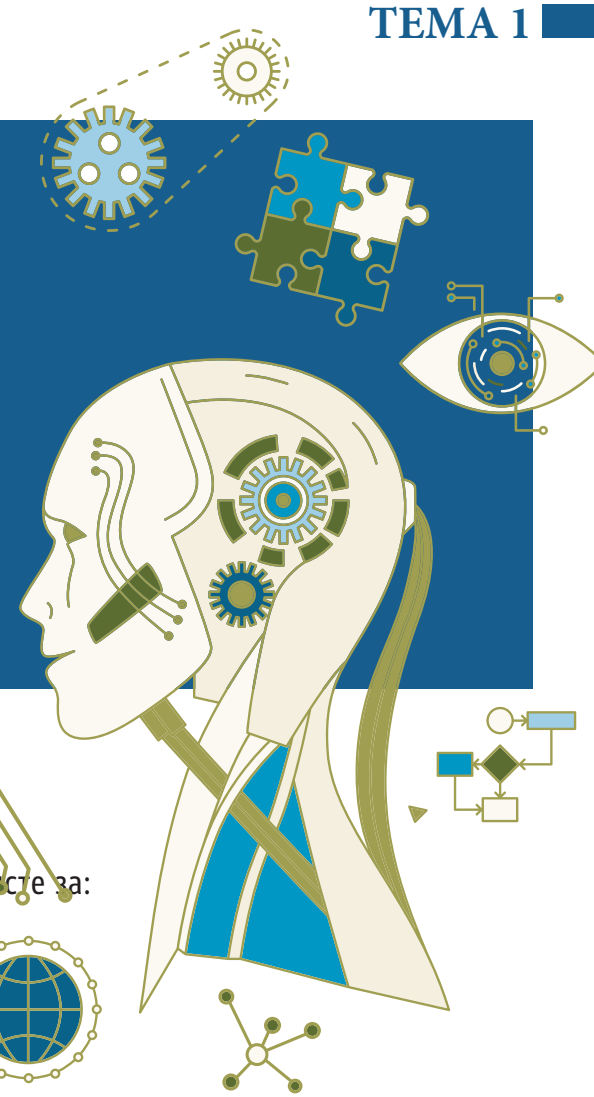
"Како може генеративна вештачка интелигенција да помогне у учењу и образовању?"

ЗАПАМТИ ШТА СИ НАУЧИО

Велики језички модели представљају савремене системе вештачке интелигенције који обрађују и стварају текст. Генеративна вештачка интелигенција може да ствара различите врсте дигиталних садржаја. Ове технологије примењују се у образовању, науци, уметности, пословању и многим другим областима. Створене садржаје треба критички анализирати и проверавати. При коришћењу генеративне вештачке интелигенције треба поштовати ауторска права и приватност. Генеративна вештачка интелигенција представља помоћни алат који допуњује, а не замењује људско знање и одговорност.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Развој генеративне вештачке интелигенције наставља се великом брзином и омогућава стварање све сложенијих дигиталних садржаја. Осим текстова, савремени системи могу да стварају слике, музику, видео-записе и рачунарске програме. У будућности се очекује да њихова примена буде још шира у образовању, науци, медицини и различитим привредним делатностима. Истовремено, развијају се правила и препоруке за њихово безбедно, етичко и одговорно коришћење, с циљем да технологија допринесе развоју друштва и побољшању квалитета живота.



ПРОВЕРИ СВОЈЕ ЗНАЊЕ

1. Заокружи тачан одговор.

Заокружи тачан одговор. Велики језички модели се користе за:

а) производњу компјутера;

б) обраду и креирање текстуалних садржаја;

в) производња електричне енергије;

г) поправка мобилних телефона.

2. Тачно или нетачно.

Велики језички модели и генеративна вештачка интелигенција

3. Допуни реченицу.

Савремени системи који могу да стварају нове дигиталне садржаје називају се _____ _____.

4. Објасни појмове.

Појам	Објашњење
Велики језички модел	_____
Генеративна ВИ	_____
Одговорно коришћење	_____

СИТУАЦИОНИ ЗАДАТАК

Ученик користи генеративну вештачку интелигенцију за припрему школског пројекта. Објасни које кораке треба да предузме како би обезбедио тачност информација и одговорно коришћење технологије.

ПРОГРАМИРАЊЕ У C++

Програмирање представља процес стварања компјутерских програма који омогућавају решавање различитих проблема и аутоматизацију задатака. Помоћу програмских језика задају се упутства према којима компјутер обрађује податке, извршава прорачуне, доноси одлуке и обавља различите активности. Данас програмирање има широку примену у многим областима савременог друштва, као што су образовање, наука, индустрија, медицина, комуникације и забава, и представља једну од основних вештина дигиталног доба.

У овој тематској целини обрађују се основни принципи програмирања помоћу програмског језика C++. Посебна пажња посвећује се начинима представљања података у компјутеру, изради алгоритама и програма, примени функција и коришћењу једнодимензионалних низова при решавању програмских задатака. Кроз практичне примере и активности развија се алгоритамско размишљање и стичу се основна знања и вештине за анализу и израду компјутерских програма.

НОВИ ПОЈМОВИ

- бинарни бројевни систем;
- оперативни систем
- октални бројевни систем;
- хексадекадни бројевни систем;
- функција
- параметар;
- библиотека;
- једнодимензионални низ;
- индекс елемента.

ВЕЋ ЗНАШ ДА...

- користиш компјутер и различите апликације;
- уносиш и обрађујеш податке;
- примењујеш логичко размишљање при решавању проблема;
- разликујеш различите врсте информација;
- следиш и извршаваш задате инструкције;
- анализираш једноставне свакодневне ситуације;
- користиш основне информатичке технологије.

TEMA 2

2.1

ДИГИТАЛНО ПРЕДСТАВЉАЊЕ ПОДАТАКА У КОМПЈУТЕРУ

ВЕЋ ЗНАШ ДА...

- објасниш шта представља компјутерски систем;
- наведеш примере дигиталних уређаја;
- разликујеш хардвер и софтвер;
- користиш основне типове података у C++;
- израдиш једноставне програме са променљивим и операторима.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објасниш како компјутер представља податке;
- разликујеш бит и бајт;
- објасниш како се представљају бројеви, текст, слике и звук;
- препознајеш различите типове података;
- анализираш како се користи меморија за чување информација.

У савременом дигиталном свету велики број информација непрестано се обрађује, складишти и преноси посредством различитих компјутерских система. Текстови који се пишу, фотографије које се снимају, музика која се слуша и видео-снимци који се приказују не чувају се у компјутеру у свом природном облику, већ се претходно претварају у дигиталне податке. Због тога је потребно разумети начин на који компјутер представља информације и како се оне чувају у меморији.

Сваки податак у рачунару представља се комбинацијама бинарних вредности. Иако корисници виде слова, бројеве, слике и чују звук, у позадини су све информације представљене нулама и јединицама. Такав начин представљања омогућава електронским компонентама рачунара да лако обрађују податке. Разумевање дигиталног представљања података представља основу за даље изучавање програмирања и информатике.

Компјутерски системи раде са електронским сигналима који могу имати два стања. Због тога се у компјутерима користи бинарни начин представљања података. Основна јединица информације назива се бит. Бит може имати само две вредности – 0 или 1. Иако један бит самостално носи веома малу количину информација, комбиновањем више битова могу се представити сложени подаци. Осам битова чини бајт, који представља основну меру величине података. У свакодневној употреби често се користе и веће мерне јединице, као што су килобајт, мегабајт, гигабајт и терабајт. Са развојем технологије количина података који се обрађују непрестано се повећава, па савремени уређаји располажу меморијама у којима се могу складиштити огромне количине информација. Подаци који се чувају у меморији морају бити организовани на начин

Пример 1: Претварање мерних јединица за количину података

main.cpp

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

int main() {
    int megabajti;

    cout << "Unesi broj megabajta : ";
    cin >> megabajti;

    int kilobajti = megabajti * 1024;

    cout << "U kilobajtima ima : " << kilobajti << endl;

    return 0;
}
```

Поред бројева, у компјутерима се представљају и текстуални подаци. Свако слово, цифра или симбол добија одређени бинарни кôд посредством посебних стандарда за кодирање. Један од најпознатијих стандарда јесте ASCII кôд, а данас се широко користи и стандард Unicode, који омогућава представљање великог броја светских језика и симбола. Захваљујући овим стандардима, текст се може правилно приказивати на различитим уређајима и оперативним системима. Поред текста, дигитално се представљају и слике. Свака дигитална слика састоји се од великог броја малих тачака које се називају пиксели. Сваки пиксел садржи информацију о боји, а комбиновањем хиљада или милиона пиксела добија се целовита слика. Што је број пиксела већи, то је квалитет слике бољи. На сличан начин представља се и звук, при чему се звучни таласи претварају у дигиталне вредности које компјутер може да обрађује. На тај начин различите врсте информација могу бити сачуване, копиране и обрађене у дигиталном облику.

Пример 2: Приказ ASCII вредности знака

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

int main() {
    char znak;

    cout << "Unesi znak : ";
    cin >> znak;

    cout << "ASCII vrednost je   : " << (int)znak << endl;

    return 0;
}
```

У програмским језицима подаци се организују помоћу различитих типова података. Сваки тип података заузима одређени простор у меморији и служи за чување различитих врста информација. На пример, тип int користи се за целе бројеве, double за реалне бројеве, char за знакове, а bool за логичке вредности. Избор одговарајућег типа података веома је важан јер утиче на тачност података и количину меморије која ће се користити. Ако се за једноставну вредност изабере тип који заузима много меморије, програм може непотребно да троши ресурсе. Са друге стране, ако се изабере тип са малим опсегом вредности, може доћи до грешака при израчунавању. Због тога програмери пажљиво бирају типове података у складу са потребама програма. Разумевање начина на који су подаци организовани у меморији представља важан корак ка стварању ефикаснијих програма.

Пример 3: Коришћење различитих типова података

*main.cpp

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

int main() {
    int uzrast = 16;
    double prosek = 4.75;
    char odeljenje = 'A';
    bool redenUcenik = true;

    cout << "Uzrast: " << uzrast << endl;
    cout << "Prosek: " << prosek << endl;
    cout << "Odeljenje: " << odeljenje << endl;
    cout << "Redovan učenik: " << redovanUcenik << endl;

    return 0;
}
```

Меморија компјутера представља простор у којем се складиште програми и подаци. Свака променљива која се користи у програму добија своје место у меморији. Компјутер аутоматски додељује адресу сваком податку, а програм приступа тим подацима преко имена променљивих. Начин на који је меморија организована има велики значај за брзину и ефикасност програма. Савремени компјутери располажу различитим типовима меморије, као што су RAM меморија и трајна меморија за складиштење података. RAM меморија служи за привремено чување информација док програм ради, док трајна меморија чува податке и након искључивања компјутера. Количина меморије коју програм користи зависи од броја и типа података који се обрађују. Зато разумевање дигиталног представљања и организације података представља важну основу за даље изучавање алгоритама, функција и структура података.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља бит?
2. Колико бита садржи један бајт?
3. Како се представљају текстуални подаци у компјутеру?
4. Која је улога типа double у C++?

ЗАДАЦИ

1. Напиши програм који ће израчунати колико ме-бајти имају задати килобајти.
2. Напиши програм који ће приказивати ASCII вредност унетог знака.
3. Напиши програм који ће чувати и приказивати податке о ученику/ци коришћењем различитих типова података.
4. Напиши програм који ће израчунавати колико габајти имају задати гигабајти.
5. Напиши програм који ће приказивати да ли је вредност унетог знака. унети знак велико или мало слово.

ЗАПАМТИ ШТА СИ НАУЧИО

- Компјутери представљају податке у бинарном облику.
- Основна јединица за информацију је бит.
- Осам бита формирају један бајт.
- Текст, слике и звук се представљају бинарним вредностима.
- Различити типови података захватају различит простор у меморији.
- Меморија омогућава чување и обраду података.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим компјутерским системима се користе специјалне технике за компресију података. Компресијом се смањује величина датотека без значајног губитка квалитета. Захваљујући томе, фотографије, музика и видео-записи могу лакше да се преносе путем интернета и заузимају мање простора у меморији. Формати као што су JPG, MP3 и MP4 користе различите алгоритме компресије како би се смањила количина података коју треба складиштити.

2.2

БРОЈНИ СИСТЕМИ И КОНВЕРЗИЈЕ

ВЕЋ ЗНАШ ДА...

- објасниш како компјутер представља податке;
- разликујеш бит и бајт;
- наведеш примере типова података;
- објасниш како се представља текст, слике и звук;
- користиш основне типове података у C++.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Колико бита имају 4 бајта?
2. Који тип података у C++ се користи за чување знака?
3. Шта представља пиксел?
4. Напиши програм који ће израчунавати колико килобајти имају задати мегабајти.
5. Објасни зашто се подаци у компјутеру представљају дигитално.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- разликујеш бинарни, декадни, октални и хексадекадни бројни систем;
- објасниш шта представља основу бројног система;
- вршиш ручне конверзије између бројних система;
- анализираш позиционе вредности цифара;
- примењујеш конверзије у програмским задацима.

У свакодневном животу најчешће се користи декадни бројевни систем, који се састоји од десет цифара – од 0 до 9. Међутим, компјутери не функционишу на исти начин. Будући да електронска кола могу поуздано да препознају само два стања, у компјутерским системима користи се бинарни бројевни систем. Због тога се све информације које компјутер обрађује представљају комбинацијама нула и јединица.

Поред бинарног система, у информатици се често користе и октални и хексадекадни бројевни систем. Ови системи омогућавају једноставније представљање дугачких бинарних записа и бољу организацију података. Разумевање бројевних система представља важан део информатике јер омогућава разумевање начина на који компјутер обрађује и чува информације.

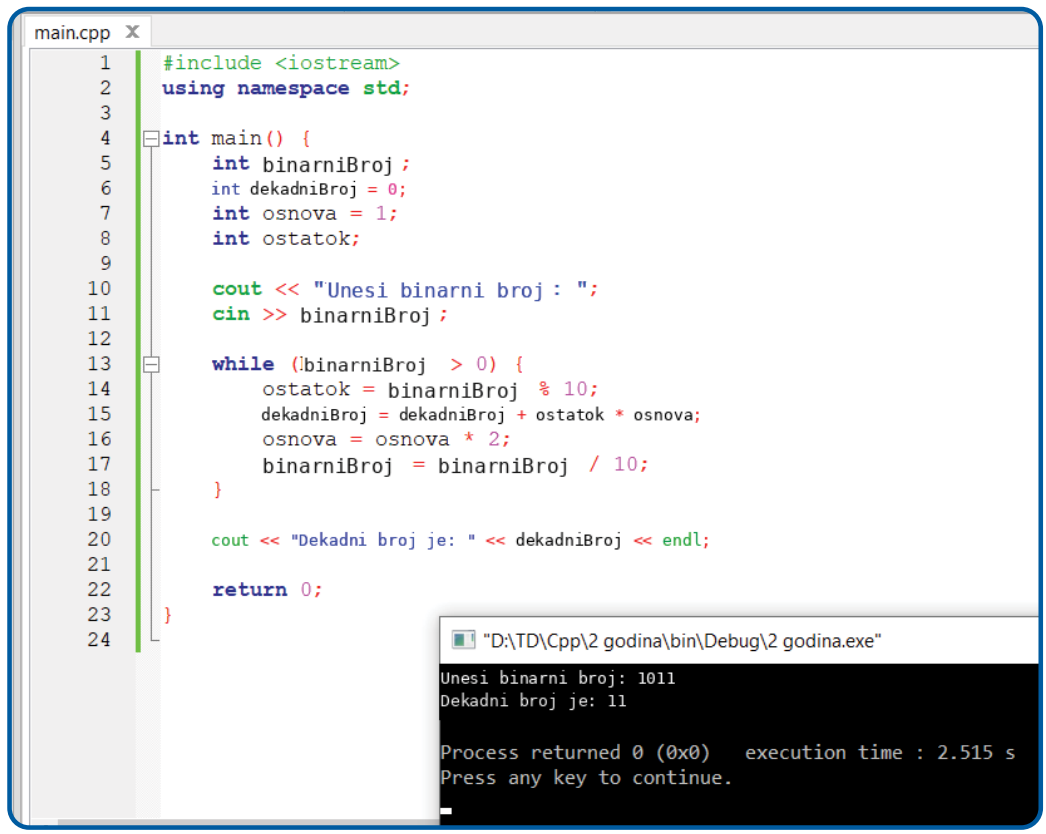
Бројевни систем представља начин записивања бројева коришћењем одређеног броја симбола. Број симбола који се користе назива се основа бројевног система. У декадном бројевном систему основа је 10, јер се користи десет цифара. У бинарном систему основа је 2, па се користе само цифре 0 и 1. Свака цифра у броју има позиционо вредност која зависи од њеног положаја и основе система. У декадном систему позиције представљају степене броја 10, док у бинарном систему представљају степене броја 2. Ако бинарни број има четири цифре, користе се степени броја 2 од 0 до 3. Ако бинарни број има 15 цифара, користе се степени броја 2 од 0 до 14. Сабирају се само вредности на позицијама на којима се у бинарном броју налази цифра 1. На пример, бинарни број 1011_2 израчунава се на следећи начин:

Бинарна цифра	**1**	**0**	**1**	**1**
Степен броја 2	2^3	2^2	2^1	2^0
Вредност	8	0	2	1

Сабирањем вредности добија се:
$8 + 0 + 2 + 1 = 11_{10}$

На тај начин врши се претварање из бинарног у декадни систем. Бинарни систем представља основу функционисања компјутера јер електронске компоненте лако могу да представе два стања – укључено и искључено. Због тога су сви подаци и инструкције у рачунару представљени у бинарном облику.

Пример 1: Претварање бинарног броја у декадни



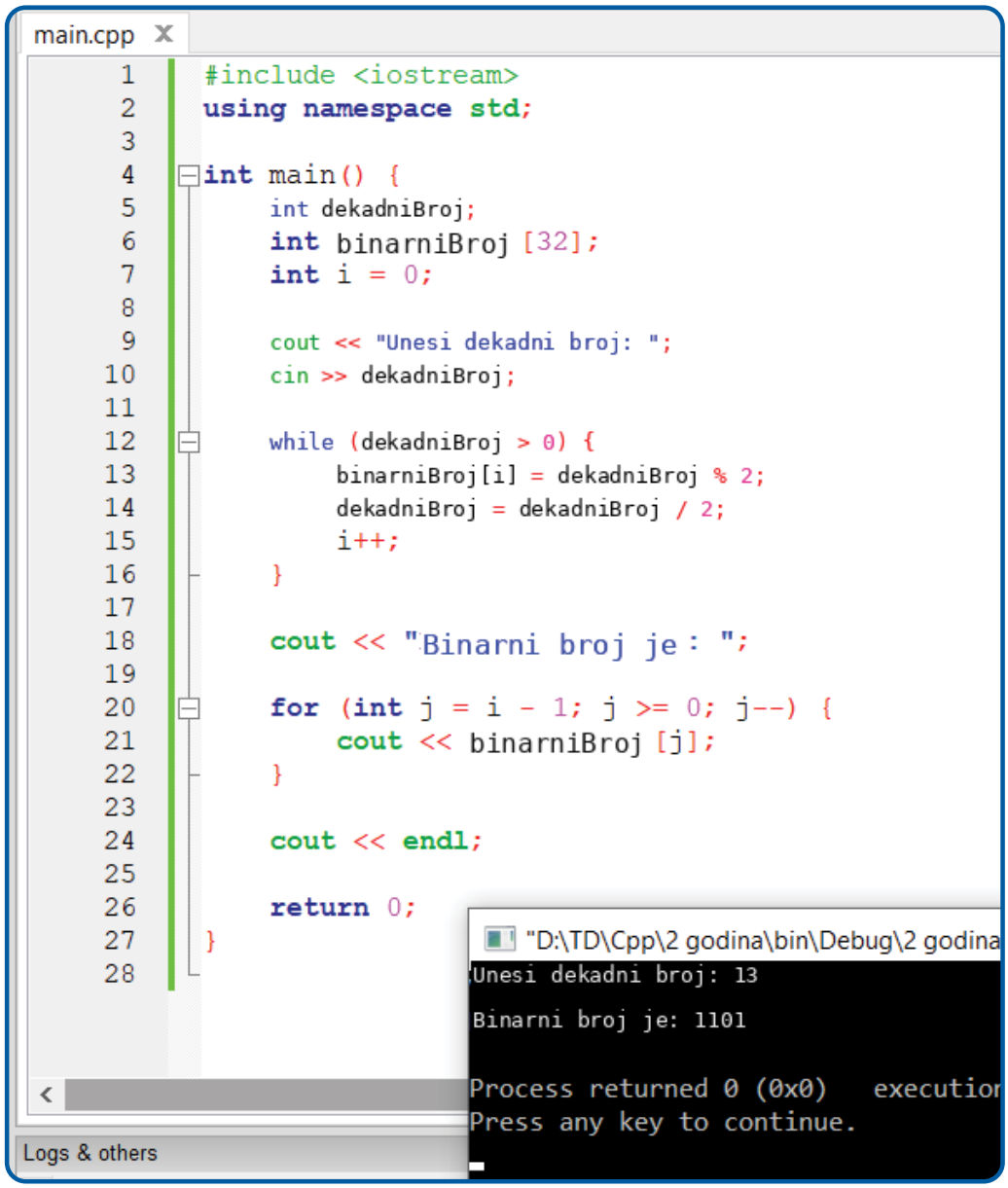
Претварање из декадног у бинарни систем врши се узастопним дељењем броја са 2. При сваком дељењу записује се остатак, а бинарни број се добија читањем остатака обрнутим редоследом. На пример, ако број 13_{10} треба претворити у бинарни систем, поступак се изводи на следећи начин:

Дељење	Количник	Остатак
13 : 2	6	1
6 : 2	3	0
3 : 2	1	1
1 : 2	0	1

Читањем остатака одозго нагоре добија се:
$13_{10} = 1101_2$

Овај поступак представља једно од најважнијих претварања у информатици. Захваљујући оваквим претварањима, програмери боље разумеју како компјутер обрађује бројеве у меморији. У програмским језицима често се користе алгоритми који аутоматски извршавају ове поступке.

Пример 2: Претварање декадног броја у бинарни



Поред бинарног система, у информатици се користе и октални и хексадекадни бројевни систем. Октални систем има основу 8 и користи цифре од 0 до 7, док хексадекадни систем има основу 16 и користи цифре од 0 до 9 и слова A, B, C, D, E и F. Ови системи омогућавају да се дугачки бинарни бројеви запишу краће и прегледније. При претварању из бинарног у октални систем бинарне цифре групишу се по три, док се при претварању у хексадекадни систем групишу по четири. Груписање се врши здесна налево. Ако на левој страни остану једна или две цифре, допуњују се нулама слева.

Пример за конверзију у октално систему:

10110110_2

Груписање по 3 цифре:

010 110 110

$010_2 = 2_8$

$110_2 = 6_8$

$110_2 = 6_8$

Значи:

$10110110_2 = 266_8$

Пример за конверзију у хексадекадни систем:

10110110_2

Груписање по 4 цифре:

1011 0110

$1011_2 = B_{16}$

$0110_2 = 6_{16}$

Значи:

$10110110_2 = B6_{16}$

Хексадекадни систем има широку примену у програмирању, записивању меморијских адреса и представљању боја у веб-дизајну. На пример, ознака боје #FF0000 представља црвену боју у RGB систему.

Пример 3: Приказ октално и хексадекадног записа

main.cpp

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

int main() {
    int broj;

    cout << "Unesi dekadni broj: ";
    cin >> broj;

    cout << "Oktalni zapis: " << oct << broj << end

    cout << "Heksadekadni zapis: " << hex << broj << endl;

    return 0;
}
```

Различити бројевни системи представљају важан део информатике и компјутерске архитектуре. Иако савремени компјутери аутоматски извршавају конверзије, програмери морају да разумеју како оне функционишу. Такво знање омогућава лакше разумевање организације меморије, начина рада процесора и обраде података. Осим тога, бројевни системи користе се и у компјутерским мрежама, криптографији, дигиталној електроници и развоју софтвера. Због тога њихово изучавање представља основу за даље изучавање програмирања и савремених информатичких технологија.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља основа бројевног система?
2. Које цифре се користе у бинарном систему?
3. Како се врши конверзија из бинарног у декадни систем?
4. Колика је основа хексадекадног система?
5. Зашто се у информатици користи хексадекадни запис?

ЗАДАЦИ

1. Ручно претвори број 10101_2 у декадни систем.
2. Ручно претвори број 25_{10} у бинарни
3. Ручно претвори број 11101010_2 у октални и у хексадецимални систем.
4. Напиши програм који ће претварати бинарни број у декадни.
5. Напиши програм који ће приказивати октални и хексадекадни запис унетог броја.

ЗАПАМТИ ШТА СИ НАУЧИО

- Бројевни систем представља начин записивања бројева.
- Бинарни систем има основу 2.
- Октални систем има основу 8.
- Хексадекадни систем има основу 16.
- Конверзија из бинарног у декадни систем врши се преко позиционих вредности.
- Конверзија из декадног у бинарни систем врши се узастопним дељењем са 2.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У компјутерским процесорима и меморијским системима често се користе хексадекадне вредности, јер омогућавају краће и прегледније представљање бинарних информација. У програмским језицима меморијске адресе често се приказују у хексадекадном облику. Хексадекадни систем користи се и за представљање боја у веб-технологијама, где се свака боја представља комбинацијом шест хексадекадних знакова.

2.3

ПРЕДСТАВЉАЊЕ ЦЕЛИХ БРОЈЕВА И ОПСЕГ ПОДАТАКА

ВЕЋ ЗНАШ ДА...

- разликујеш бинарни, декадни, октални и хексадекадни бројни систем;
- вршиш конверзије између бројних система;
- објасниш шта представља основа бројног система;
- анализираш позиционе вредности;
- користиш основне операторе у C++.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Колика је основа окталног бројног система?
2. Ручно претвори број 1111_2 у декадни систем.
3. Ручно претвори број 45_{10} у бинарни систем.
4. Шта је хексадекадни запис бинарног броја 10101111_2?
5. Напиши програм који ће приказивати октални запис унетог броја.

О СОДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објасниш како се представљају цели бројеви у компјутеру;
- разликујеш позитивне и негативне бројеве у меморији;
- објасниш шта представља опсег података;
- анализираш overflow појаву;
- користиш различите типове целих бројева у C++.

У свакодневној комуникацији бројеви се најчешће користе без размишљања о начину на који се чувају у компјутеру. Међутим, компјутерски системи располажу ограниченом меморијом, па сваки број мора бити представљен у одређеном формату и ограниченим бројем битова. Начин представљања бројева у меморији има велики утицај на тачност израчунавања, брзину обраде и могућност складиштења информација.

У програмским језицима постоје различити типови података за представљање целих бројева. Сваки тип заузима различит простор у меморији и омогућава представљање различитог опсега вредности. Ако програм покуша да сачува број који је већи од дозвољеног опсега, може доћи до грешке која се назива прекорачење (енгл. overflow). Због тога програмери морају пажљиво да бирају типове података које ће користити у програмима.

Компјутери представљају све бројеве у бинарном облику. Пошто је меморија ограничена, за сваки број резервише се одређени број битова. На пример, ако се за један број користи осам битова, њима се може представити ограничен број различитих вредности. Код позитивних бројева сваки бит учествује у формирању броја посредством степена броја 2. На пример, број 13 у бинарном облику представља се као 00001101_2.

У меморији сваки бит има тачно одређено место и значење. Што се више битова користи, то се може представити већи опсег бројева. У програмским језицима најчешће се користе типови као што су short, int, long long и други, при чему сваки тип користи различиту количину меморије. Због тога избор типа података непосредно утиче на количину меморије коју програм користи. У савременим компјутерским системима правилан избор типова података представља важан део оптимизације програма.

Пример 1: Приказ величине типова података

main.cpp

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

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

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

Поред позитивних бројева, у компјутерима се морају представити и негативни бројеви. У ту сврху користи се посебан начин записивања који се назива комплемент двојке (енгл. two's complement). Код овог начина представљања крајњи леви бит обично означава знак броја. Захваљујући таквом систему, процесор може једноставно да сабира и одузима и позитивне и негативне бројеве. На пример, ако се за представљање бројева користи осам битова, могу се представити вредности од –128 до 127. То значи да постоји ограничење највећег и најмањег броја који се може сачувати у меморији. Ако програм покуша да сачува број изван дозвољеног опсега, резултат може бити нетачан. Оваква ограничења често се не примећују код малих програма, али у великим системима могу да изазову озбиљне проблеме. Због тога разумевање опсега података представља важан део програмирања

Пример 2: Приказ на минимална и максимална вредност

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

int main() {

    cout << "Minimalna vrednost za int: " << INT_MIN << endl;
    cout << "Maksimalna vrednost za int: " << INT_MAX << endl;

    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.exe"
Minimalna vrednost za int: -2147483648
Maksimalna vrednost za int: 2147483647
```

Опсег података представља интервал вредности које се могу представити одређеним типом података. Што је већи број битова који се користе, то је већи и опсег вредности. На пример, тип short заузима мање меморије од типа long long, али може да чува мање бројеве. У програмима се често мора пажљиво проценити колики ће опсег бити потребан. Ако се за велика израчунавања користи тип са малим опсегом, може доћи до прекорачења (енгл. overflow). Прекорачење представља ситуацију у којој резултат премашује највећу вредност која се може сачувати. У таквим случајевима програм може да прикаже неочекиван или погрешан резултат. Овакве грешке посебно су опасне у финансијским, научним и безбедносним системима. Због тога програмери морају да разумеју ограничења типова података и правилно их бирају у складу са потребама програма.

Пример 3: Overflow код целог броја

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

int main() {

    int broj = INT_MAX;

    cout << "Pre uvećanja : " << broj << endl;

    broj++;

    cout << "Posle uvećanja: " << broj << endl;

    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.exe"
Pre uvećanja e: 2147483647
Posle uvećanja e: -2147483648
```

Савремени програмски језици омогућавају коришћење различитих типова података како би програми били ефикаснији и прецизнији. У неким ситуацијама потребно је чувати веома велике бројеве, док је у другим ситуацијама важно уштедети меморију. Због тога програмери бирају различите типове података у складу

са конкретним проблемом. Поред тога, правилно разумевање опсега бројева омогућава лакше откривање грешака и бољу анализу програма. Познавање начина на који су бројеви представљени у меморији представља основу за даље изучавање алгоритама, структура података и оптимизације програма.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Зашто компјутери користе бинарно представљање бројева?
2. Шта представља опсег података?
3. Шта је overflow?
4. Који тип података може да чува веће вредности – int или long long?
5. Зашто је важно правилно да се изабере тип података?

ЗАДАЦИ

1. Напиши програм који ће приказивати величину типова char, int и double.
2. Напиши програм који ће приказивати минималну и максималну вредност типа int.
3. Напиши програм који ће демонстрирати overflow код типа short.
4. Напиши програм који ће уносити два велика броја и сабирати их коришћењем типа long long.
5. Истражи колики опсег има тип unsigned int и упореди га са типом int.

ЗАПАМТИ ШТА СИ НАУЧИО

- Компјутери представљају бројеве у бинарном облику.
- Сваки тип података користи различити број бајти.
- Опсег представља најмању и највећу вредност која може да се сачува.
- Overflow се појављује када вредност превазилази дозвољени опсег.
- Правилни избор типа података је важан за тачност и ефикасност програма.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим програмским језицима постоје и типови података са веома великим опсегом, који се користе у научним симулацијама, криптографији и обради великих база података. Неке програмске библиотеке омогућавају рад са бројевима који садрже стотине или хиљаде цифара. Такви бројеви не могу се непосредно представити стандардним типовима података, па се за њихову обраду користе посебни алгоритми и структуре.

2.4

ОПТИМИЗАЦИЈА И ЕФИКАСНОСТ АЛГОРИТАМА

ВЕЋ ЗНАШ ДА...

- објасниш шта представља алгоритам;
- користиш if и switch структуре;
- користиш for и while петље;
- анализираш временску сложеност;
- решаваш текстуалне задатке у C++.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта представља алгоритам?
2. Напиши пример за for петља која се понавља 10 пута.
3. Која је разлика између for и while петљи?
4. Шта представља временска сложеност?
5. Напиши програма који ће израчунавати квадрат унетог броја.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- анализираш ефикасност алгоритма;
- упоређујеш различита решења за исти проблем;
- препознајеш непотребне операције;
- оптимизираш алгоритме;
- анализираш утицај на величину података на брзину програма.

У програмирању често постоји више различитих начина за решавање истог проблема. Иако различита решења могу давати исти резултат, она не раде увек истом брзином и не користе исту количину меморије. Код малих програма такве разлике често нису приметне, али при обради великих количина података могу значајно да утичу на рад система.

Због тога савремено програмирање није усмерено само на тачност програма већ и на његову ефикасност. Програмери непрестано анализирају колико операција алгоритам извршава, колико меморије користи и да

ли постоји бољи начин за решавање проблема. Тај процес назива се оптимизација алгоритама и представља важан део развоја савременог софтвера.

Ефикасност алгоритма најчешће се анализира на основу броја операција које се извршавају при решавању проблема. Што је више операција потребно, то ће бити потребно више времена за извршавање програма. На пример, ако треба израчунати збир бројева од 1 до n, може се користити петља која ће сабирати све бројеве један по један. Такав приступ је исправан, али ће се при веома великим вредностима n извршити огроман број операција. Са друге стране, исти проблем може се решити математичком формулом која одмах даје резултат. У том случају алгоритам извршава знатно мањи број операција и ради брже. Због тога програмери непрестано траже начине за смањење броја непотребних израчунавања. Овакав начин размишљања представља основу оптимизације алгоритама и један је од најважнијих делова савременог програмирања.

Пример 1: Поређење два алгоритма

main.cpp

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

int main() {

    int n;
    cin >> n;

    int zbir1 = 0;

    for (int i = 1; i <= n; i++) {
        zbir1 += i;
    }

    int zbir2 = n * (n + 1) / 2;

    cout << "Pomoću petlje: " << zbir1 << endl;
    cout << "Pomoću formule : " << zbir2 << endl;

    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godin...
500
So ciklus: 125250
Pomoću formule a: 125250

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

При анализи алгоритама веома је важно размотрити како се брзина програма мења са повећањем количине података. Алгоритам који ради брзо са 10 елемената може постати веома спор када обрађује милион елемената. Због тога програмери не анализирају само тренутну брзину већ и понашање алгоритма при великим количинама улазних података. У информатици се за описивање сложености често користе ознаке као што су O(n), O(n^2) и O(log n). Алгоритам сложености O(n^2) при великим количинама података најчешће постаје знатно спорији од алгоритма сложености O(n). На пример, угнежђене петље често изазивају значајно повећање броја операција. Због тога програмери пажљиво анализирају да ли се непотребна понављања могу избећи и да ли постоји боље решење. У великим рачунарским системима правилна оптимизација може значајно да побољша рад програма.

Пример 2: Неефикасно и поефикасно пребарување

*main.cpp

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

int main()
{
    int niza[5] = {2, 4, 6, 8, 10};
    int broj;
    bool pronadjen = false;
    cin >> broj;
    for (int i = 0; i < 5; i++) {

        if (niza[i] == broj) {
            pronadjen = true;
            break;
        }
    }
    if (pronadjen) {
        cout << "Broj je pronađen." << endl;
    }
    else {
        cout << "Broj nije pronađen." << endl;
    }
    return 0;
}

```

У претходном примеру користи се наредба break за прекидање петље одмах након проналажења броја. Без наредбе break, петља би наставила да проверава и непотребне елементе. Иако је разлика мала код низа са пет елемената, код великих низова то може значајно да утиче на брзину програма. Оваква мала побољшања често имају велико значење у професионалном програмирању.

Програмери непрестано анализирају како да смање број операција и ефикасније користе меморију. Тај процес не подразумева само „бржи програм“ већ и бољу организацију кода и лакше одржавање система.

Пример 3: Утицај на угњеждене циклусе

main.cpp

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

int main()
{
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cout << "* ";
        }
        cout << endl;
    }
    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.ex...
5
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

Process returned 0 (0x0)   execution time : 1.186 s
Press any key to continue.
```

У претходном примеру користе се угнежђене петље. Ако n има вредност 5, извршиће се 25 понављања. Ако n има вредност 1.000, извршиће се милион понављања. Ово показује да брзина алгоритма може значајно да се промени са повећањем количине улазних података. Због тога програмери морају пажљиво да размишљају када користе угњеждене циклусе и да траже ефикасније решење. У савременим системима који обрађују огромне количине података, оптимизација представља један од најважнијих фактора за успешно функционисање програма.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља оптимизација алгоритма?
2. Зашто могу два алгоритма да имају различиту брзину?
3. Како утичу угњеждене петље на сложеност?
4. Зашто је важно да се избегавају непотребне операције?
5. Како величина улазних података утиче на брзину програма?

ЗАДАЦИ

1. Напиши програм који израчунава колико је бројева од 1 до n дељиво са 7.
2. Напиши програм који ће уносити n бројеве и пронаћи највећи број.
3. Напиши програм са угнежденим петљама које ће штампати правоугаоник од знака #.
4. Анализирај колико провера ће се извршити ако програм проверава сваки елемент низа са 1000 елемената.
5. Напиши два различита решења за израчунавање збира од 1 до n и упореди које решење је ефикасније.

ЗАПАМТИ ШТА СИ НАУЧИО

- Различити алгоритми могу решавати исти проблем са различитом ефикасношћу.
- Оптимизација има за циљ да смањи број операција и потрошњу меморије.
- Угнежђени циклуси значајно повећавају сложеност.
- break може помоћи да се смањи број непотребних провера.
- Анализа алгоритама је важна при раду са великим количинама података.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим интернет услугама и системима вештачке интелигенције сваке секунде обрађују се милиони података. Ако алгоритми не би били довољно оптимизовани, сервери би радили знатно спорије и трошили огромну количину ресурса. Због тога се велики део посла професионалних програмера састоји од анализирања и унапређивања алгоритама. У појединим случајевима оптимизација алгоритма може убрзати програм стотинама или хиљадама пута.

2.5

КОМБИНОВАЊЕ УСЛОВА И ЦИКЛУСА

ВЕЋ ЗНАШ ДА...

- користиш if и switch структуре;
- користиш for и while циклусе;
- анализираш ефикасност алгоритама;
- користиш угнеждене циклусе;
- упоређујеш различита програмска решења.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта представља оптимизација алгоритма?
2. Зашто break може да побољша ефикасност?
3. Шта се догађа када се користе два угнеждена циклуса?
4. Напиши програм који ће израчунати збир од 1 до n.
5. Објасни зашто велики подаци траже ефикасније алгоритме.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- комбинујеш услове и циклусе у истом програму;
- решаваш сложеније текстуалне задатке;
- користиш угнеждене услове у циклусима;
- анализираш логику понављања и избора;
- конструишеш организованија програмска решења.

У програмирању се многи проблеми не могу решити само условима или само циклусима У стварним програмима најчешће је потребно комбиновати различите програмске структуре како би се добило потпуно решење. На пример, програм може више пута да уноси податке и при сваком уносу проверава да ли је испуњен одређени услов. Такве ситуације захтевају истовремену употребу циклуса и услова.

Комбиновање програмских структура представља важан корак ка развоју сложенијих алгоритама. Таквим приступом програми постају флексибилнији и могу да обрађују различите ситуације. Због тога програмери

морају пажљиво да анализирају где треба поставити услов, када поступак треба понављати и како правилно организовати логику програма.

Петље омогућавају понављање одређених инструкција, док услови омогућавају доношење одлука. Када се ове структуре комбинују, програм може истовремено да понавља операције и проверава различите услове.

На пример, ако треба пребројати само парне бројеве од 1 до n, петља ће пролазити кроз све бројеве, а услов ће проверавати да ли је сваки број паран. На тај начин услов се извршава унутар петље. Овакав начин рада представља једну од најчешћих техника у програмирању.

У савременим програмима често се обрађује велики број података, при чему се за сваки податак обављају различите провере. Због тога правилно комбиновање услова и петљи представља основу за решавање сложенијих проблема и стварање ефикасних алгоритама.

Пример 1: Пребројавање парних бројева

main.cpp

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

int main()
{
    int n;
    int brojac = 0;

    cout << "Unesi broj : ";
    cin >> n;

    for (int i = 1; i <= n; i++) {

        if (i % 2 == 0) {
            brojac++;
        }
    }

    cout << "Broj parnih brojeva : " << brojac << endl;

    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina....
Unesi broj: 10
Broj parnih brojeva: 5

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

У многим задацима потребно је угнездити услове у петље или петље у услове. Такве конструкције омогућавају решавање много сложенијих проблема. На пример, при анализи успеха ученика у одељењу може бити потребно уносити оцене све док се не унесе одређена вредност, а затим за сваку оцену проверити да ли је позитивна или негативна.

Овакви алгоритми захтевају пажљиво организовање логике јер погрешно постављен услов може да изазове бесконачну петљу или нетачан резултат. Због тога програмери често користе постепени приступ – најпре анализирају понављање, а затим додају услове. Такав начин рада омогућава лакше тестирање и бољу организацију кода.

Пример 2: Збир позитивних бројева

main.cpp

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

int main()
{
    int broj;
    int zbir = 0;

    for (int i = 1; i <= 5; i++) {

        cout << "Unesi broj : ";
        cin >> broj;

        if (broj > 0) {
            zbir += broj;
        }
    }

    cout << "Zbir pozitivnih brojeva je  : " << zbir << endl;

    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina....
Unesi broj j: 2
Unesi broj j: -3
Unesi broj j: 4
Unesi broj j: -5
Vnesi broj: 1
Zbir pozitivnih brojeva je      : 7

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

Logs & others

Комбиновање услова и петљи често се користи и при решавању текстуалних задатака. У таквим задацима потребно је анализирати различите ситуације и изабрати одговарајући поступак. На пример, програм може да проверава да ли је број прост, да ли је дељив другим бројем или да ли истовремено испуњава више услова.

У оваквим ситуацијама услови могу бити веома сложени и садржати више логичких оператора. Поред тога, циклуси се често користе за понављање провера над већим бројем података. Због тога овакви програми представљају добру основу за развој алгоритамског размишљања и логичке анализе. Што се више различитих структура комбинује, програми постају моћнији и способнији за решавање стварних проблема.

Пример 3: Провера простог броја

main.cpp

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

int main()
{
    int broj;
    bool prost = true;
    cout << "Unesi broj  ";
    cin >> broj;
    if (broj <= 1)
    {
        prost = false;
    }
    for (int i = 2; i < broj; i++)
    {
        if (broj % i == 0)
        {
            prost = false;
        }
    }
    if (prost)
    {
        cout << "Broj je prost ." << endl;
    }
    else
    {
        cout << "Broj nije prost  ." << endl;
    }
    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.
Unesi broj: 9
Broj nije prost.

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

У претходном примеру циклус редом проверава могуће делиоце броја, док услов проверава да ли постоји дељивост без остатка. Комбинација ових структура омогућава правилну анализу броја и добијање тачног резултата. Овакав приступ користи се у великом броју алгоритама у којима су потребни анализа података, провера услова и понављање поступака. Због тога комбиновање услова и циклуса представља један од најважнијих корака у развоју сложенијих програмских решења.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Зашто се у програмима комбинују услови и петље?
2. Шта представља угнежђени услов?
3. 3. Какву улогу има услов унутар петље?
4. Зашто је важно правилно да се организује логика програма?
5. Како петље помажу при обради веће количине података?

ЗАДАЦИ

1. Напиши програм који ће пребројавати непарне бројеве од 1 до n.
2. Напиши програм који ће уносити 10 бројева и израчунавати збир само негативних бројева.
3. Напиши програм који ће проверавати да ли је унети број савршен број.
4. Напиши програм који ће штампати све бројеве дељиве и са 3 и са 5 у одређеном интервалу.
5. Напиши програм који проверава колико унетих бројева је позитивно, а колико негативно.

ЗАПАМТИ ШТА СИ НАУЧИО

- Циклуси и услови најчешће се користе заједно.
- Услови омогућавају проверу различитих ситуација.
- Циклуси омогућавају понављање поступака.
- Комбиновање структура омогућава решавање сложенијих проблема.
- Правилна организација логике је веома важна у програмирању.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим компјутерским системима огроман број алгоритама заснива се на комбиновању услова и понављања. Интернет претраживачи, системи за препоруке и вештачка интелигенција непрестано обрађују огромне количине података коришћењем различитих услова и петљи. У неким програмима извршавају се милијарде провера у секунди. Због тога правилна организација услова и петљи има огроман утицај на брзину и ефикасност савремених система.

2.6

РЕШАВАЊЕ КОМПЛЕКСНИХ ТЕКСТУАЛНИХ ЗАДАТАКА

ВЕЋ ЗНАШ ДА...

- комбиновање услова и циклуса
- користиш угнеждене циклусе;
- анализираш ефикасност алгоритама;
- оптимизираш једноставна програмска решења;
- организујеш логику у програму.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ:

1. Шта представља оптимизацију алгоритама?
2. Зашто угнежђени циклуси могу значајно да успоре програм?
3. Напиши програм који ће пребројавати колико бројева од 1 до n је дељиво са 5.
4. Шта представља непотребну операцију у алгоритму?
5. Објасни зашто разна решења могу да имају разну ефикасност.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- анализираш комплексне текстуалне задатке;
- издвајаш потребне податке и услове;
- конструишеш алгоритме у више корака;
- комбинујеш више програмских техника;
- тестираш и побољшаваш програмска решења.

У програмирању се многи задаци не решавају једноставним израчунавањем или коришћењем само једне програмске структуре. Често је потребно истовремено анализирати више услова, обрађивати велики број података и организовати више корака у оквиру једног програма. Овакви задаци називају се сложени текстуални задаци јер захтевају детаљнију анализу и пажљивију организацију алгоритма.

При решавању сложених проблема веома је важно најпре пажљиво прочитати задатак и издвојити кључне информације. Потребно је одредити улазне податке, услове које треба проверавати и резултат који треба добити. Затим се проблем постепено дели на мање кораке који се лакше могу реализовати у програму. Такав приступ омогућава лакше откривање грешака и стварање организованијих решења.

При анализи сложеног текстуалног задатка први корак представља правилно разумевање проблема. У многим случајевима задатак садржи више услова који морају истовремено бити испуњени. Ако програмер пажљиво не анализира све захтеве, програм може давати нетачне резултате. Због тога се често користи постепени приступ, при којем се проблем дели на више подзадатака. На пример, ако треба анализирати резултате тестирања ученика, најпре се може пребројати колико је ученика положило тест, затим израчунати просек, а на крају приказати статистички подаци. Таквом поделом алгоритам постаје јаснији и лакши за реализацију. У савременом програмирању овакав начин размишљања представља основу за решавање сложених проблема и развој већих софтверских система.

Пример 1: Анализа оцена

main.cpp

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

int main()
{
    int n;
    int ocena ;
    int polozili = 0;

    cout << "Koliko ocena će se uneti? ";
    cin >> n;

    for (int i = 1; i <= n; i++)
        {
        cin >> ocena ;
        if (ocena >= 2)
        {
            polozili++;
        }
    }
    cout << "Broj učenika koji su položili : " << polozili << endl;
    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina....
Koliko ocena će se uneti? 5
2
1
3
5
4
Broj učenika koji su položili: 4
```

Сложени задаци често захтевају комбиновање више програмских техника у истом програму. У једном задатку могу се истовремено користити петље, услови, бројачи и различита израчунавања. Због тога је веома важно да програм буде добро организован и логички подељен.

Програмери често користе додатне променљиве за праћење различитих стања у програму. На пример, при анализи бројева могу се истовремено пратити највећа вредност, најмања вредност и број позитивних бројева. Овакви задаци захтевају пажљиво планирање алгоритма пре почетка програмирања. Ако логика није правилно

организована, програм може постати тежак за разумевање и одржавање. Зато, постепено конструисање решења представља веома важан део програмирања.

Пример 2: Проналажење највеће и најмање вредности

main.cpp

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

int main()
{
    int n;
    int broj;
    cout << "Koliko brojeva će se uneti     ? ";
    cin >> n;
    cout << "Koliko brojeva će se uneti? ";
    int maksimum = broj;
    int minimum = broj;
    for (int i = 2; i <= n; i++)
        {
        cin >> broj;
        if (broj > maksimum)
        {
            maksimum = broj;
        }
        if (broj < minimum)
        {
            minimum = broj;
        }
    }
    cout << "Maksimum: " << maksimum << endl;
    cout << "Minimum: " << minimum << endl;
    return 0;
}
```

```
"D:\TD\Cpp\2 godina\bin\Debug\2 godina.exe"
Koliko brojeva će se uneti   5
2 4 -2 8 102
Maksimum: 102
Minimum: -2
```

При решавању сложенијих задатака веома је важно тестирати алгоритам са различитим улазним подацима. Неки програми правилно раде са малим вредностима, али дају грешке при већим или неочекиваним подацима. Због тога програмери непрестано тестирају различите ситуације како би проверили да ли је програм стабилан и тачан.

Посебно је важно анализирати граничне случајеве, као што су унос нуле, негативних бројева или веома великих вредности. Оваквим тестирањем лакше се откривају логичке грешке и проблеми у алгоритму. У професионалном програмирању тестирање представља обавезан део развоја сваког програма.

Пример 3: Анализа позитивних и негативних бројева

main.cpp

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

int main()
{
    int n;
    int broj;
    int pozitivni = 0;
    int negativni = 0;
    cout << "Koliko brojeva će se uneti     ? ";
    cout << "Koliko brojeva će se uneti? ";
        {
        cin >> broj;
        if (broj >= 0)
        {
            pozitivni++;
        }
        else
        {
            negativni++;
        }
    }
    cout << "Pozitivni: " << pozitivni << endl;
    cout << "Negativni: " << negativni << endl;
    return 0;
}
```

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

Сложени текстуални задаци представљају важан део развоја алгоритамског размишљања. При њиховом решавању није довољно само познавати синтаксу програмског језика, већ је потребно пажљиво анализирати проблем и правилно организовати кораке. Што задаци постају сложенији, то се више развија способност логичког размишљања, анализе и планирања решења. Због тога овакви задаци представљају важан корак ка развоју озбиљнијих програмерских вештина.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља комплексни текстуални задатак?
2. Зашто је важно да се задатак подели на мање кораке?
3. Зашто програми треба да се тестирају различитим подацима?
4. Какву улогу имају бројачи у програмима?
5. Зашто је важна организација алгоритма?

ЗАДАЦИ

1. Напиши програм који ће уносити n бројева и израчунати њихов просек.
2. Напиши програм који ће пребројати колико је унетих бројева дељиво са 3.
3. Напиши програм који ће пронаћи други највећи број међу унетим вредностима.
4. Напиши програм који ће проверити колико је унетих оцена одлично.
5. Напиши програм који ће анализирати колико је унетих бројева парно, а колико непарно.

ЗАПАМТИ ШТА СИ НАУЧИО

- Комплексни задаци траже пажљиву анализу.
- Проблем се најчешће дели на више мањих корака.
- У сложеним задацима комбинује се више програмских техника.
- Тестирање је важан део развоја програма.
- Добра организација алгоритма омогућава јаснија и ефикаснија решења.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим софтверским системима програмери ретко решавају једноставне задатке. Велики број савремених апликација обрађује огромне количине података и истовремено проверава велики број услова. Банкарски системи, интернет продавнице и друштвене мреже користе сложене алгоритме који непрекидно анализирају информације и доносе одлуке. Због тога способност решавања сложених текстуалних задатака представља једну од најважнијих вештина у савременом програмирању.

2.7

МОДУЛАРНО ПРОГРАМИРАЊЕ И КОНЦЕПТ "ЗАВАДИ ПА ВЛАДАЈ"

ВЕЋ ЗНАШ ДА...

- користиш условне структуре у C++;
- користиш петље за понављање операција;
- комбинујеш услове и циклусе у истом програму;
- анализираш ефикасност алгоритама;
- решаваш сложеније програмске задатке.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ:

1. Шта представља оптимизацију алгоритама?
2. Зашто је важно да се смањи број непотребних операција у програму?
3. Напиши програм који ће пребројавати бројеве од 1 до n који су дељиви са 4.
4. Шта представља угнежђени циклус?
5. Објасни зашто се већи програми све тежи за одржавање од мањих.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објасниш шта представља модуларно програмирање;
- примењујеш принцип "завади па владај";
- анализираш сложене проблеме и да их делиш на мање задатке;
- организујеш програмска решења у логичке целине;
- припремаш програме за коришћење функција.

Са повећањем сложености програма јавља се потреба за бољом организацијом кода и ефикаснијим приступом решавању проблема. Мали програми могу се релативно лако написати као једна целина, али код већих програмских решења такав приступ ствара бројне тешкоће. Ако се све инструкције сместе на једно место, програм постаје тежак за читање, разумевањебирање, тестирање и одржување.

Због тога се у савременом програмирању користе технике које омогућавају организовање сложених проблема на уређенији начин.

Један од најважнијих принципа који се користе при решавању сложених проблема јесте принцип „подели па владај". Уместо да се цео проблем посматра као један велики задатак, он се дели на више мањих задатака који се лакше могу анализирати и решити. Након што се сваки мањи задатак успешно реши, добијена решења комбинују се у једно целовито решење. Овакав приступ представља основу модуларног програмирања и користи се у готово свим савременим програмским језицима и софтверским системима.

У свакодневном животу многи проблеми природно се решавају поделом на више мањих активности. Када се организује школска екскурзија, сви задаци се не решавају истовремено. Најпре се планира путања, затим се организује превоз, резервише смештај и на крају припремају сви потребни документи. Свака од ових активности представља посебан задатак који се може независно анализира и реализује. Иста идеја се примењује и у програмирању. Уместо да покуша да одједном реши цео проблем, програмер га дели на више логичких целина. На тај начин решење постаје разумљивије и лакше за развој. Поред тога, грешке се лакше откривају јер се сваки део програма може анализирати одвојено.

Овакав начин рада представља суштину модуларног програмирања. Модули најчешће извршавају тачно одређене задатке и међусобно сарађују како би се добио коначни резултат. Иако се у овој лекцији функције још неће користити, започеће се са развијањем начина размишљања који ће касније омогућити лакше разумевање функција и њихове примене у програмима.

Пример 1: Анализа проблема

Задатак: Израчунати успех ученика/це на основу пет оцена.
Проблем може да се подели на следеће мање задатке:
Унос оцена.
Израчунавање збира оцена.
Израчунавање просека.
Одређивање успеха.
Приказивање резултата.
На овај начин, уместо да се анализира цео проблем одједном, он се дели на више једноставних корака.

Након што се проблем подели на мање задатке, следећи корак је њихова организација. Веома је важно да сваки задатак има јасно одређену улогу. На пример, део задужен за унос података не треба истовремено да обавља сложене прорачуне. Такође, део који израчунава резултате не треба да буде задужен за приказивање података на екрану. Оваква подела омогућава да програм буде организованији и лакши за разумевање. У великим програмским системима један програм може да садржи десетине или стотине логичких целина које међусобно комуницирају. Ако организација није добра, код постаје хаотичан и тешко се може надограђивати. Због тога већ у раним фазама учења програмирања треба развијати навику поделе проблема и организовања решења у јасне логичке целине. Овакав начин размишљања касније знатно олакшава коришћење функција, библиотека и других програмских техника.

Пример 2: Псеудокод према принципу "завади па владај"

Задатак: Да се израчуна коначна цена производа са могућим попустом.
Проблем може да се подели на следеће мање задатке:
Уношење цене производа.
Уношење количине.
Пресметување на вкупната сума.
Провера да ли следује попуст.
Обрачун коначне цене.
Приказивање резултата.

Псеудокод:
Почетак
Унеси cena
Унеси kolicina
Израчунај suma = cena * kolicina
Ако suma > 5000
Израчунај popust = suma * 0.10
Израчунај konacnaCena = suma - popust
Иначе
konacnaCena = suma
Крај ако
Прикажи konecna
Cena Крај

У овом примеру може се уочити да је проблем најпре подељен на неколико мањих задатака. Сваки задатак има јасно одређену улогу у решењу. Захваљујући таквој организацији, алгоритам постаје лакши за разумевање и припремљен је за даљу реализацију у програмском језику.

Управо овакав начин размишљања представља суштину принципа „подели па владај" и основу модуларног програмирања.

Пример 3: Програм организован у логичке целине

У следећем примеру биће приказано како се један програм може организовати у више логичких целина иако се функције још не користе. Програм израчунава коначну цену производа са могућим попустом.

main.cpp

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

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

```
Unesi cenu proizvoda: 20
Unesi količinu: 30
Konačna cena: 600
----------
```

У програму се јасно могу уочити три логичке целине. Најпре се уносе подаци, затим се обрађују израчунавањем износа и попуста, а на крају се приказује резултат. Иако је програм написан у оквиру функције main(), логичке целине већ се лако могу препознати. У наредним лекцијама ове целине почеће да се издвајају у посебне функције. На тај начин програми ће постати уређенији, лакши за разумевање и једноставнији за одржавање. Ово је први практични корак ка правом модуларном програмирању.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља модуларно програмирање?
2. Шта представља принцип "завади па владај"?
3. Зашто се сложени проблеми деле на мање задатке?
4. Како модуларни приступ помаже при откривању грешака?
5. Зашто је важно да програм буде организован у логичке целине?

ЗАДАЦИ

1. Анализирај програм за израчунавање плате и подели га на логичке целине.
2. Анализирај програм за израчунавање потрошње горива и одреди које задатке треба извршити по редоследу.
3. Напиши псеудокод за програм који ће израчунавати просек од четири теста.
4. За програм за евиденцију библиотеке наведи најмање четири логичке целине.
5. Објасни како би принцип „завади па владај" помогао при изради школског информационог система.

ЗАПАМТИ ШТА СИ НАУЧИО

- Модуларно програмирање представља организацију програма у мање логичке целине.
- Принцип "завади па владај" омогућава да се сложени проблеми решавају преко подела на мање задатке.
- Добра организација програма омогућава лакше разумевање и одржавање кода.
- Логичке целине представљају основу за коришћење функција.
- Модуларно размишљање представља важну вештину за сваког програмера.
-

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У великим софтверским компанијама ретко који програмер самостално ради на целокупном програму. Уместо тога, програм се дели на велики број модула, а сваки тим развија одређени део система. На пример, код једне интернет продавнице један тим може да ради на регистрацији корисника, други на плаћању, трећи на обради наруџбина, а четврти на статистичким извештајима.

Иако сваки тим ради на различитом модулу, сви модули на крају функционишу као јединствен систем. Управо зато модуларно програмирање представља једну од најважнијих идеја у савременом развоју софтвера и основу за стварање великих и сложених програмских решења.

2.8

ПАРАМЕТРИ И ВРАЋАЊЕ ВРЕДНОСТИ

ВЕЋ ЗНАШ ДА...

- објасниш шта представља модуларно програмирање;
- примењујеш принцип "завади па владај";
- анализираш програмски проблем и да делиш га на мање задатке;
- организујеш програм у логичке целине;
- користиш услове и петље у C++.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта представља модуларно програмирање?
2. Шта представља принцип „завади па владај"?
3. Зашто се велики проблеми деле на мање задатке?
4. Наведи три логичке целине које могу да постоје у једном програму.
5. Објасни како добра организација програма помаже при откривању грешака.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објасниш шта представља функција;
- дефинишеш једноставне функције у C++;
- позиваш функције из главног програма;
- разликујеш дефиницију и позив на функцију;
- организујеш програме са више функција.

Како програми постају већи, јавља се потреба за бољом организацијом кода. У претходној лекцији објашњено је да се сложени проблеми могу поделити на више мањих задатака. Такав начин размишљања представља основу модуларног програмирања. Ипак, да би се ова подела практично реализовала у програмском језику, потребан је механизам који омогућава да се сваки задатак смести у посебну програмску целину.

У програмском језику C++ ту улогу имају функције. Функција је део програма (потпрограм) и представља именовани блок инструкција који извршава одређени задатак. Уместо да се исти кôд више пута пише у програму, он се може сместити у функцију, а затим позвати када је потребно. Захваљујући томе, програми постају уређенији, краћи и лакши за одржавање. Због тога функције представљају један од најважнијих алата у савременом програмирању.

У сваком C++ програму постоји најмање једна функција – main(). Она представља почетну тачку извршавања програма. Поред функције main(), програмер може да креира и сопствене функције за извршавање различитих задатака. Такав приступ омогућава организовање програма у више делова, уместо смештања целокупног кода у једну функцију. Када свака функција има јасно одређен задатак, програм постаје много лакши за читање и разумевање. Поред тога, ако се појави грешка, пажња се може усмерити на конкретну функцију уместо на цео програм. Због тога коришћење функција представља важан корак у развоју добрих програмерских навика.

При дефинисању функције, испред њеног имена наводи се тип функције. Тип одређује да ли ће функција након извршавања вратити резултат и ког ће типа тај резултат бити.

У овој лекцији се користи тип:

void

Кључна реч void означава да функција не враћа никакву вредност. Такве функције најчешће се користе за извршавање одређеног задатка, као што су приказивање поруке на екрану, исписивање наслова или извршавање одређених инструкција, без враћања резултата програму.

Општи облик овакве функције је:

```
void imeNaFunkcija()
{
  instrukcije
}
```

У програмирању се користе и функције које враћају вредност, на пример: int zbir()

```
{
  return 10;
}
```

У овом случају функција враћа цео број (int), док функција типа void не враћа ништа.

Пример 1: Едноставна функција

main.cpp

```cpp
#include <iostream>

using namespace std;

void pozdrav()
{
    cout << "Dobro došli u svet funkcija!" << endl;
}

int main()
{
    pozdrav();

    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Dobro došli u svet funkcija!

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

У примеру је дефинисана функција под називом pozdrav(). Када се функција позове, на екрану ће бити исписана порука. Веома је важно уочити да дефинисање функције не подразумева и њено аутоматско извршавање. Да би се извршиле инструкције у функцији, програм мора да је позове. Позив се врши навођењем назива функције и заграда. Овај механизам омогућава да се иста функција користи више пута без поновног писања истог кода. У реалним програмима то значајно смањује дужину кода и побољшава његову читљивост. Програмери често креирају функције за задатке који се понављају. Тако се избегава непотребно дуплирање инструкција и побољшава организација програма.

Пример 2: Вишеструко позивање функције

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  void linija()
6  {
7      cout << "====================" << endl;
8  }
9
10 int main()
11 {
12     linija();
13     cout << "Prvi deo programa    " << endl;
14
15     linija();
16     cout << "Drugi deo programa    " << endl;
17
18     linija();
19
20     return 0;
21 }
22
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
====================
Prvi deo programa
====================
Drugi deo programa
====================
```

У претходном примеру иста функција позвана је више пута. Уместо да се линија знакова сваки пут поново исписује у програму, она је смештена у функцију. Захваљујући томе, кôд је краћи и уређенији. Ово је једна од најважнијих предности функција – могућност поновне употребе већ написаног кода. У великим програмима иста функција може бити позвана десетинама или стотинама пута. Ако је потребно променити начин рада, довољно је изменити кôд само у функцији, без претраживања целог програма. Због тога функције знатно олакшавају развој и одржавање софтвера.

Пример 3: Програма са више функција

main.cpp

```cpp
#include <iostream>

using namespace std;

void naslov()
{
    cout << "Izveštaj za učenika " << endl;
}

void kraj()
{
    cout << "Kraj izveštaja   " << endl;
}

int main()
{
    naslov();

    cout << "Učenik ima odličan uspeh ." << endl;

    kraj();

    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Izveštaj za učenika
Učenik ima odličan uspeh.
Kraj izveštaja
```

У овом примеру користе се две различите функције. Једна је задужена за приказивање наслова, а друга за приказивање завршне поруке. На тај начин задаци у програму подељени су на више логичких целина.

Иако је програм мали, лако се може уочити како функције доприносе бољој организацији кода. У наредним лекцијама функције ће добити додатне могућности коришћењем параметара и повратних вредности, чиме ће моћи да извршавају још сложеније задатке

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља функција?
2. Која је улога функције main()?
3. Шта представља позив функције?
4. Разликујеш дефиницију и позив на функцију;
5. Зашто функције побољшавају организацију програма?

ЗАДАЦИ

1. Напиши функцију која ће приказивати поруку "Добар дан!".
2. Напиши функцију која ће приказивати линију састављену од 30 знакова *.
3. Напиши програм који ће користити две функције: једна за приказивање наслова и друга за приказивање завршне поруке.
4. Напиши програм који ће три пута позвати исту функцију.
5. Објасни како функције помажу при развоју великих програма.

ЗАПАМТИ ШТА СИ НАУЧИО

- Функција представља именовани блок инструкција.
- Функције омогућавају поделу програма на мање целине.
- Дефиниција описује функцију, а позив је активира.
- Иста функција може бити позвана више пута.
- Функције доприносе бољој организацији и одржавању програма.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим програмским језицима постоје библиотеке које садрже хиљаде већ припремљених функција. Уместо да увек пишу кôд од почетка, програмери могу да користе ове функције за математичка израчунавања, рад са текстом, обраду датотека и многе друге задатке. Захваљујући оваквом приступу, развој софтвера постаје бржи и ефикаснији. Зато функције представљају једну од најважнијих идеја у програмирању и основу за развој сложених софтверских система.

2.9

ПАРАМЕТРИ И ВРАЋАЊЕ ВРЕДНОСТИ

ВЕЋ ЗНАШ ДА...

- објасниш шта представља функција;
- дефинишеш једноставне функције у C++;
- позиваш функције из главног програма;
- разликујеш дефиницију и позив на функцију;
- организујеш програме у више функција.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта представља функцију?
2. Која је улога функције main()?
3. Шта представља позив на функцију?
4. Напиши функцију која ће приказивати поруку „Добродошли!“.
5. Зашто функције доприносе бољој организацији програма?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

користиш параметре у функцијама;
разликујеш параметре и аргументе;
дефинишеш функције које примају податке;
користиш функције које враћају вредност;
примењујеш наредбу return.

Функције омогућавају да програми буду организовани у више логичких целина и доприносе бољој прегледности и одржавању кода. При њиховој употреби често се јавља потреба за обрадом различитих података при сваком позиву функције. У ту сврху се у програмском језику C++ користе параметри преко којих се функцији прослеђују вредности које треба обрадити. На овај начин омогућава се креирање флексибилнијих програмских решења и поновна употреба истог кода у различитим ситуацијама.

Поред примања података, веома често је потребно да функција врати резултат који ће моћи да се користи у другом делу програма. На пример, функција може да израчуна површину, квадрат броја или просек и да добијени резултат врати назад у програм. Захваљујући овој могућности, функције постају много моћније и флексибилније. У овој лекцији биће обрађена употреба параметара и започеће изучавање функција које враћају вредност. Када функција треба да обрађује различите податке, ти подаци се прослеђују преко параметара. Параметри представљају промењиве које се наводе у заградама функције. При сваком позиву функције, параметрима се додељују нове вредности. На тај начин иста функција може да се користи за обраду различитих података без поновног писања кода. Овакав приступ значајно повећава поновну употребљивост програма и омогућава креирање флексибилнијих решења. Уместо да се креира посебна функција за сваку вредност, може се креирати једна функција која ће радити са различитим подацима.

Због тога су параметри једна од најважнијих карактеристика функција. Они омогућавају комуникацију између главног програма и функције и представљају основу за развој сложенијих програмских решења.

ПАРАМЕТРИ И АРГУМЕНТИ

При раду са функцијама често се користе појмови параметар и аргумент.

- Параметар је променљива наведена у дефиницији функције.
- Аргумент је вредност која се прослеђује при позиву функције.

Пример 1: Функција са једним параметром

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  void prikaziBroj(int broj)
6  {
7      cout << "Uneti broj je  : " << broj << endl;
8  }
9
10 int main()
11 {
12     prikaziBroj(15);
13
14     return 0;
15 }
16
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Uneti broj je  : 15

Process returned 0 (0x0)   execution time : 0.036 s
Press any key to continue.
```

У примеру функција прима једну вредност преко параметра broj. При позиву функције, аргумент 15 се прослеђује параметру и затим се приказује на екрану. Захваљујући параметрима, иста функција може да се користи са различитим вредностима без икаквих измена у њеном коду. Овакав приступ знчајно по-

већава флексибилност програма и омогућава креирање краћег и организованијег кода. Функције могу истовремено да примају више параметара. У таквим случајевима сваки параметар добија своју вредност при позиву функције. Ово омогућава функцији да ради са више података и извршава сложеније задатке. У пракси се често користе функције које примају две, три или више вредности. При томе је редослед аргумената веома важан, јер се свака вредност додељује одговарајућем параметру према њеној позицији.

Овакав механизам представља основу за комуникацију различитих делова програма. Захваљујући њему, функције могу да се користе у великом броју различитих ситуација без потребе за креирањем нових функција за сваки појединачни случај.

Пример 2: Функција са два параметра

main.cpp

```
#include <iostream>

using namespace std;

void pecatiPravoagolnik(int redovi, int koloni)
{
    for (int i = 1; i <= redovi; i++)
    {
        for (int j = 1; j <= koloni; j++)
        {
            cout << "* ";
        }

        cout << endl;
    }
}

int main()
{
    pecatiPravoagolnik(3, 5);

    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Deb
* * * * *
* * * * *
* * * * *

Process returned 0 (0x0
Press any key to contir
```

У примеру функција прима два параметра. Први параметар одређује број редова, а други број колона. Захваљујући томе, иста функција може да штампа правоугаонике различитих димензија, без потребе за изменом кода у функцији. Ово су функције које примају податке преко параметара. Ипак, у многим ситуацијама није довољно да функција само изврши одређени задатак. Често постоји потреба да се резултат обраде врати назад у програм како би могао да се користи у наставку. У ту сврху користе се функције које враћају вредност. За разлику од void функција, које не враћају резултат, ове функције имају одређени тип података који ће бити враћен након њиховог извршавања. Враћање вредности реализује се помоћу наредбе return.

Када се изврши return, рад функције се завршава и добијена вредност се враћа на место са којег је функција позвана. Захваљујући овој могућности, функције могу да се користе као део израза, израчунавања и других програмских конструкција. Овакав начин рада значајно проширује њихову примену и омогућава изградњу сложенијих програмских решења. Због тога су функције које враћају вредност један од најважнијих алата у програмирању.

Функције које враћају вредност

Општи облик функције која враћа вредност је:

```
tip imeNaFunkcija(parametri)
{
   instrukcije

   return vrednost;
}
```

Уместо void, пред именом функције се наводи тип података који ће бити враћен. На пример, ако функција враћа цео број, користи се тип int.

Пример 3: Функција која враћа вредност

main.cpp

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

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
KvKvadrat je: 49

Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

У примеру функција kvadrat() прима један параметар и враћа резултат помоћу наредбе return. Враћена вредност затим се чува у променљивој rezultat. Захваљујући овој могућности, функције могу да извршавају израчунавања и добијене резултате прослеђују другим деловима програма. То представља једну од најчешћих примена функција у савременом програмирању.

Функције које враћају вредност често се користе за математичка израчунавања, анализу података и решавање различитих програмских проблема. Уместо да се исто израчунавање више пута понавља у програму, оно се може сместити у функцију и затим користити кад год је потребно. На тај начин кôд постаје краћи и разумљивији.

Поред тога, ако је потребно променити начин израчунавања, довољно је изменити само функцију. Остатак програма наставља да ради без измена. Због тога функције доприносе бољој организацији, већој могућности поновне употребе кода и лакшем одржавању програма.

У наредним лекцијама ови концепти биће додатно проширени изучавањем различитих начина прослеђивања података функцијама и рада са променљивама у оквиру њиховог опсега.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља параметар?
2. Шта представља аргумент?
3. У чему је разлика између функције типа void и функције која враћа вредност?
4. У чему је улога наредбе return?
5. Зашто су функције са параметрима флексибилније од функција без параметара?

ЗАДАЦИ

1. Напиши функцију stampajIme() која ће примати име као параметар и приказиваће га на екрану.
2. Напиши функцију veciBroj() која ће примати два цела броја и враћати већи број.
3. Напиши функцију povrsinaPravougaonika() која ће примати дужину и ширину и враћати површину.
4. Напиши функцију која ће враћати kub() куб унетог броја.
5. Напиши програм који ће користити функцију за израчунавање обима квадрата и приказиваће резултат.

ЗАПАМТИ ШТА СИ НАУЧИО

- Параметри представљају променљиве наведене у дефиницији функције.
- Аргументи су вредности које шаљу при позивању на функцију.
- Функције могу да приме један или више параметара.
- Наредба return се користи за враћање вредности функције.
- Функције које враћају вредност може да се користе у израчунавању и изразима.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У великим софтверским системима постоје функције које примају десетине параметара и враћају сложене резултате. Ипак, у професионалном програмирању препоручује се да функције имају јасну намену и извршавају један конкретан задатак. Такав приступ чини кôд лакшим за читање, тестирање и одржавање.

Многи савремени програмски језици садрже хиљаде готових функција које програмери свакодневно користе за рад са текстом, датотекама, мрежама и графиком, као и за математичка израчунавања. Функције представљају једну од најважнијих основа савременог програмирања, а њихово добро познавање знатно олакшава развој сложених програмских решења.

2.10

ЛОКАЛНЕ И ГЛОБАЛНЕ ПРОМЕНЉИВЕ КОД ФУНКЦИЈА

ВЕЋ ЗНАШ ДА...

- дефинишеш и позиваш функције;
- користиш параметре у функцијама;
- разликујеш параметре и аргументе;
- користиш функције које враћају вредност;
- примењујеш наредбу return.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта представља параметар?
2. Шта представља аргумент?
3. У чему је улога наредбе return?
4. Напиши функцију која ће примати број и приказиваће га на екрану.
5. Зашто су функције са параметрима флексибилније од функција без параметра?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објасниш шта представља опсег променљиве;
- разликујеш локалне и глобалне променљиве;
- анализираш где може да се користи одређена променљива;
- користиш глобалне променљиве у више функција;
- препознајеш предности и недостатке различитих врста променљивих.

При развоју програмских решења непрестано се користе променљиве за чување података. Међутим, не могу се све променљиве користити у сваком делу програма. Могућност приступа одређеној променљивој зависи од места на којем је она декларисана. Због тога се у програмском језику C++ користи појам област важења променљиве.

Опсег одређује у којем делу програма се одређена променљива може користити. У зависности од места декларације, променљиве могу бити локалне или глобалне.

Познавање ових појмова је много важно, јер правилан избор врсте променљиве доприноси бољој организацији кода и лакшем откривању грешака у програмима.

Свака променљива декларисана у програму има одређену област важења. То значи да постоје правила која одређују где се она може користити, а где јој се не може приступити. Када се променљива декларише унутар функције, њена употреба ограничена је само на ту функцију. Такве променљиве називају се локалне променљиве. Оне представљају најчешће коришћену врсту променљивих у програмима јер омогућавају бољу контролу података и смањују могућност појаве нежељених грешака. Свака функција може имати сопствене локалне променљиве које нису видљиве другим функцијама. Захваљујући томе, различите функције могу независно да раде са сопственим подацима, а да не утичу једна на другу. Овај принцип доприноси већој поузданости и бољој организацији програмског кода.

Пример 1: Локална променљива

main.cpp

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

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
10

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

У примеру је променљива broj декларисана у функцији poruka(). Због тога се може користити само унутар те функције. Ако се покуша њено коришћење у функцији main() или некој другој функцији, програм ће пријавити грешку. Овакво понашање резултат је ограничене области важења локалних променљивих.

Поред локалних променљивих, у програмима се могу користити и глобалне променљиве. За разлику од локалних, глобалне променљиве декларишу се изван свих функција. Због тога им се може приступити из више функција у истом програму. То омогућава дељење одређених података између различитих делова програма.

Ипак, овакав приступ треба користити пажљиво. Када велики број функција користи и мења исту глобалну променљиву, постаје теже пратити ток података и открити могуће грешке. Због тога се у савременом програмирању најчешће даје предност локалним променљивама, док се глобалне користе само када заиста постоји потреба за дељењем података.

Пример 2: Глобална променљива

main.cpp

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

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
25
25

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

У овом примеру променљива broj декларисана је изван свих функција. Због тога се може користити и у функцији prikaziBroj() и у функцији main().

При раду са више функција може се појавити ситуација у којој локална и глобална променљива имају исто име. У том случају предност има локална променљива унутар функције у којој је декларисана. То значи да се у тој функцији неће користити глобална променљива са истим именом.

Ова појава може изазвати забуну код програмера, нарочито у већим програмима са великим бројем променљивих. Због тога при именовању променљивих треба бирати јасна и недвосмислена имена. Добра организација променљивих доприноси бољој читљивости кода и лакшем откривању логичких грешака.

Поред тога, коришћење локалних променљивих омогућава да свака функција буде независна од осталих делова програма. На тај начин смањује се могућност ненамерне промене вредности коју користи друга функција. Управо зато локалне променљиве најчешће представљају препоручени начин рада са подацима у програмским решењима.

Пример 3: Локална и глобална променљива са истим именом

main.cpp

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

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

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

У примеру постоје две променљиве са истим именом. У функцији prikazi() користи се локална променљива са вредношћу 20, док се у функцији main() користи глобална променљива са вредношћу 100. Овај пример показује да место декларације значајно утиче на то која ће променљива бити коришћена у програму.

Избор између локалних и глобалних променљивих зависи од конкретног проблема који се решава. Локалне променљиве омогућавају бољу контролу података и смањују могућност појаве нежељених промена у програму. Са друге стране, глобалне променљиве омогућавају коришћење истих података у више функција без потребе за непрестаним прослеђивањем тих вредности као параметара.

Ипак, прекомерна употреба глобалних променљивих може учинити програм тежим за разумевање и одржавање. Због тога се у пракси најчешће препоручује да се прво користе локалне променљиве, а да се глобалне уводе само када постоји јасна потреба за њиховим коришћењем у више делова програма. Такав приступ доприноси стварању уређенијих, поузданијих и лакше одрживих програмских решења.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља опсег променљиве?
2. Када се једна променљива сматра локалном?
3. Када се једна променљива сматра глобалном?
4. Која је основна разлика између локалних и глобалних променљивих?
5. Зашто прекомерна употреба глобалних променљивих може да створи проблеме?

ЗАДАЦИ

1. Напиши програм који ће садржати локалну променљиву у функцији и приказивати њену вредност!
2. Напиши програм који ће користити једну глобалну променљиву у две различите функције!
3. Напиши програм који ће садржати локалну и глобалну променљиву са различитим именима и приказиваће њихове вредности!
4. Анализирај у којим функцијама може да се користи глобална променљива декларисана ван свих функција!
5. Објасни у којим ситуацијама је боље да се користи локална променљива уместо глобалне!

ЗАПАМТИ ШТА СИ НАУЧИО

- Опсег променљиве одређује где може бити употребљена.
- Локалне променљиве се декларишу у оквиру функције и могу да се користе само у њој.
- Глобалне променљиве се декларишу ван свих функција.
- Глобалне променљиве могу да буду коришћене у више функција.
- Локалне променљиве најчешће омогућавају бољу организацију и већу сигурност програма.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У великим програмским системима могу постојати хиљаде променљивих. Када би све оне биле глобалне, праћење њихових вредности било би изузетно тешко. Због тога савремена програмерска пракса препоручује ограничавање употребе глобалних променљивих и коришћење локалних променљивих кад год је то могуће.

Овакав приступ омогућава бољу организацију програмског кода, лакше тестирање функција и смањење броја логичких грешака током развоја софтверских решења.

2.11

ОСНОВНЕ МАТЕМАТИЧКЕ ФУНКЦИЈЕ ИЗ БИБЛИОТЕКЕ CMATH (ДЕО 1)

ВЕЋ ЗНАШ ДА...

- дефинишеш и позиваш функције;
- користиш параметре у функцијама;
- користиш функције које враћају вредност;
- разликујеш локалне и глобалне променљиве;
- организујеш програм у више логичких целина.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта представља локалну променљиву?
2. Шта представља глобалну променљиву?
3. У ком делу програма може да буде коришћена локална променљива?
4. У чему је основна предност локалних променљивих?
5. Напиши функцију која прима број и враћа његов квадрат.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- користиш библиотеку cmath;
- примењујеш готове математичке функције;
- израчунаваш квадратни корен и степен броја;
- користиш функције за заокруживање;
- примењујеш тригонометријске функције;
- користиш математичке функције у програмским решењима.

При изради програма често се јавља потреба за извршавањем математичких прорачуна. Део тих прорачуна може лако да буде реализован аритметичким операторима, али за одређене опера- ције су потребни посложени алгоритми. На пример, израчунавање квадратног корена, степеновање или тригонометријске функције би тражило значајно већи број инструкција ако се реализују ручно.

Ради олакшавања рада програмера, програмски језик C++ садржи велики број готових функција смештених у различитим библиотекама. Једна од најчешће коришћених библиотека јесте <cmath>, у којој се налазе функције за математичка израчунавања. Њиховом применом сложене операције могу се извршавати брзо и једноставно, без потребе за писањем додатног кода. Захваљујући томе, програми постају краћи, разумљивији и лакши за одржавање. Библиотека <cmath> представља скуп готових математичких функција које се могу користити у програмима. За њихову употребу потребно је на почетку програма укључити библиотеку помоћу директиве #include <cmath>. Након укључивања библиотеке, програм добија приступ великом броју функција које извршавају различите математичке операције. Уместо да програмер самостално развија алгоритам за израчунавање квадратног корена или степена броја, може непосредно да употреби функције које се већ налазе у библиотеци. Овакав приступ скраћује време потребно за развој програма и доприноси већој поузданости резултата.

Функције из библиотеке <cmath> темељно су тестиране и користе се у великом броју професионалних софтверских решења. Због тога њихово познавање представља важан део програмерских вештина.

Пример 1: Функција sqrt()

Функција sqrt() се користи за израчунавање квадратног корена броја.

main.cpp

```cpp
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    double broj = 81;

    double koren = sqrt(broj);

    cout << "Kvadratni koren je   : " << koren << endl;

    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
KvKvadratni koren je :: 9

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

У примеру се израчунава квадратни корен броја 81. Добијени резултат је 9. Функција sqrt() прима број као аргумент и враћа вредност његовог квадратног корена.

Поред квадратног корена, често се користи и степеновање бројева. У математици степеновање представља операцију у којој се један број одређени број пута множи самим собом. У библиотеци <cmath> ова операција реализована је помоћу функције pow().

Она прима два аргумента. Први аргумент представља основу, а други степен на који број треба подићи. Захваљујући овој функцији, лако се могу реализовати бројне математичке формуле и алгоритми.

Употреба готових функција доприноси томе да кôд буде краћи и разумљивији у поређењу са ручним извођењем истих операција. Због тога је функција pow() једна од најчешће коришћених функција из библиотеке <cmath>.

Пример 2: Функција pow()

main.cpp

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

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Rezultat je: 32

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

У примеру се израчунава вредност израза 2^5. Добијени резултат је 32. Први аргумент представља основу, а други аргумент – степен. Поред функција за израчунавање квадратног корена и степеновање, библиотека cmath садржи и функције за рад са апсолутним вредностима и заокруживање бројева.

Апсолутна вредност једног броја представља његову удаљеност од нуле на бројевној правој и увек има ненегативну вредност. За њено израчунавање користи се функција fabs(). При раду са реалним бројевима често се јавља потреба да вредности буду заокружене. У ту сврху користе се функције ceil() и floor(). Функција ceil() заокружује број на први већи цео број, док функција floor() заокружује број на први мањи цео број.

Ове функције имају широку примену у различитим програмским решењима, посебно када је потребно да добијени резултати буду представљени у облику целих бројева. Захваљујући њима, програми могу једноставно да обрађују реалне вредности и прилагођавају их конкретним потребама проблема који се решава.

Неке често коришћене функције из библиотеке cmath

Функција	Намена
sqrt(x)	Квадратни корен
pow(a,b)	Степеновање
fabs(x)	Апсолутна вредност
ceil(x)	Заокруживање навише
floor(x)	Заокруживање наниже
sin(x)	Синус
cos(x)	Косинус

Пример 3: Коришћење више математичких функција

main.cpp

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

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Apsolutna vrednost: 15.7
Zaokruživanje naniže : -15
Zaokruživanje naniže :u: -16

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

У примеру се користе три различите функције из библиотеке <cmath>. Функцијом fabs() добија се апсолутна вредност броја, док функције ceil() и floor() обављају различите врсте заокруживања. Овакве операције често се користе при обради података и математичким израчунавањима у програмима.

У библиотеци <cmath> налазе се и тригонометријске функције које имају значајну примену у математици, физици, инжењерству и рачунарској графици. Најчешће се користе функције sin() и cos(), које израчунавају синус и косинус задатог угла. При њиховој употреби треба знати да се углови задају у радијанима. Иако се у школској математици најчешће ради са степенима, рачунарски програми обично користе радијане као мерну јединицу за углове. Тригонометријске функције примењују се при изради симулација, игара, анимација, графичких апликација и великог броја научних програма. Захваљујући библиотеци <cmath>, ова израчунавања могу се лако обавити без потребе за развијањем сопствених математичких алгоритама.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. За шта се користи библиотека cmath?
2. Која је намена функције sqrt()?
3. Која је намена функције pow()?
4. У чему је разлика између ceil() и floor()?
5. За шта се користе функције sin() и cos()?

ЗАДАЦИ

1. Напиши програм који ће израчунавати квадратни корен унетог броја!
2. Напиши програм који ће израчунавати вредност израза 3^4 коришћењем функције pow()!
3. Напиши програм који ће приказивати апсолутну вредност унетог реалног броја!
4. Напиши програм који ће приказивати резултате од ceil() и floor() за унети број!
5. Напиши програм који ће израчунавати синус и косинус угла задат у радијанима!

ЗАПАМТИ ШТА СИ НАУЧИО

- Библиотека cmath садржи готове математичке функције.
- Функција sqrt() се користи за израчунавање квадратног корена.
- Функција pow() се користи за степеновање.
- Функција fabs() израчунава апсолутна вредност броја.
- Функције ceil() и floor() се користе за заокруживање.
- Функције sin() и cos() се користе за тригонометријске прорачуне.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Библиотека <cmath> садржи много више функција од оних које су обрађене у овој лекцији. У њој се могу пронаћи функције за израчунавање логаритама, експоненцијалне, тригонометријске и инверзне тригонометријске функције, као и многе друге математичке операције.

У професионалним програмским решењима ове функције користе се при развоју научних симулација, инжењерских апликација, финансијских анализа, рачунарских игара и система за обраду података. Познавање библиотеке <cmath> омогућава програмеру да знатно брже решава математичке проблеме и ствара ефикаснија програмска решења.

2.12

РЕШАВАЊЕ ТЕКСТУАЛНИХ ЗАДАТАКА ПРИМЕНОМ ФУНКЦИЈА

ВЕЋ ЗНАШ ДА...

- дефинишеш и позиваш функције;
- користиш параметре и аргументе;
- користиш функције које враћају вредност;
- разликујеш локалне и глобалне променљиве;
- примењујеш функције из библиотеке cmath.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

- дефинишеш и позиваш функције;
- користиш параметре и аргументе;
- користиш функције које враћају вредност;
- разликујеш локалне и глобалне променљиве;
- примењујеш функције из библиотеке cmath.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- дефинишеш и позиваш функције;
- користиш параметре и аргументе;
- користиш функције које враћају вредност;
- разликујеш локалне и глобалне променљиве;
- примењујеш функције из библиотеке cmath.

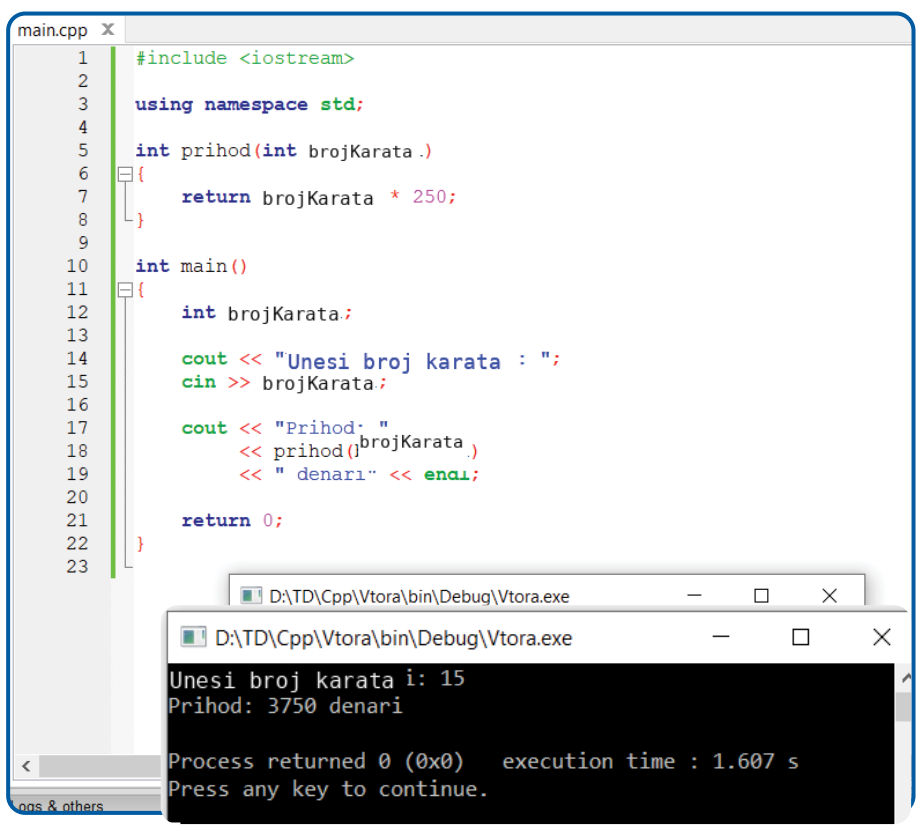
Функције представљају једну од најкориснијих алатки за организацију програмског кода. Њихова права вредност најбоље се уочава при решавању реалних проблема, када један програм треба да извршава више различитих задатака. Уместо да све инструкције буду смештене у функцију main(), програм се може поделити на више мањих целина, при чему свака функција добија јасно одређен задатак.

При решавању текстуалних задатака најпре треба анализирати проблем. Затим треба утврдити које се операције понављају и који делови решења могу бити издвојени у посебне функције. Овакав приступ омогућава да програми буду уреднији, лакши за разумевање и једноставнији за одржавање.

При решавању програмског проблема најпре треба утврдити који се подаци уносе, које прорачуне треба извршити и какав резултат треба приказати. Након ове анализе, решење се може поделити на више логичких целина. Свака целина може бити реализована као посебна функција. На тај начин добија се програм састављен од више мањих делова, уместо једне велике и сложене целине. Овакав приступ следи принцип „подели па владај" и представља основу модуларног програмирања.

Пример 1: Израчунавање цене карата

У биоскопу се продају улазнице по цени од 250 денара. Израчунати укупан приход за унети број продатих улазница.



У биоскопу се продају улазнице по цени од 250 денара. Израчунати укупан приход за унети број продатих улазница.

У многим ситуацијама потребно је да један програм користи више функција. Свака функција може да извршава различит задатак, а затим се добијени резултати могу комбиновати у главном програму. Овакав приступ је нарочито користан када проблем садржи више израчунавања која се могу независно реализовати.

Пример 2: Пресметување плоштина и периметар

Унети дужину и ширину правоугаоника. Израчунати његову површину и обим.

main.cpp

```
#include <iostream>

using namespace std;

double Površina (double a,double b)
{
    return a*b;
}

double perimetar(double a,double b)
{
    return 2*(a+b);
}

int main()
{
    double a,b;
    cin >>a>>b;
    cout <<"Površina : "<< Površina (a,b)<< endl;
    cout <<"Perimetar: "<< perimetar(a,b)<< endl;

    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
4
5
Površina a: 20
Perimetar: 18

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

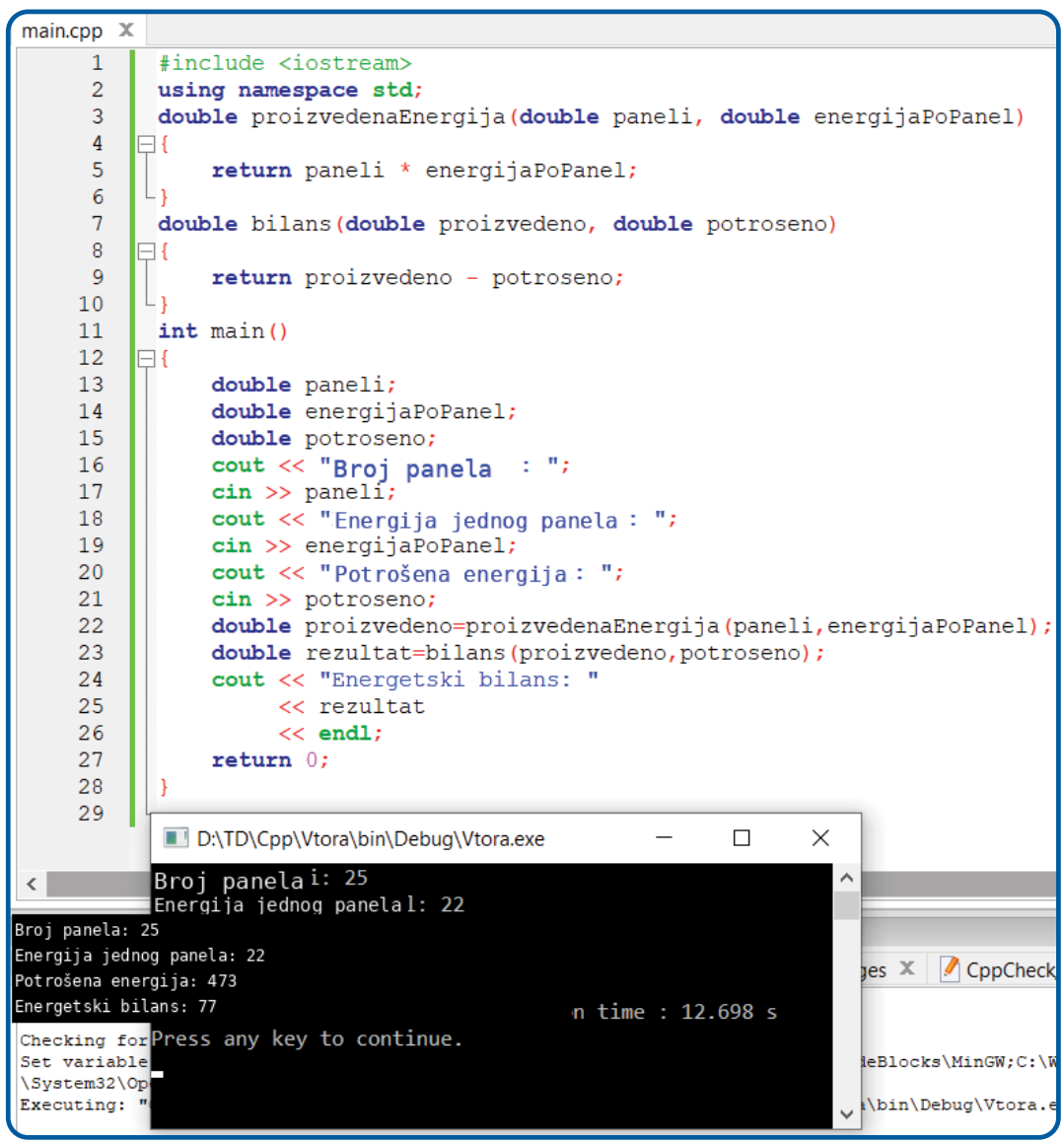
У примеру свака функција има јасно одређен задатак. Једна израчунава површину, а друга обим.

У практичним програмским решењима често није довољно користити само једну функцију. Већи проблеми најчешће су састављени од више мањих задатака које треба решавати независно један од другог. Због тога при анализи проблема најпре треба утврдити које ће се активности извршавати, а затим сваку од њих реализовати у посебној функцији. На тај начин програм добија јасну структуру, а свака функција конкретну одговорност. Када је програмски код организован на овакав начин, много се лакше откривају грешке и додају нове могућности. Поред тога, функције се могу поново користити и у другим програмима.

Управо зато анализа проблема и правилна подела задатка представља важан корак у развоју сваког програмског решења.

Пример 3: Истраживачка станица на Марсу

У истраживачкој станици на Марсу сваког дана се евидентирају количина електричне енергије произведене помоћу соларних панела и количина енергије коју потроше системи за одржавање живота. Израчунати да ли је остварен енергетски вишак или дефицит.



У истраживачкој станици на Марсу сваког дана се евидентирају количина електричне енергије произведене помоћу соларних панела и количина енергије коју потроше системи за одржавање живота. Израчунати да ли је остварен енергетски вишак или дефицит.

Функције представљају важан алат за развој квалитетних програмских решења. Њиховом применом сложени проблеми могу се поделити на више мањих целина којима се лакше управља. На тај начин омогућавају

се боља организација програмског кода, лакше тестирање појединачних делова решења и ефикасније откривање грешака. При анализи текстуалног задатка увек треба утврдити које активности могу бити издвојене у функције и који подаци треба да се прослеђују преко параметара. Овакав приступ представља основу за развој већих програмских пројеката и савремених софтверских система. Због тога се функције убрајају међу најважније концепте у програмском језику C++ и представљају основу за даље изучавање структура за складиштење и обраду података.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Зашто је корисно да се сложени проблеми поделе на више функција?
2. Када је оправдано да се креира нова функција?
3. У чему је улога параметара при решавању текстуалних задатака?
4. У чему је предност функција које враћају вредност?
5. Како функције доприносе бољој организацији програмског кода?

ЗАДАЦИ

1. У зоолошком врту се води евиденција о дневној потрошњи хране. Напиши програм у коме ће бити употребљене две функције: једна израчунава храну потребну за лавове, а друга за храну потребну за тигрове!
2. На аеродрому се следи количина горива у авионима. Напиши програм у коме једна функција израчунава укупну количину наточеног горива, а друга ће израчунавати преостало гориво по лету!
3. У ботаничкој башти засађују се кружне цветне површине. Напиши програм у коме ће једна функција израчунавати површин, а друга – цену уређења ако се цена по квадратном метру уноси са тастатуре!
4. У подводној истраживачкој бази прати се потрошња кисеоника. Напиши програм у коме ће бити употребљена функција која израчунава количину преосталог кисеоника према броју радних часова!
5. У астрономској опсерваторији израчунава се време потребно да би сигнал стигао од сателита до Земље. Напиши програм у коме ће бити употребљена функција која израчунава ову вредност на основу унетог растојања!

ЗАПАМТИ ШТА СИ НАУЧИО

- Текстуални задаци најпре треба да буду анализирани.
- Сложени проблеми могу да буду подељени на више мањих задатака.
- Сваки задатак може да буде реализован у посебној функцији.
- Функције омогућавају бољу организацију и поновну употребу кода.
- Параметри и повратне вредности омогућавају размену података између функција и програма.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Савремене рачунарске игре, мобилне апликације и оперативни системи садрже милионе линија програмског кода. Такви системи не би могли да буду развијени када би целокупан код био смештен у једној јединој функцији. Због тога програмери организују програме у велики број функција и модула, при чему свака функција решава мали део укупног проблема. На тај начин омогућава се да више програмера истовремено ради на истом пројекту, а програми постају лакши за одржавање и надоградњу. Овим се завршава тема о функцијама и ствара основа за изучавање структура које омогућавају обраду већег броја података, као што су једнодимензионални низови.

2.13

ЗАДАЦИ ЗА ВЕЖБАЊЕ ИЗ ФУНКЦИЈА

ЗАДАЦИ ЗА ФУНКЦИЈЕ У C++

Лаки задаци

1. У метеоролошкој станици свакодневно се прати температура ваздуха. Напиши програм у којем ћеш користити функцију која прима температуру у степенима Целзијуса и израчунава одговарајућу температуру у степенима Фаренхајта.
2. У грађевинској компанији израђују се квадратне бетонске плоче. Напиши програм у којем ћеш користити функцију која прима дужину странице плоче и израчунава њен обим.
3. Туристичка агенција организује путовање које траје више дана. Напиши програм у којем ћеш користити функцију која прима број дана и израчунава колико целих недеља садржи тај период.
4. У продавници се организује сезонска распродаја. Напиши програм у којем ћеш користити функцију која прима цену производа и проценат попуста и израчунава нову цену.
5. У здравственој анкети уносе се године испитаника. Напиши програм у којем ћеш користити функцију која прима број година и израчунава колико још година недостаје до навршених 100 година.
6. У парку се планира изградња кружне фонтане. Напиши програм у којем ћеш користити функцију која прима полупречник фонтане и израчунава дужину њеног обода.
7. У спортској сали води се евиденција о трајању тренинга. Напиши програм у којем ћеш користити функцију која прима број минута и израчунава колико часова представља тај број минута.
8. У лабораторији се упоређују резултати два мерења. Напиши програм у којем ћеш користити функцију која прима две вредности и израчунава њихову апсолутну разлику. У функцији користити локалну променљиву под називом razlika.
9. На бензинској пумпи анализира се количина горива. Напиши програм у којем ћеш користити функцију која прима количину горива изражену у литрима и претвара је у милилитре.
10. У банци се анализира стање на рачуну. Напиши програм у којем ћеш користити функцију која прима број и утврђује да ли је његова вредност позитивна.

Средње тешки задаци

1. У средњовековном замку сваки становник плаћа месечни порез од 12 новчића. Напиши програм у којем ћеш користити функцију која прима број становника и израчунава укупан месечни порез.
2. У радионици се проверавају три дрвене летве за израду троугластог оквира. Напиши програм у којем ћеш користити функцију која прима три дужине и проверава да ли се од њих може саставити троугао.
3. На школском такмичењу додељује се посебна ознака бројевима који су дељиви и са 3 и са 5. Напиши програм у којем ћеш користити функцију која прима број и проверава да ли испуњава овај услов.
4. На мапи се положај објекта означава координатама x и y. Напиши програм у којем ћеш користити функцију која прима координате и израчунава растојање објекта од координатног почетка.
5. У домаћинству се обрачунава рачун за електричну енергију. Напиши програм који садржи глобалну променљиву cenaKwh = 7 и функцију која прима број потрошених киловат-сати и израчунава износ за плаћање.
6. У планинарском дому једно ноћење кошта 850 денара. Напиши програм који садржи глобалну променљиву cenaNocenja = 850 и функцију која прима број ноћења и израчунава укупан износ.
7. У информационом систему проверава се дужина унетог идентификационог броја. Напиши програм у којем ћеш користити функцију која прима природан број и израчунава колико цифара он садржи.
8. У спортској дисциплини упоређују се резултати тројице такмичара. Напиши програм у којем ћеш користити функцију која прима три резултата и враћа највећи. У функцији користити локалну променљиву najveci.
9. У школској лабораторији израчунава се синус угла задатог у степенима. Напиши програм у којем ћеш користити функцију која прима угао у степенима, претвара га у радијане и израчунава његов синус коришћењем библиотеке cmath.
10. У финансијској апликацији обрађују се реалне вредности које треба заокружити. Напиши програм у којем ћеш користити функцију која прима реалан број и приказује резултате функција ceil() и floor().

Тешки задаци

1. У музеју се продају улазнице за одрасле и децу. Напиши програм који садржи глобалне променљиве cenaOdraslog = 200 и cenaDete = 120, као и функцију која прима број одраслих и број деце и израчунава укупан приход.

2. У календарској апликацији треба проверити да ли је одређена година преступна. Напиши програм у којем ћеш користити функцију која прима годину и враћа податак о томе да ли је та година преступна.

3. У безбедносном систему број се анализира преко збира његових цифара. Напиши програм у којем ћеш користити функцију која прима природан број и израчунава збир његових цифара.

4. У математичкој апликацији израчунавају се производи узастопних бројева. Напиши програм у којем ћеш користити функцију која прима природан број и израчунава његов факторијел.

5. У видео-игри играч откључава ниво у зависности од броја прикупљених златника. Напиши програм у којем ћеш користити функцију која прима број златника и одређује који је ниво откључан: до 100 златника – ниво 1, од 101 до 300 – ниво 2, а више од 300 – ниво 3.

6. У фабрици се две серије производа пакују у једнаке групе без остатка. Напиши програм у којем ћеш користити функцију која прима два броја и израчунава њихов највећи заједнички делилац.

7. У систему за мерење времена уноси се трајање у секундама. Напиши програм у којем ћеш користити функцију која прима број секунди и приказује сате, минуте и секунде. У функцији користити локалне променљиве casovi, minuti и sekundi.

8. У метеоролошкој станици мере се температура и влажност ваздуха. Напиши програм у којем ћеш користити функцију која прима две вредности и враћа податак о томе да ли постоји могућност појаве магле ако је температура нижа од 5 °C, а влажност већа од 85%.

9. У навигационој апликацији израчунава се растојање између две тачке на мапи. Напиши програм у којем ћеш користити функцију која прима координате обе тачке и израчунава растојање између њих.

10. У астрономској опсерваторији прати се кретање комете. Напиши програм у којем ћеш користити функцију која прима пређено растојање и време и израчунава просечну брзину. У функцији користити локалну променљиву brzina.

2.14

ЈЕДНОДИМЕНЗИОНАЛНИ НИЗ КАО СТРУКТУРА ПОДАТАКА

ВЕЋ ЗНАШ ДА...

- користиш промењљиве различитог типа;
- уносиш и приказујеш податке;
- користиш условне конструкције;
- користиш циклусе;
- дефинишеш и позиваш функције.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта представља функција?
2. У чему је улога параметара?
3. За шта се користи наредба return?
4. Зашто је корисно да један програм буде подељен на више функција?
5. Наведи једну функцију из библиотеке cmath.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објасниш шта представља низ;
- препознаш ситуације у којима је потребан низ;
- разликујеш промењљиву од низа;
- објасниш како се чувају елементи низа у меморији;
- користиш једнодимензионалне низе у једноставним програмским решењима.

При развоју програмских решења често се јавља потреба за обрадом великог броја података исте врсте. На пример, може бити потребно сачувати оцене ученика у једном одељењу, дневне температуре током једног месеца или резултате такмичара на спортском турниру. Ако се за сваки податак користи посебна променљива, програм брзо постаје сложен и тежак за одржавање.

За ефикаснију организацију података у програмском језику C++ користе се низови. Низ омогућава чување више података истог типа под једним заједничким именом. Захваљујући томе, подаци се могу лакше уносити, обрађивати и анализирати. Због тога низови представљају једну од најважнијих структура за чување података у програмирању.

Променљива омогућава чување једне вредности у меморији рачунара. Када је потребно сачувати десет, двадесет или сто вредности исте врсте, коришћење великог броја појединачних променљивих није практично решење. У таквим ситуацијама користи се низ.

Низ представља скуп елемената истог типа података који су смештени један поред другог у меморији. Сви елементи повезани су једним заједничким именом, а сваки елемент има своју позицију. Захваљујући оваквој организацији, велики број података може се обрађивати као једна целина. Низови се користе у готово свим савременим програмским решењима јер омогућавају ефикасну обраду података и представљају основу за многе сложеније структуре у програмирању.

У програмском језику C++ низ се декларише навођењем типа податка, назива низа и броја елемената које ће садржати. Општи облик је:

```
tip imeNaNiza[velicina];
На пример: int ocene[5];
```

У овом примеру кључна реч int показује да ће се у низу чувати цели бројеви. Назив низа је ocenki, а број 5 означава да низ садржи пет елемената. Иако низ има пет елемената, њихове позиције не почињу од 1, већ од 0. Због тога се први елемент налази на позицији 0, други на позицији 1, трећи на позицији 2, четврти на позицији 3, а пети на позицији 4. Опште правило гласи: код низа величине n, први елемент налази се на позицији 0, а последњи на позицији n - 1.

За низ:

```
int ocenki[5];
позиције елемената су:
ocene[0] ocenki[1] ocene[2] ocene[3] ocene[4]
```

Због тога се приликом приступања елементима увек мора водити рачуна да се не користи позиција већа од последње дозвољене вредности, јер то може проузроковати неправилан рад програма.

Пример 1: Проблем без коришћења низа

Да се прикажу оцене пет ученика.

main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
    int ocena1 = 5;
    int ocena2 = 4;
    int ocena3 = 5;
    int ocena4 = 3;
    int ocena5 = 5;

    cout << ocena1 << endl;
    cout << ocena2 << endl;
    cout << ocena3 << endl;
    cout << ocena4 << endl;
    cout << ocena5 << endl;

    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
5
4
5
3
5

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

У примеру је за сваку оцену употребљена посебна променљива. Решење је прихватљиво када је број података мали, али постаје непрактично ако треба обрадити десетине или стотине вредности.

Када више података има исти тип и исту намену, они се могу организовати у низ. Уместо коришћења великог броја различитих имена променљивих, све вредности групишу се под једним заједничким именом. На тај начин програм постаје краћи и уреднији. Поред тога, подаци се могу лако обрађивати помоћу циклуса. Управо ова могућност представља једну од највећих предности низова и разлог њихове широке примене у програмирању.

Пример 2: Исти подаци организовани у низу

main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
    int ocene [5] = {5, 4, 5, 3, 5};

    cout << ocene [0] << endl;
    cout << ocene [1] << endl;
    cout << ocene [2] << endl;
    cout << ocene [3] << endl;
    cout << ocene [4] << endl;

    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
5
4
5
3
5

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

У примеру су све оцене сачуване у једном низу под називом ocenki. На тај начин више вредности организовано је под једним заједничким именом, што програм чини једноставнијим и лакшим за одржавање.

Елементи једног низа смештају се на узастопне меморијске локације. То значи да их рачунар чува један до другог у меморији. Захваљујући оваквој организацији, приступ елементима веома је брз и ефикасан. Иако су сви елементи повезани једним заједничким именом, сваки од њих представља посебну вредност која се може независно прочитати или изменити. Овакав начин организовања података омогућава да се велики број вредности обрађује као једна целина. Управо због тога низови имају значајну улогу у програмирању и представљају основу за велики број сложенијих структура података. При раду са низовима увек треба имати у виду да сви елементи морају бити истог типа. Није дозвољено да се у истом низу истовремено чувају цели бројеви, реални бројеви и текстуалне вредности.

Пример 3: Број посетиоца у музеју

Током пет дана евидентиран је број посетилаца у једном музеју. Податке треба сачувати у низу и приказати на екрану.

```
#include <iostream>

using namespace std;

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

   cout << "Dan 1: " << posetioci[0] << endl;
   cout << "Dan 2: " << posetioci[1] << endl;
   cout << "Dan 3: " << posetioci[2]<< endl;
   cout << "Dan 4: " << posetioci[3]<< endl;
   cout << "Dan 5: " << posetioci[4]<< endl

   return 0;
}
```

У примеру су подаци за свих пет дана смештени у један низ. Уместо коришћења пет различитих променљивих, све вредности организоване су под једним заједничким именом. На тај начин програм постаје уреднији и лакши за одржавање.

Низови се користе увек када треба обрадити већи број података истог типа. Могу се применити за чување оцена, температура, финансијских података, спортских резултата, мерења са сензора и многих других информација. Без њих би развој већих програмских решења био знатно тежи. Због тога изучавање низова представља важан корак у развоју програмерских вештина. Након савладавања основног концепта може се прећи на унос, обраду и анализу елемената низова. Управо ове активности представљају основу великог броја алгоритама и програмских решења која се користе у пракси.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља једнодимензионални низ?
2. Када је оправдано коришћење низа уместо више појединачних променљивих
3. Који тип података може да садржи један низ
4. Како су организовани елементи низа у меморији?
5. Које су предности коришћења низова?

ЗАДАЦИ

1. У једној библиотеци треба сачувати бројеве позајмљених књига током седам дана. Напиши програм који ће податке сместити у низ и приказати их на екрану.
2. У спортском клубу евидентира се број освојених поена на пет утакмица. Напиши програм који ће податке сачувати у низу и приказати их.
3. У метеоролошкој станици бележи се шест дневних температура. Напиши програм који ће температуре сачувати у низу и исписати их.
4. У продавници се прати продаја четири производа. Напиши програм који ће количине сачувати у низу и приказати их на екрану.
5. У школи се води евиденција о броју ученика у пет одељења. Напиши програм који ће податке сачувати у низу и исписати их.

ЗАПАМТИ ШТА СИ НАУЧИО

- Низ представља скуп елемената истог типа.
- Сви елементи низа имају заједничко име.
- Елементи су смештени у узастопним меморијским локацијама.
- Низови омогућавају организовано чување већег броја података.
- Низови представљају једну од најважнијих структура података у програмирању.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим рачунарским системима огромне количине података организују се коришћењем структура које се заснивају на низовима. Листе корисника на друштвеним мрежама, резултати претраживања на Интернету, подаци из научних експеримената и статистичке анализе често почињу управо од једноставних низова.

Иако на први поглед изгледају као једноставна структура, низови представљају темељ на којем су изграђене многе сложеније структуре које се користе у савременом програмирању и развоју софтверских система.

2.15

ДЕКЛАРИСАЊЕ И ИНИЦИЈАЛИЗАЦИЈА ЈЕДНОДИМЕНЗИОНАЛНИХ НИЗОВА

ВЕЋ ЗНАШ ДА...

- објасниш шта представља једнодимензионални низ;
- препознајеш ситуације у којима је потребно коришћење низа;
- разликујеш променљиву из низа;
- објасниш шта представља величину низа;
- препознајеш да се елементи низа нумеришу почињући од 0.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

- објасниш шта представља једнодимензионални низ;
- препознајеш ситуације у којима је потребно коришћење низа;
- разликујеш променљиву из низа;
- објасниш шта представља величину низа;
- препознајеш да се елементи низа нумеришу почињући од 0.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- декларишеш једнодимензионалне низове;
- иницијализираш низове при декларацији;
- разликујеш декларацију од иницијализације;
- користиш различите начине иницијализовања низова;
- препознајеш најчешће грешке при раду са низовима.

Пре смештања података у низ потребно је најпре створити простор у меморији рачунара. Овај поступак назива се декларисање низа. При декларисању се одређују тип података који ће бити сачувани и број елемената које ће низ садржати.

Након декларисања, елементима низа могу се доделити почетне вредности. Овај поступак назива се иницијализација. У програмском језику C++ постоји више начина за иницијализовање низова, а њихов правилан избор доприноси бољој прегледности и једноставнијем раду са подацима.

Декларисање низа представља поступак којим се у меморији резервише простор за одређени број елемената истог типа. При декларисању се обавезно наводе тип податка, име низа и његова величина. Величина одређује колико елемената низ може да садржи. Након декларисања, низ се може користити за чување и обраду података. Ако при декларисању елементима нису додељене вредности, они ће садржати неодређене вредности. Због тога је у многим ситуацијама препоручљиво одмах иницијализовати низ. На тај начин програм постаје јаснији, а могућност појаве грешака знатно се смањује.

Пример 1: Декларисање низова

*main.cpp

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

У примеру је декларисан низ под називом godine. У њему се може чувати шест целих бројева. У овом тренутку резервисан је простор у меморији, али елементима још нису додељене вредности.

Када су вредности унапред познате, могу се доделити још приликом декларисања низа. Овај поступак назива се иницијализација. При иницијализацији вредности се наводе у витичастим заградама и раздвајају запетама. Број вредности најчешће је једнак величини низа.

Овакав начин рада често се користи при тестирању програма или када су подаци су унапред познати. Иницијализирани низови омогућавају да програм одмах располаже подацима који могу да буду обрађени и анализирани.

Пример 2: Иницијализација низа

main.cpp

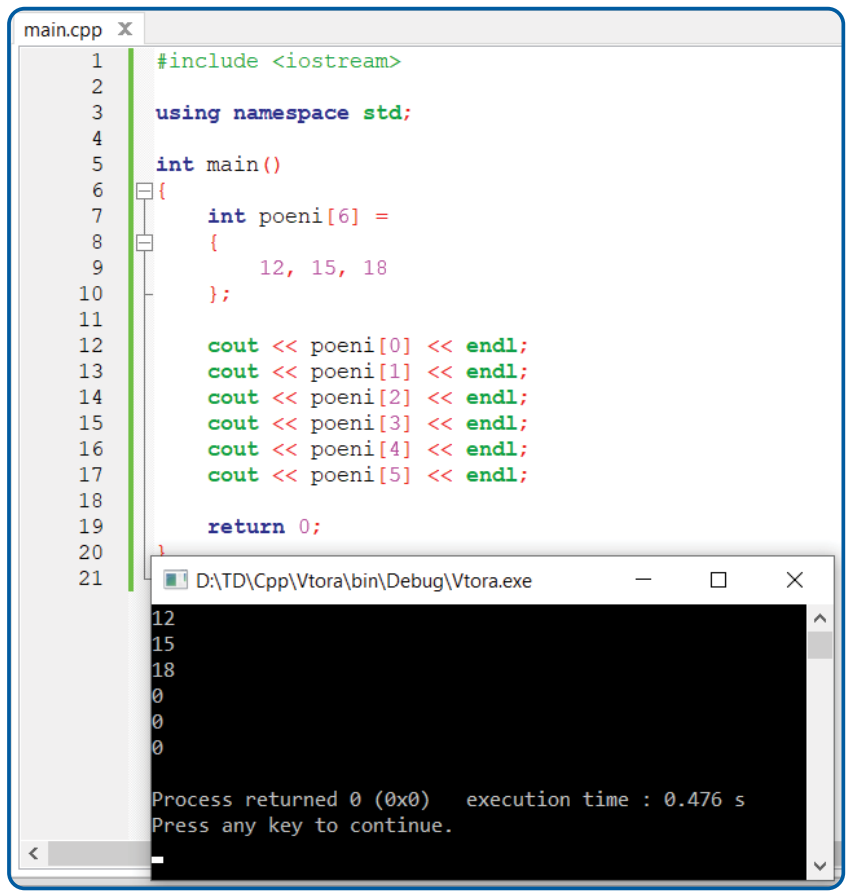
```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       int temperature [7] =
8       {
9           18, 20, 21, 19, 22, 24, 23
10      };
11
12      cout << temperature[0] << endl;
13
14      return 0;
15  }
16
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
18

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

У примеру је низ иницијализован приликом декларисања. Сваком елементу додељена је почетна вредност и низ је одмах спреман за употребу.

При иницијализацији низова није неопходно задати вредности за све елементе. Ако се наведе мање вредности од величине низа, први елементи добиће задате вредности, а преостали ће бити постављени на нулу. Овај начин назива се делимична иницијализација. Такав приступ може бити користан када је у тренутку креирања низа познат само део података. Ипак, при раду са оваквим низовима треба пажљиво проверити који елементи садрже стварне податке, а којима је вредност аутоматски додељена. Познавање начина иницијализације важно је јер директно утиче на резултате добијене након извршавања програма.

Пример 3: Делимична иницијализација низа



У примеру прва три елемента добијају вредности 12, 15 и 18. Преостала три елемента аутоматски добијају вредност 0. Захваљујући овој могућности, није потребно увек наводити вредности за све елементе низа.

У програмском језику C++ величина низа може се аутоматски одредити на основу броја наведених вредности. У таквим ситуацијама број елемената не наводи се у угластим заградама. Компајлер самостално одређује број елемената на основу броја задатих вредности. Овакав приступ побољшава читљивост програма и смањује могућност грешака при бројању елемената. Ипак, овај начин рада погодан је само када су све вредности познате већ при декларисању низа. Ако се подаци уносе касније, величина низа мора бити унапред одређена. Зато избор начина декларисања зависи од конкретног проблема који се решава и доступности података у тренутку креирања низа.

```
    Аутоматско одређивање величине int
    int meseci[] =
{
   31, 28, 31, 30,
   31, 30, 31, 31,
   30, 31, 30, 31
};
```

У овом примеру величина низа није наведена. Пошто је унето дванаест вредности, компајлер аутоматски креира низ од дванаест елемената.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља декларацију низа?
2. Шта представља иницијализацију низа?
3. У чему је разлика између декларације и иницијализације?
4. Шта се дешава са елементима који нису добили вредност при делимичној иницијализацији?
5. У којим ситуацијама може да се користи аутоматско одређивање величине низа?

ЗАДАЦИ

1. Напиши програм у коме ће бити декларисан низ са десет целих бројева без иницијализације.
2. Напиши програм у коме ће бити иницијализован низ са седам температура измерених током једне седмице.
3. Напиши програм у коме ће бити креиран низ са пет вредности које представљају резултате једног спортисте на пет утакмица.
4. Напиши програм у коме ће бити употребљена делимично иницијализиран низ са осам елемената, при чему ће бити задате само прве четири вредности.
5. Напиши програм у коме ће величина низа бити аутоматски одређена на основу броја унетих вредности.

ЗАПАМТИ ШТА СИ НАУЧИО

- Декларација резервише простор у меморији за елементе низа.
- Иницијализација додељује почетне вредности елемената.
- Величина низа одређује број елемената који могу бити сачувани.
- При делимичној иницијализацији преостали елементи добијају вредност 0.
- Величина низа може аутоматски бити одређена када су наведене све вредности при декларацији.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим програмским језицима постоје структуре које могу аутоматски да мењају своју величину током извршавања програма. Такве структуре омогућавају додавање и уклањање елемената без потребе да њихов број буде унапред одређен. Ипак, класични низови остају једна од најбржих и најефикаснијих структура за складиштење података. Због своје једноставности и ефикасности, они представљају основу за велики број сложенијих структура које се користе у савременом програмирању.

2.16

ИНДЕКС ЕЛЕМЕНТА И ПРИСТУПАЊЕ ЕЛЕМЕНТИМА НИЗА

ВЕЋ ЗНАШ ДА...

- објасниш шта представља једнодимензионални низ;
- декларишеш низ;
- иницијализираш низ;
- разликујеш декларацију од иницијализације;
- препознајеш да се елементи низа нумеришу почињући од 0.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта представља декларација низа?
2. Шта представља иницијализација низа?
3. Колико елемената садржи низ int brojevi[8];?
4. Која је позиција последњег елемента у низу са величином 10?
5. Шта се дешава са осталим елементима при делимичној иницијализацији?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објасниш шта представља индекс елемента;
- приступаш појединачним елементима низа;
- мењаш вредности елемената у низу;
- препознајеш невалидне индексе;
- избегаваш најчешће грешке при раду са индексима.

Иако су сви елементи једног низа повезани заједничким именом, сваком елементу мора се моћи појединачно приступити и обрадити га. У ту сврху користе се индекси. Индекс одређује позицију елемента у низу и омогућава програму да приступи одређеној вредности.

Правилно коришћење индекса представља основу за успешан рад са низовима. Велики број програмских грешака настаје управо због неправилног коришћења индекса. Због тога је потребно пажљиво разумети како функционише нумерисање елемената и које вредности представљају дозвољене индексе.

Сваки елемент низа има своју позицију која се назива индекс. Индекс омогућава одређивање конкретног елемента с којим ће се радити. У програмском језику C++ нумерисање почиње од 0. Због тога први елемент има индекс 0, други индекс 1, трећи индекс 2 и тако даље. Ако низ садржи пет елемената, последњи елемент нема индекс 5, већ индекс 4. Ово правило важи за све низове, без обзира на њихову величину. Елементима се приступа навођењем имена низа и индекса у угластим заградама. На тај начин програм тачно зна с којим елементом треба да ради.

Пример 1: Приступање елементу из низа

```
#include <iostream>

using namespace std;

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

   cout << poeni[2] << endl;

   return 0;
}
```

У примеру је приказан елемент са индексом 2. Пошто нумерисање почиње од 0, то је трећи елемент низа и његова вредност је 20.

Поред читања вредности, индекси омогућавају и мењање елемената низа. Да би се променила вредност одређеног елемента, потребно је навести његов индекс и доделити му нову вредност. На тај начин појединачни елементи могу се мењати независно, без утицаја на остале елементе низа. Ова могућност посебно је важна при обради података јер омогућава ажурирање конкретних информација без потребе за поновним креирањем целог низа.

Пример 2: Промена вредности у низу

```
#include <iostream>

using namespace std;

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

   zaliha[1] = 22;

   cout << zaliha[1] << endl;

   return 0;
}
```

У примеру се вредност елемента са индексом 1 мења са 18 на 22.

При раду са низовима посебно је важно користити дозвољене индексе. Сваком елементу може се приступити само помоћу индекса који припада низу. Ако низ садржи пет елемената, дозвољени индекси су 0, 1, 2, 3 и 4. Покушај приступања елементу са индексом 5 или већим представља грешку. Такође, није дозвољено коришћење негативних индекса.

Овакве грешке могу проузроковати неправилан рад програма или приказивање неочекиваних резултата. Због тога увек треба пажљиво проверити да ли се коришћени индекс налази у дозвољеном опсегу. Разумевање овог правила представља један од најважнијих предуслова за успешан рад са низовима.

Пример 3: Приступање већем броју елемената

У спортском клубу евидентирани су бодови освојени на пет такмичења. Приказати први, трећи и последњи резултат.

```
#include <iostream>

using namespace std;

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

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

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

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

    return 0;
}
```

У примеру се приступа трима различитим елементима истог низа. Први елемент има индекс 0, трећи индекс 2, а последњи индекс 4. То показује да позиција елемента није увек једнака његовом редном броју у свакодневном бројању.

Индекси представљају основни механизам за приступ елементима низова. Без њих не би било могуће читати, мењати или обрађивати појединачне вредности. Због тога при раду са низовима увек треба знати њихову величину и дозвољени опсег индекса. Након савладавања приступа појединачним елементима може се прећи на унос и обраду већег броја података, што представља следећи корак у изучавању једнодимензионалних низова.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља индекс елемента?
2. Од ког броја почиње нумерисање елемената у низу?
3. Који је индекс последњег елемента у низу величине 12?
4. Како се приступа елементу низа?
5. Зашто је важно користити само дозвољене индексе?

ЗАДАЦИ

1. У једној библиотеци евидентирано је пет дневних посета ученика са бројевима чланских карата: 35, 42, 38, 50 и 47. Напиши програм који ће приказати први и последњи елемент низа!
2. У продавници се прати продаја шест производа. Напиши програм који ће податке сачувати у низу и приказати четврти елемент!
3. На спортском турниру забележени су освојени поени једне екипе на пет утакмица. Напиши програм који ће променити вредност друге утакмице и приказаће нову вредност!
4. У метеоролошкој станици сачувано је седам температура у низу. Напиши програм који ће приказати температуре првог, трећег и шестог дана!
5. У школској евиденцији сачувани су бројеви ученика у пет одељења. Напиши програм који ће променити вредност последњег елемента и приказати резултат!

ЗАПАМТИ ШТА СИ НАУЧИО

- Индекс одређује позицију елемента у низу.
- Нумерисање елемената почиње од 0.
- Последњи елемент има индекс једнак величини низа умањеној за 1.
- Индекси се користе за читање и измену елемената.
- Треба да се користе само индекси који припадају низу.
-

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У многим програмским језицима, као што су C++, Java и C#, нумерисање елемената низова почиње од 0. Ово правило потиче од начина на који рачунар израчунава меморијске адресе елемената. Иако у почетку може изгледати необично, овакав начин нумерисања омогућава једноставније и брже прорачуне приликом приступа подацима. Управо зато је индексирање од 0 прихваћено као стандард у великом броју савремених програмских језика.

2.17

УНОШЕЊЕ И ШТАМПАЊЕ ЕЛЕМЕНАТА НИЗА

ВЕЋ ЗНАШ ДА...

- декларишеш једнодимензионални низ;
- иницијализираш низ;
- приступаш елементима преко индекса;
- мењаш вредности елемената у низу;
- користиш циклус for.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта представља индекс елемента?
2. Од ког броја започиње нумерисање елемената у низу?
3. Који је индекс последњег елемента у низу са величином 15?
4. Како се мења вредност једног елемента у низу?
5. Зашто не треба да се користе индекси ван дозвољеног опсега?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- уносиш вредности у низу;
- користиш циклус за унос елемената;
- приказујеш елементе низа;
- комбинујеш низе и циклусе;
- избегаваш најчешће грешке при уношењу података.

Једна од најважнијих предности низова јесте могућност обраде великог броја података помоћу малог броја програмских инструкција. Ако је потребно унети десет, двадесет или сто вредности, није практично користити посебну променљиву за сваку вредност. Уместо тога, подаци се могу сместити у низ. При раду са низовима

веома често је потребно да корисник унесе податке. У ту сврху користе се циклуси који омогућавају унос више елемената понављањем истих инструкција. На сличан начин може приказати и садржај низа.

При уносу података у низ сваки елемент треба да добије вредност. Уместо писања посебне наредбе за сваки елемент, најчешће се користи циклус for. Бројач циклуса представља позицију елемента који се у том тренутку обрађује. На тај начин могу се лако унети сви елементи, без обзира на то да ли низ садржи пет, десет или сто вредности. Комбинација низова и циклуса представља једну од најважнијих техника у програмирању јер омогућава ефикасну обраду већег броја података.

Пример 1: Унос пет бројева у низ

main.cpp

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

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Unesi broj: 1
Unesi broj: 2
Unesi broj: 5
Unesi broj: 3
Unesi broj: 4

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

У примеру се циклус извршава пет пута. При сваком извршавању уноси се вредност у други елемент низа.

Након уноса података често је потребно приказати их. И у овој ситуацији користи се циклус for. Уместо писања посебне наредбе за сваки елемент, циклус редом приступа свим позицијама у низу. Овакав приступ чини програм краћим и разумљивијим. Поред тога, исти кôд може се применити и на много веће низове без значајних измена.

Пример 2: Приказивање елемената низа

main.cpp

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int brojevi [5] =
8      {
9          10, 20, 30, 40, 50
10     };
11
12     for(int i = 0; i < 5; i++)
13     {
14         cout << brojevi [i] << " ";
15     }
16
17     return 0;
18 }
19
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
10 20 30 40 50
Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

У примеру се сви елементи низа приказују помоћу циклуса. Без циклуса било би потребно пет посебних наредби cout.

У практичним програмским решењима најчешће није довољно само унети податке. Након уноса обично следи њихова обрада, провера или приказивање. Због тога се у једном програму веома често користе два циклуса. Први циклус служи за унос података у низ, а други за њихово приказивање. Овакав приступ омогућава да сви подаци најпре буду сачувани у меморији, а затим обрађени на различите начине. Са повећањем броја елемената предности овакве организације постају још израженије. Уместо писања десетина или стотина наредби за унос и приказивање, потребна су само два кратка циклуса. Управо зато комбинација низова и циклуса представља једну од најчешће коришћених техника у програмирању.

Пример 3: Уношење и приказивање броја посетилаца

У музеју се евидентира број посетилаца током пет дана. Унети податке, а затим их приказати.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int posetioci[5];
    for(int i = 0; i < 5; i++)
    {
        cout << "Posetioci za dan "
             << i + 1
             << ": ";

        cin >> posetioci[i];
    }
    cout << endl;
    cout << "Uneti podaci:" << endl;
    for(int i = 0; i < 5; i++)
    {
        cout << "Dan "
             << i + 1
             << ": "
             << posetioci[i]
             << endl;
    }
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vt
Posetioci za dan 1: 21
Posetioci za dan 2: 45
Posetioci za dan 3: 8
Posetioci za dan 4: 105
Posetioci za dan 5: 55

Uneti podaci:
Dan 1: 21
Dan 2: 45
Dan 3: 8
Dan 4: 105
Dan 5: 55

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

У примеру први циклус уноси податке у низ, а други их приказује. На тај начин све вредности могу се лако проверити након завршетка уноса.

При раду са низовима и циклусима посебно је важно правилно одредити број понављања. Ако се циклус извршава више пута него што има елемената у низу, доћи ће до покушаја приступа непостојећим елементима. С друге стране, ако се циклус извршава мање пута, део елемената неће бити обрађен. Због тога горња граница циклуса треба да буде усклађена са величином низа. Ово правило ће се примењивати у наставку како би се обављали различити прорачуни и анализе.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Зашто се најчешће користи циклус при уношењу елемената у низу?
2. У чему је улога бројача у циклусу?
3. Зашто је важно да број понављања одговара величини низа?
4. Које су предности коришћења циклуса за приказивање елемената?
5. Зашто се низе и циклуси често користе заједно?

ЗАДАЦИ

1. На археолошком налазишту пронађено је седам предмета различите старости, изражене у годинама. Напиши програм који ће унети њихове старости у низ, а затим их приказати редоследом којим су унете!
2. У астрономској опсерваторији бележи се број уочених метеора током десет ноћи. Напиши програм који ће податке унети у низ, а затим приказати све вредности!
3. У фабрици бицикала евидентира се број произведених бицикала током шест радних смена. Напиши програм који ће унети податке у низу и приказаће их на екрану!
4. У националном парку прати се број посетилаца на пет различитих туристичких стаза. Напиши програм који ће податке унети у низ, а затим их приказати уз поруку за сваку стазу!
5. У свемирској станици мере се количине воде (у литрима) потрошене током осам узастопних дана. Напиши програм који ће вредности унети у низ и приказати их након завршетка уноса!

ЗАПАМТИ ШТА СИ НАУЧИО

- Уношење елемената у низу се најчеше врши са циклусом for.
- Бројач циклуса се користи као индекс елемената.
- Приказивање елемената исто тако може да се реализује циклусом.
- Величина низа треба да буде усаглашена са бројем понављања циклуса.
- Комбинација низа и циклуса омогућава ефикасну обраду великог броја података.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У великом броју савремених апликација подаци пристижу из различитих извора, као што су сензори, интернет услуге, базе података или мобилни уређаји. Без обзира на њихово порекло, они се најчешће чувају у структурама које омогућавају истовремену обраду више вредности. Низови представљају једну од најједноставнијих таквих структура. Захваљујући циклусима, хиљаде података могу се обрадити помоћу релативно малог броја програмских инструкција. Ова могућност представља основу за развој система за статистичку анализу, научна истраживања, обраду слика, машинско учење и многе друге савремене области информатике.

2.18

ОСНОВНЕ ОПЕРАЦИЈЕ СА ЕЛЕМЕНТИМА НИЗА

ВЕЋ ЗНАШ ДА...

- декларишеш и иницијализираш низ;
- уносиш и приказујеш елементе низа;
- приступаш елементима преко индекса;
- користиш циклус for;
- комбинујеш низе и циклусе.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта представља индекс елемента?
2. Од ког броја започиње нумерисање елемената?
3. Зашто се при раду са низовима користи циклус for?
4. Како се уноси више елемената у низу?
5. Шта ће се догодити ако се индекс употреби ван граница низа?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- израчунаваш суму елемената низа;
- израчунаваш производ елемената низа;
- користиш додатне променљиве при обради података;
- комбинујеш аритметичке операције и низе;
- решаваш практичне задатке обрадом свих елемената.

У великом броју програмских решења није довољно само унети и приказати податке. Често је потребно израчунати њихову укупну вредност, производ свих вредности или неки други заједнички резултат. На пример, може бити потребно израчунати укупну количину произведеног меда, укупан број путника или укупну дужину путовања састављеног од више деоница.

За решавање оваквих проблема најчешће се користи циклус који редом обрађује све елементе низа. При сваком проласку кроз циклус ажурира се резултат израчунавања. На тај начин велики број података може се лако обрадити помоћу релативно малог броја програмских инструкција.

При обради елемената низа често је потребно израчунати њихову укупну вредност. У ту сврху користи се додатна променљива којој се на почетку додељује вредност 0. Затим се при сваком проласку кроз циклус на њу додаје тренутни елемент низа. Након обраде последњег елемента, променљива садржи укупан збир свих вредности. Овакав начин рада примењује се у великом броју практичних ситуација и представља једну од основних техника за обраду података у низовима.

Пример 1: Производња меда

У пчелињаку су измерене количине меда које су произведене у пет кошница. Израчунај укупну призводњу меда!

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int med[5]={18,22,20,25,19};

    int zbir=0;

    for(int i=0;i<5;i++)
    {
        zbir = zbir + med[i];
    }
    cout << "Ukupna proizvodnja: " << zbir << " kg";
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Ukupna proizvodnja o: 104 kg
Process returned 0 (0x0)   execution time : 0.164 s
Press any key to continue.
```

У примеру се све количине произведеног меда постепено сабирају. Након завршетка циклуса добија се укупна количина произведеног меда.

Није увек потребно израчунати збир. У одређеним ситуацијама треба израчунати производ више вредности. У таквим задацима користи се додатна променљива којој се на почетку додељује вредност 1. Разлог за то је што множење бројем 1 не мења вредност бројева, док множење нулом увек даје резултат 0. При сваком проласку кроз циклус текући елемент множи се претходно добијеним резултатом.

Пример 2: Укупан коефицијент преноса

У научном експерименту је измерено пет коефицијената појачања. Израчунај њихов производ!

main.cpp

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

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Proizvod je :: 240
Process returned 0 (0x0)   execution time : 1.718 s
Press any key to continue.
```

У примеру се променљива proizvod постепено ажурира при обради сваког елемента низа. Након завршетка циклуса добија се производ свих вредности.

При обради података сачуваних у низу није увек потребно израчунати само збир или производ. У великом броју практичних ситуација потребно је над сваким елементом извршити одређени прорачун, а добијене резултате постепено комбиновати. Овакав начин рада омогућава решавање различитих проблема из свакодневног живота и различитих научних области.

Без обзира на врсту прорачуна, основна идеја остаје иста. Циклус редом обрађује све елементе низа, а над сваким елементом примењује се потребна операција. На тај начин могу се израчунати укупна потрошња, произведена енергија, новчани износ или многе друге вредности.

Пример 3: Потрошња воде

У истраживачком кампу евидентира се дневна потрошња воде током пет експедиција. Цена једног литра воде износи 2 денара. Израчунати укупан износ који треба платити.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int voda[5]={120,150,110,140,130};
    int ukupno = 0;
    for(int i=0;i<5;i++)
    {
        ukupno+=voda[i]*2;
    }
    cout << "Ukupan iznos : "<<ukupno<<" denari"<<endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Ukupan iznos : 1300 denari

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

У примеру се не сабирају само вредности из низа. За сваки елемент најпре се израчунава његова цена, а затим се та вредност додаје укупном износу. На тај начин комбинују се аритметичке операције и обрада елемената низа. При обради низова избор операције која ће бити извршена зависи од проблема који се решава. У неким ситуацијама израчунава се збир, у другим производ, а у трећим одређени математички израз за сваки елемент. Заједничка карактеристика свих ових задатака јесте то што се сваки елемент низа обрађује тачно једном. Захваљујући томе, програми остају једноставни и лаки за разумевање, чак и када обрађују већи број података. Овакав начин обраде представља основу за велики број алгоритама који ће се изучавати у наредним лекцијама.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Зашто се при обради низа најчешће користи циклус for?
2. Којом почетном вредношћу најчешће започиње променљива за суму?
3. Зашто променљива за производ најчешће започиње вредношћу 1?
4. Да ли се над сваким елементом низа може извршити различита аритметичка операција?
5. Која је предност обраде свих елемената једним циклусом?

ЗАДАЦИ

1. У ветропарку су измерене количине произведене електричне енергије из шест ветрогенератора. Напиши програм који ће израчунати укупну произведену енергију.
2. У фабрици се производе делови који се пакују у групе. Напиши програм који ће израчунати производ бројева група сачуваних у једном низу.
3. У еколошкој акцији евидентира се број засађених стабала у пет насељених места. За свако стабло издваја се по 50 динара за одржавање. Напиши програм који ће израчунати укупан потребан износ.
4. У роботској лабораторији бележи се број успешно извршених задатака четири робота. Напиши програм који ће израчунати укупан број успешно извршених задатака.
5. У истраживачкој станици мери се количина прикупљених узорака током седам дана. За сваки узорак додељују се по 3 поена. Напиши програм који ће израчунати укупан број освојених поена.

ЗАПАМТИ ШТА СИ НАУЧИО

- Елементи једног низа могу се редом обрадити помоћу циклуса for.
- За израчунавање укупне вредности најчешће се користи променљива за суму.
- За израчунавање производа користи се променљива која започиње вредношћу 1.
- Над сваким елементом низа може се применити аритметичка операција.
- Обрада свих елемената једним циклусом омогућава једноставна и ефикасна програмска решења.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим информационим системима често се обрађују велике количине података. На пример, могу се анализирати мерења са сензора, финансијске трансакције, научни експерименти или статистички подаци. Иако такви системи раде са хиљадама или милионима вредности, основна идеја остаје иста – подаци се редом обрађују, а над сваким елементом извршавају се потребне операције. Познавање ових основних техника представља важну основу за изучавање сложенијих алгоритама и структура података.

2.19

УНОС ЕЛЕМЕНАТА У НИЗ И ОСНОВНА ИЗРАЧУНАВАЊА

ВЕЋ ЗНАШ ДА...

- декларишеш и иницијализираш једнодимензионалне низе;
- приступаш елементима низа;
- користиш циклус for;
- израчунаваш суму и производ унапред задатих елеменатаи;
- обрађујеш елементе низа помоћу циклуса.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Како се декларише једнодимензионални низ?
2. Од ког броја започиње нумерисање елемената?
3. Зашто се при раду са низовима најчешће користи циклус for?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- уносиш елементе у низ;
- обрађујеш унете податке;
- израчунаваш суму унетих вредности;
- израчунаваш производ унетих вредности;
- израчунаваш аритметичку средину унетих података

У великом броју програмских решења подаци нису унапред познати. Уместо да буду унети у програм приликом његовог писања, они се уносе током извршавања. На тај начин исти програм може се користити за обраду различитих скупова података.

При раду са низовима подаци се најчешће уносе са тастатуре помоћу циклуса. Након уноса над њима се могу извршити различити прорачуни, као што су одређивање збира, производа или аритметичке средине. Овакав начин рада представља основу за велики број практичних програмских задатака.

При уносу елемената у низ најчешће се користи циклус for. Бројач циклуса одређује позицију елемента који се у том тренутку уноси. Након уноса свих вредности, оне се могу обрадити помоћу другог циклуса или током самог уноса. Захваљујући томе, велики број података може се брзо и ефикасно обрадити. Овакав приступ има широку примену у различитим информационим системима и представља једну од основних техника при раду са једнодимензионалним низовима.

Пример 1: Укупна дужина бициклистичких стаза

У националном парку уносе се дужине пет бициклистичких стаза. Израчунати њихову укупну дужину.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int staze[5];
    int zbir = 0;

    for(int i = 0; i < 5; i++)
    {
        cout << "Dužina staze     "<<i+1<< ": ";
        cin >> staze[i];
        zbir += staze[i];
    }
    cout << "Ukupna dužina : "<<zbir <<" km"<< endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Dužina staze     1: 14
Dužina staze     2: 7
Dužina staze     3: 2
Dužina staze     4: 9
Dužina staze     5: 18
Ukupna dužina  a: 50 km

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

У примеру се подаци уносе са тастатуре, а при сваком уносу нова вредност додаје се укупном збиру.

Након завршетка циклуса добија се укупна дужина свих стаза.

У практичним програмским решењима није увек потребно израчунати збир. У неким ситуацијама потребно је одредити производ више вредности. У таквим случајевима користи се додатна променљива којој се на почетку додељује вредност 1. При сваком уносу нова вредност множи се претходним резултатом. На тај начин постепено се добија производ свих унетих елемената.

Пример 2: Производ фактора

У научној лабораторији се уноси пет фактора појачања. Израчунај њихов производ!

main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
    int faktor[5];
    int proizvod = 1;
    for(int i = 0; i < 5; i++)
    {
        cout<<"Faktor "<<i+1<<": ";
        cin>> faktor[i];
        proizvod*=faktor[i];
    }
    cout<<"Proizvod: "<<proizvod<< endl;
    return 0;
}
```

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

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

При обради унетих података често је потребно одредити њихову просечну вредност. У ту сврху најпре треба израчунати укупан збир свих елемената низа. Затим се добијени резултат дели бројем унетих елемената. Пошто резултат дељења може садржати децимални део, за израчунавање аритметичке средине најчешће се користи променљива типа double. На тај начин добија се прецизнији резултат и избегава губитак децималних вредности. Овакав начин рада има широку примену у статистичким анализама, научним истраживањима и обради различитих врста података.

Пример 3: Просечна дневна изложеност сунцу

У астрономској опсерваторији бележи се број часова сунчеве светлости током пет дана. Израчунати укупан број часова и просечну дневну изложеност сунчевој светлости.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int sati [5];
    int zbir = 0;
    double prosek;
    for(int i = 0; i < 5; i++)
        cout << "Sati za dan " << i+1 << ": ";
        cin >> sati[i];
        zbir += sati[i];
    }
    cout << "Ukupno sati: " << zbir << endl;
    cout << "Prosečna vrednost: " << prosek << endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Sati za dan 1: 2
Sati za dan 2: 3
Sati za dan 3: 4
Sati za dan 4: 1
Sati za dan 5: 3
Ukupno sati: 13
Prosečna vrednost: 2.6

Process returned 0 (0x0)   execution time : 6.037 s
Press any key to continue.
```

У примеру се подаци најпре уносе у низ. Током уноса постепено се израчунава њихов збир. Након завршетка циклуса одређује се аритметичка средина дељењем укупног збира бројем елемената. За чување резултата користи се променљива типа double, која омогућава приказивање и децималног дела.

При обради података које је корисник унео могу се извршити различити прорачуни, у зависности од постављеног проблема. У неким ситуацијама израчунава се збир, у другим производ, а у трећим аритметичка средина. Заједничка карактеристика свих ових задатака јесте то што се подаци најпре уносе у низ, а затим обрађују помоћу циклуса. Захваљујући овом поступку, један програм може се користити за обраду различитих скупова података без потребе за његовим мењањем. Овакав приступ представља основу за велики број програмских решења која ће се изучавати у наредним лекцијама.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Зашто се при уносу више елемената у низ користи циклус for?
2. Зашто се при израчунавању аритметичке средине често користи тип double?
3. Која је предност уноса података са тастатуре уместо њиховог унапред задавања?

ЗАДАЧИ

1. У ботаничкој башти уносе се висине шест новозасађених стабала. Напиши програм који ће израчунати њихову укупну висину.
2. У центру за роботику уноси се број успешно завршених задатака пет робота. Напиши програм који ће израчунати производ унетих вредности.
3. У океанографској станици уноси се пет дневних мерења висине таласа. Напиши програм који ће израчунати њихову аритметичку средину.
4. У археолошкој експедицији уноси се број пронађених предмета на седам локалитета. Напиши програм који ће израчунати укупан број предмета.
5. У центру за обновљиву енергију уноси се произведена енергија из пет соларних система. Напиши програм који ће израчунати укупну и просечну произведену енергију.

ЗАПАМТИ ШТА СИ НАУЧИО

- Елементи низа могу се уносити са тастатуре помоћу циклуса for.
- Унети подаци могу одмах да се обраде.
- За израчунавање укупне вредности користи се променљива за суму.
- За израчунавање производа користи се променљива са почетном вредношћу 1.
- Аритметичка средина добија се дељењем укупне суме бројем елемената.
- При израчунавању просека најчешће се користи тип double.
-

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим информационим системима велики део података није унапред познат, већ се уноси током рада програма. На пример, мерења са сензора, финансијске трансакције, медицинске анализе и спортски резултати непрестано се допуњују новим информацијама. Због тога програми треба да буду способни да примају податке од корисника и одмах их обрађују. Управо ова могућност представља једну од најважнијих примена једнодимензионалних низова и циклуса у савременом програмирању.

2.20

ОДРЕЂИВАЊЕ НАЈВЕЋЕГ И НАЈМАЊЕГ ЕЛЕМЕНТА У НИЗУ

ВЕЋ ЗНАШ ДА...

- декларишеш и иницијализираш једнодимензионални низ;
- уносиш елементе у низу;
- обрађујеш податке циклусом for;
- израчунаваш суму, производ и аритметичку средину;
- комбинујеш унос и обраду података.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Како се уноси више елемената у низ?
2. Зашто се при раду са низовима најчешће користи циклус for?
3. Којом почетном вредношћу започиње променљива за суму?
4. Како се израчунава аритметичка средина?
5. Зашто је корисно да се подаци обраде одмах након њиховог уноса?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- одређујеш највећи елемент у низу;
- одређујеш најмањи елемент у низу;
- упоређујеш елементе из низа;
- користиш додатне променљиве при поређењу;
- обрађујеш податке применом условних конструкција.

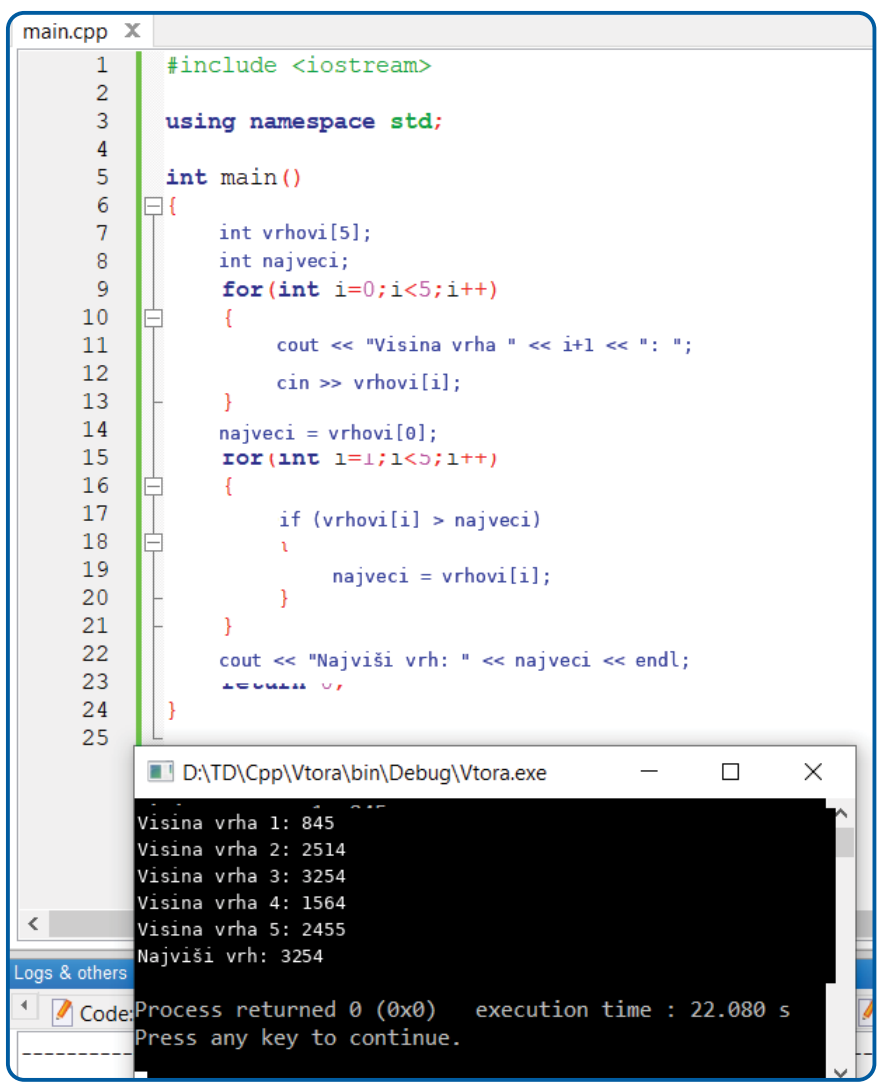
При обради података често је потребно одредити највећу или најмању вредност у одређеном скупу података. На пример, може бити потребно одредити највећу измерену висину, најмању количину падавина или највећу постигнуту брзину.

За решавање оваквих задатака користи се поређење елемената низа. Најпре се једна вредност бира као почетна, а затим се остали елементи редом пореде с њом. Ако се пронађе већа или мања вредност, почетна вредност замењује се новом.

При одређивању највећег или најмањег елемента потребно је најпре изабрати једну вредност из низа као почетну. Најчешће је то први елемент низа. Затим се помоћу циклуса редом разматрају остали елементи. Ако је неки од њих већи од тренутно запамћене највеће вредности, она се замењује. На исти начин може се одредити и најмања вредност. Захваљујући овом поступку, сви подаци обрађују се једним проласком кроз низ.

Пример 1: Највши планински врх

У планинарски водич уносе се надморске висине пет планинских врхова. Одредити највиши врх.

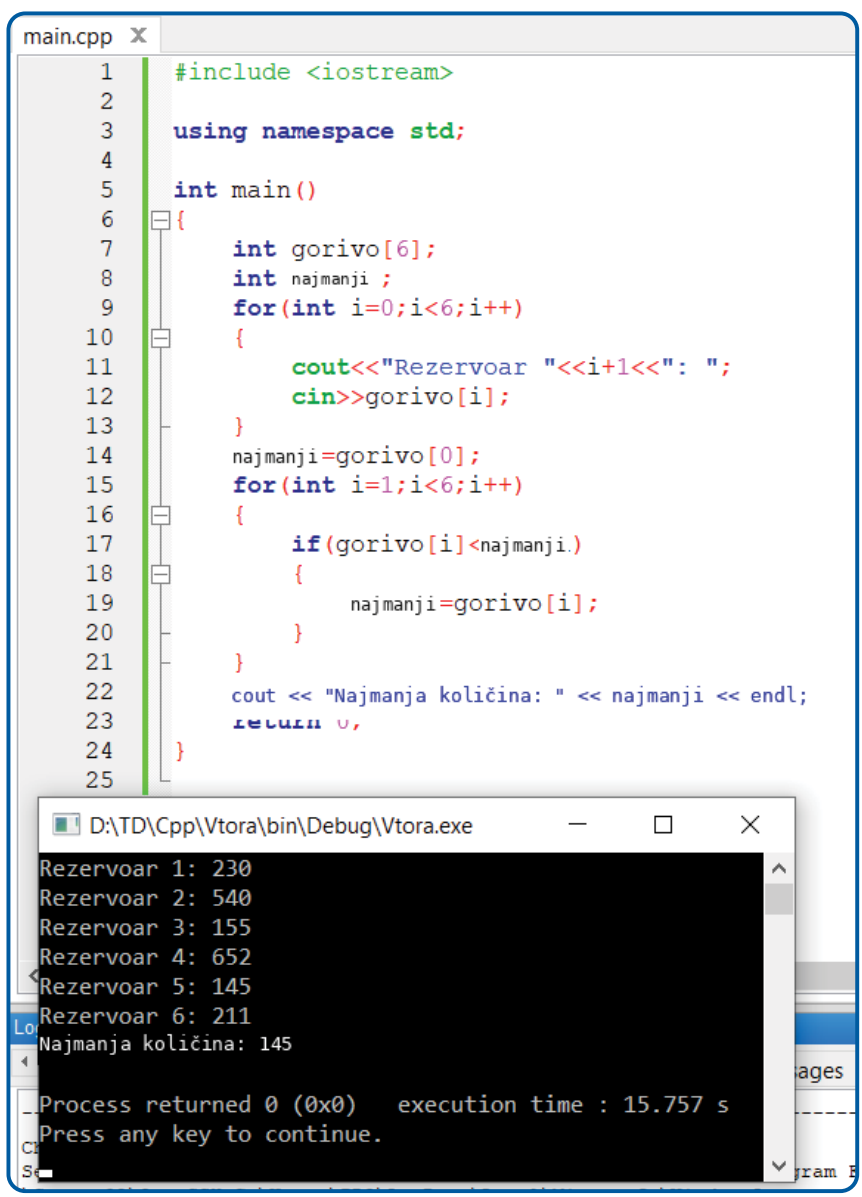


У примеру се први елемент бира као почетна вредност. Затим се сви остали елементи пореде с њим. Ако се пронађе већа вредност, она постаје нова највећа вредност.

При одређивању најмањег елемента користи се исти поступак. Разлика је у томе што се, уместо тражења веће вредности, проверава да ли је текући елемент мањи од тренутно запамћене најмање вредности. На тај начин може се одредити најмања вредност у једном скупу података без додатних сложених прорачуна

Пример 2: Најмања количина горива

У истраживачкој експедицији уносе се количине горива у шест резервоара. Одредити резервоар са најмањом количином горива.



У примеру се примењује иста програмска идеја, али се овога пута одређује најмања вредност међу унетим подацима.

У практичним програмским задацима често је потребно истовремено одредити и највећу и најмању вредност у једном скупу података. При томе није потребно два пута обрађивати низ. Током једног проласка кроз елементе могу се истовремено ажурирати и највећа и најмања вредност. Након завршетка обраде над добијеним резултатима могу се извршити и додатни прорачуни. Један од њих је одређивање распона података, који се добија као разлика између највеће и најмање вредности.

Пример 3: Распон измерених вредности

У научној станици уноси се шест измерених вредности. Одредити највећу и најмању вредност, као и распон мерења.

main.cpp

```
#include <iostream>
using namespace std;
int main()
{
    int merenja[6];
    int najveci;
    int najmanji;
    int raspon;
    for(int i=0;i<6;i++)
    {
        cout << "Merenje " << i+1 << ": ";
        cin >> merenja[i];
    }
    najveci = merenja[0];
    najmanji = merenja[0];

    for(int i=1;i<6;i++)
    {
        if (merenja[i] > najveci)
        {
            najveci = merenja[i];
        }
        if (merenja[i] < najmanji)
        {
            najmanji = merenja[i];
        }
    }
    raspon = najveci - najmanji;
    cout << "Najveća vrednost: " << najveci << endl;
    cout << "Najmanja vrednost: " << najmanji << endl;
    cout << "Raspon: " << raspon << endl;
    -------- 0;
}
```

```
D:\TD...
Merenje 1: 55
Merenje 2: 24
Merenje 3: 68
Merenje 4: 144
Merenje 5: 65
Merenje 6: 42
Najveća vrednost: 144
Najmanja vrednost: 24
Raspon: 120

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

У примеру се једним проласком кроз низ одређују и највећа и најмања вредност. Затим се помоћу њих израчунава распон података.

При обради података одређивање највећих и најмањих вредности има широку примену. На пример, може се одредити највиша температура, најмања потрошња енергије или највећа измерена брзина. Без обзира на врсту података, основни поступак остаје исти. Најпре се бира почетна вредност, затим се сви остали елементи редом пореде с њом, а резултати поређења ажурирају се по потреби.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Зашто се при одређивању највеће вредности као почетна најчешће бира први елемент?
2. Како се одређује најмањи елемент у низу?
3. Да ли се највећа и најмања вредност могу одредити једним проласком кроз низ?
4. Шта представља распон података?
5. Зашто се у овим задацима најчешће користи циклус for?

ЗАДАЦИ

1. У метеоролошкој станици уносе се количине снега измерене у пет планинских центара. Напиши програм који ће одредити највећу измерену количину.
2. У центру за заштиту животиња уносе се тежине шест младих медведа. Напиши програм који ће одредити најмању измерену тежину.
3. У океанографској експедицији уносе се дубине седам мерења. Напиши програм који ће одредити највећу и најмању дубину.
4. У истраживању квалитета ваздуха уносе се дневне вредности са пет мерних станица. Напиши програм који ће израчунати распон измерених вредности.
5. У археолошкој експедицији уносе се старости шест пронађених предмета. Напиши програм који ће одредити највећу и најмању старост и разлику између њих.

ЗАПАМТИ ШТА СИ НАУЧИО

- Највећа и најмања вредност могу се одредити поређењем елемената низа.
- Почетна вредност најчешће је први елемент низа.
- При проналажењу веће или мање вредности резултат се ажурира.
- Највећа и најмања вредност могу се одредити једним проласком кроз низ.
- Распон података добија се као разлика између највеће и најмање вредности.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Одређивање највеће и најмање вредности представља један од најчешће коришћених поступака при анализи података. У науци се може одредити највиша измерена температура, у медицини највећа измерена вредност одређеног показатеља, а у спорту најбољи постигнути резултат. Иако савремени информациони системи обрађују огромне количине података, основна идеја остаје иста – подаци се редом пореде, а добијени резултати ажурирају када се пронађе већа или мања вредност.

2.21

ПРЕБРОЈАВАЊЕ ЕЛЕМЕНАТА У НИЗУ

ВЕЋ ЗНАШ ДА...

- уносиш елементе у једнодимензионални низ;
- обрађујеш елементе циклусом for
- израчунаваш суму и производ;
- одређујеш највећи и најмањи елемент;
- користиш условне конструкције при обради података.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Зашто се при обради низа најчешће користи циклус for?
2. Како се опредељује највећи елемент у низу?
3. Како се одређује најмањи елемент?
4. Да ли је могуће да више вредности буду обрађене једним проласком кроз низ?
5. Шта представља распон података?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- пребројаваш елементе који испуњавају задати услов;
- користиш променљиву за бројање;
- комбинујеш циклусе и условне конструкције;
- примењујеш логичке изразе при обради низа;
- анализираш податке према различитим критеријумима.

При обради података често није потребно одредити само њихов збир или највећу вредност. У многим практичним ситуацијама потребно је утврдити колико елемената има одређено својство. На пример, може бити потребно одредити број дана са падавинама, број возила која су прекорачила дозвољену брзину или број успешно обављених мерења.

За решавање оваквих задатака користи се променљива за бројање. На почетку она добија вредност 0. При обради сваког елемента проверава се да ли је испуњен задати услов. Ако је испуњен услов, вредност

променљиве се повећава за 1. После обраде свих елемената, она садржи укупан број елемената који испуњавају услов.

При пребројавању елемената низа најпре се креира променљива за бројање којој се додељује почетна вредно Затим се сви елементи редом обрађују помоћу циклуса. За сваки елемент проверава се да ли испуњава задати услов. Ако је услов испуњен, вредност променљиве за бројање повећава се за 1. Након обраде последњег елемента, променљива садржи коначан резултат. Овакав начин рада има широку примену у статистичкој обради и анализи различитих врста података.

Пример 1: Високе биљке

У ботаничкој башти уносе се висине седам биљака. Одредити колико је биљака више од 100 cm.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int visina[7];
    int broj =0;
    for(int i=0;i<7;i++)
    {
        cout<<"Visina "<<i+1<<": ";
        cin>>visina[i];
        if(visina[i] > 100)
        {
            broj++;
        }
    }
    cout << "Broj biljaka     : "<<broj<< endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Visina 1: 105
Visina 2: 92
Visina 3: 56
Visina 4: 213
Visina 5: 84
Visina 6: 155
Visina 7: 142
Broj na Broj biljaka: 4

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

У примеру се проверава свака унета вредност. Ако је биљка виша од 100 cm, променљива за бројање повећава се за 1. У практичним задацима услов не мора бити повезан са величином вредности. Често је потребно пребројати елементе који имају одређено математичко својство. Једно од најчешћих својстава јесте парност бројева. У ту сврху користи се оператор %, који омогућава одређивање остатка при дељењу.

Пример 2: Парни резултати

У спортски центар се уноси шест резултата. Одредити колико има парних бројева.

```cpp
#include <iostream>

using namespace std;

int main()
{
    int rezultat[6];
    int broj = 0;
    for(int i = 0; i < 6; i++)
    {
        cout<<"Rezultat "<<i+1<<": ";
        cin>> rezultat[i];
        if(rezultat[i]%2==0)
        {
            broj++;
        }
    }
    cout<<"Parni rezultati: "<<broj<<endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Rezultat 1: 2
Rezultat 2: 3
Rezultat 3: 4
Rezultat 4: 5
Rezultat 5: 6
Rezultat 6: 4
Parni rezultati: 4

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

У примеру се за сваку унету вредност проверава да ли је паран број. Ако је услов испуњен, број парних вредности повећава се за 1.

При обради података често је потребно одредити број елемената који се налазе у одређеном интервалу. На пример, може бити потребно одредити број температура између две вредности, број ученика са оценама у одређеном опсегу или број производа који испуњавају постављене услове.

температура између две вредности, број ученика са оценама у одређеном опсегу или број производа који испуњавају постављене услове. У таквим ситуацијама за сваки елемент проверава се више услова истовремено. Ако су сви услови испуњени, променљива за бројање повећава се за 1. На тај начин лако се може одредити број елемената који припадају задатом интервалу.

Пример 3:Вредности у задатом интервалу

У истраживачкој станици уноси се седам измерених вредности. Одредити колико се њих налази у интервалу од 20 до 50.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    int vrednost[7];
    int broj = 0;
    for(int i=0;i<7;i++)
    {
        cout<<"Vrednost "<<i+1<<": ";
        cin >> vrednost[i];
        if(vrednost[i]>=20&&vrednost[i]<=50)
        {
            broj++;
        }
    }
    cout<<"Broj elemenata  : "<<broj<<endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Vrednost 1: 18
Vrednost 2: 22
Vrednost 3: 87
Vrednost 4: 54
Vrednost 5: 35
Vrednost 6: 40
Vrednost 7: 11
Broj elemenata i: 3

Process returned 0 (0x0)   execution time : 16.578 s
Press any key to continue.
```

У примеру се за сваку унету вредност проверава да ли припада задатом интервалу. Ако је услов испуњен, променљива за бројање повећава се за 1. Након обраде свих елемената добија се коначан резултат.

При пребројавању елемената услови који треба да буду испуњени могу бити веома различити. У неким задацима проверава се да ли је вредност већа од задатог броја, у другим да ли је парна, а у трећим да ли припада одређеном интервалу. Без обзира на постављени проблем, основни поступак остаје исти. Елементи се редом обрађују, услов се проверава за сваки елемент,

а променљива за бројање повећава се када је услов испуњен. Захваљујући овој техници може се решити велики број практичних програмских задатака.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Чему служи променљива за бројање?
2. Којом почетном вредношћу најчешће започиње променљива за бројање?
3. Када се повећава њена вредност?
4. Како се проверава да ли је број паран?
5. Како се може проверити да ли елемент припада одређеном интервалу?

ЗАДАЦИ

1. У националном парку уносе се висине осам водопада. Напиши програм који ће одредити колико је водопада више од 50 метара.
2. У роботској лабораторији уносе се бројеви успешно завршених задатака шест робота. Напиши програм који ће одредити колико је резултата непарних бројева.
3. У центру за обновљиву енергију уноси се дневна производња седам соларних система. Напиши програм који ће одредити колико се вредности налази између 100 и 300.
4. У океанографској експедицији уносе се дубине шест мерења. Напиши програм који ће одредити колико је мерења веће од 500 метара.
5. У астрономској опсерваторији уносе се бројеви уочених метеора током пет ноћи. Напиши програм који ће одредити у колико је ноћи регистровано више од 20 метеора.

ЗАПАМТИ ШТА СИ НАУЧИО

- За пребројавање елемената користи се променљива за бројање.
- Променљива за бројање најчешће започиње вредношћу 0.
- Њена вредност се повећава када је испуњен задати услов.
- Услов може бити једноставан или састављен од више провера.
- Једним проласком кроз низ може се одредити број елемената који имају одређено својство.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Пребројавање елемената представља једну од најчешће примењиваних техника при анализи података. У медицини се може одредити број пацијената који имају одређени симптом, у екологији број дана са падавинама, а у спорту број успешно реализованих покушаја. Иако савремени информациони системи обрађују огромне количине података, основна идеја остаје непромењена – сваки податак се проверава, а ако испуњава задати услов, резултат пребројавања се ажурира.

2.22

НИЗОВИ КАРАКТЕРА И ПРЕТРАЖИВАЊЕ ЗНАКОВА

ВЕЋ ЗНАШ ДА...

- декларишеш и користиш једнодимензионалне низове целих бројева;
- уносиш и обрађујеш елементе у низу;
- користиш циклус for;
- примењујеш условне конструкције;
- пребројаваш елементе који испуњавају одређени услов.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Шта претставља једнодимензионални низ?
2. Од ког броја почиње нумерисање елемената?
3. Зашто се при раду са низовима користи циклус for?
4. Чему служи променљива за бројање?
5. Како може да буде проверено да ли елемент испуњава одређени услов?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- користиш низове карактера;
- уносиш знаке у низу;
- претражујеш знак у низу;
- одређујеш да ли има знак;
- пребројаваш појављивања знака.

Осим бројева, у програмима је често потребно обрађивати и знакове. На пример, може бити потребно проверити да ли се одређено слово налази у коду, да ли се одређени симбол појављује у тексту или колико пута је употребљен одређени знак.

За чување појединачних знакова у програмском језику C++ користи се тип char. Више знакова може се сачувати у низу знакова. На тај начин сваки знак има своју позицију у низу и може се појединачно обрађивати.

Низ знакова декларише се на исти начин као и низови целих бројева. Разлика је у томе што се уместо типа int користи тип char. Сваки елемент низа садржи по један знак. Елементи могу бити слова, цифре или други симболи. При обради низа сваки знак може се упоредити са одређеном вредношћу и на основу резултата донети одлука. Захваљујући томе могу се решавати различити практични задаци повезани са обрадом текстуалних података.

Пример 1: Провери да ли има знак

У контролном систему уноси се шест ознака. Проверити да ли се међу њима налази слово A.

```cpp
#include <iostream>

using namespace std;

int main()
{
    char oznaka[6];
    bool pronadjen=false;
    for(int i=0;i<6;i++)
    {
        cout<<"Oznaka "<<i+1<<": ";
        cin>>oznaka[i];
    }
    for(int i=0;i<6;i++)
    {
        if(oznaka[i]=='A')
        {
            pronadjen=true;
        }
    }
    if(pronadjen)
    {
        cout<<"Znak postoji .";
    }
    else
    {
        cout<<"Znak ne postoji .";
    }
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Oznaka 1: A
Oznaka 2: B
Oznaka 3: A
Oznaka 4: 3
Oznaka 5: c
Oznaka 6: a
Znak postoji .
Process returned 0 (0x0)   execution time : 16.964 s
Press any key to continue.
```

У примеру се сви знакови најпре уносе у низ. Затим се редом проверава да ли је неки од њих једнак слову А. Ако се слово пронађе, резултат се чува.

При обради знакова често није довољно утврдити да ли одређени знак постоји. У многим ситуацијама потребно је одредити колико се пута он појављује. У ту сврху користи се променљива за бројање која се повећава сваки пут када се пронађе тражени знак.

Пример 2: Пребројавање појављивања

У комуникационом систему уноси се осам ознака. Одредити колико се пута појављује слово К.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    char znak[8];
    int broj = 0;

    for(int i=0;i<8;i++)
    {
        cout<<"Znak "<<i+1<<": ";
        cin>>znak[i];
    }
    for(int i=0;i<8;i++)
    {
        if(znak[i]=='K')
        {
            broj++;
        }
    }
    cout<<"Broj pojavljivanja : "<<broj<<endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Znak 1: K
Znak 2: A
Znak 3: k
Znak 4: R
Znak 5: x
Znak 6: X
Znak 7: K
Znak 8: k
Broj pojavljivanja a: 2

Process returned 0 (0x0)   execution time : 24.749 s
Press any key to continue.
```

У примеру се примењује техника пребројавања. Сваки пут када се пронађе слово К, променљива за бројање повећава се за 1.

При раду са низовима знакова није увек потребно тражити само одређено слово. У практичним програмским задацима често се проверава да ли се појављује одређени симбол. То могу бити знакови #, @, * или друге ознаке које имају посебно значење у одређеним системима. Поступак обраде остаје непромењен. Сваки елемент низа редом се пореди са траженим знаком, а ако се он пронађе, доноси се одговарајући закључак.

Пример 3: Провера специјалног знака

У систем за приступ унето је седам ознака. Да се провери да ли је међу њима знак #.

main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    char oznaka[7];
    bool pronadjen=false;
    for(int i=0;i<7;i++)
    {
        cout<<"Oznaka "<<i+1<<": ";
        cin>>oznaka[i];
    }
    for(int i=0;i<7;i++)
    {
        if(oznaka[i]=='#')
        {
            pronadjen=true;
        }
    }
    if(pronadjen)
    {
        cout<<"Znak # je pronađen."
<endl;
    }
    else
    {
        cout<<"Znak # nije pronađen."
<endl;
    }
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Oznaka 1: a
Oznaka 2: s
Oznaka 3: d
Oznaka 4: s
Oznaka 5: F
Oznaka 6: W
Oznaka 7: Q
Znak # nije pronađen.

Process returned 0 (0x0)   execution time : 9.477 s
Press any key to continue.
```

У овом примеру сви унети знаци узастопно се упоређују са знаком #. Ако буде пронађен, програм обавештава корисника да је знак присутан у низу.

При обради низова знакова могу се поставити различити услови за претраживање. У неким задацима тражи се присуство одређеног слова, у другим се пребројава број појављивања одређеног знака, а у трећим се проверава да ли се појављује одређени симбол. Независно од врсте услова, основни поступак остаје исти. Сваки елемент низа се обрађује и упоређује, а на основу резултата доноси се закључак. Ова техника представља основу за обраду текстуалних података у програмским језицима.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Шта представља низ карактера?
2. Који тип података се користи за чување једног знака?
3. Како се може проверити да ли се одређени знак налази у низу?
4. Како се може одредити колико пута се појављује одређени знак?
5. Да ли у низу карактера могу да се чувају слова, цифре и специјални симболи?

ЗАДАЦИ

1. У контролном систему уноси се осам ознака. Напиши програм који ће проверити да ли се међу њима налази слово M.
2. У комуникационом центру уноси се шест знакова. Напиши програм који ће одредити колико пута се појављује цифра „5“.
3. У информатичкој лабораторији уноси се десет знакова. Напиши програм који ће проверити да ли је унет знак @.
4. У систему за евиденцију уноси се седам ознака. Напиши програм који ће одредити колико пута се појављује слово S.
5. У истраживачкој станици уноси се девет симбола. Напиши програм који ће проверити да ли се појављује знак *.

ЗАПАМТИ ШТА СИ НАУЧИО

- Низ карактера представља скуп појединачних знакова.
- За чување једног знака користи се тип char.
- Знакови могу бити слова, цифре или специјални симболи.
- Претраживање знака врши се поређењем елемената низа.
- Помоћу променљиве за бројање може се одредити број појављивања одређеног знака.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим информационим системима обрада текстуалних података представља једну од најважнијих програмских активности. При уносу корисничких имена, лозинки, ознака производа или различитих кодова, програми често проверавају да ли су коришћени одређени знакови или симболи. Иако се у овој лекцији обрађују појединачни знакови, исти принципи касније се примењују и при раду са речима, реченицама и већим текстуалним подацима.

2.23

ПРИМЕНА МАТЕМАТИЧКИХ ФУНКЦИЈА НА ЕЛЕМЕНТЕ НИЗА

ВЕЋ ЗНАШ ДА...

- уносиш елементе у једнодимензионални низ;
- обрађујеш елементе циклусом for;
- израчунаваш суму, производ и просек;
- одређујеш највећи и најмањи елемент;
- користиш функције математичке библиотеке.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

1. Чему служи библиотека cmath?
2. Која функција израчунава квадратни корен?
3. Која функција се користи за степеновање?
4. Чему служи функција abs()?
5. Да ли функције из математичке библиотеке могу да се примењују на елементе низа?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- примењујеш математичке функције на елементе низа;
- комбинујеш низове и библиотеку cmath;
- обрађујеш унете податке са математичким прорачунима;
- упоређујеш добијене резултате;
- решаваш практичне задатке са комбинованом применом изучених техника.

При обради података често је потребно да се над сваким елементом низа изврши одређена математичка операција. У практичним програмским решењима може бити потребно израчунати квадратни корен, квадрат броја или апсолутну вредност унетих података. У ту сврху се у програмском језику C++ користи математичка библиотека cmath.

Функције из ове библиотеке могу се применити на сваки елемент низа. Помоћу циклуса се редом обрађују све вредности, а над сваком од њих извршава се потребна математичка операција. На овај начин могу се решити различити практични проблеми из науке, технике и свакодневног живота.

При примени математичких функција на елементе низа, најпре се уносе подаци, а затим се сваки елемент редом обрађује. За сваки елемент може се израчунати квадратни корен, квадрат, апсолутна вредност или нека друга математичка функција. Добијени резултати могу се одмах приказати или додатно обрађени. Захваљујући овом поступку, велики број података може се брзо анализирати и обрадити помоћу релативно малог броја програмских инструкција.

Пример 1: Квадратни корен измерених растојања

У научној лабораторији уноси се пет измерених растојања. Израчунати квадратни корен сваког растојања.

main.cpp

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    double Rastojanje[5];
    for(int i=0;i<5;i++)
    {
        cout<<"Rastojanje "<<i+1<<": ";
        cin>>Rastojanje[i];
    }
    cout<<endl;
    for(int i=0;i<5;i++)
    {
        cout<<"Kvadratni koren : "<<sqrt(Rastojanje[i])<<endl;
    }
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Rastojanje 1: 5
Rastojanje 2: 4
Rastojanje 3: 3
Rastojanje 4: 81
Rastojanje 5: 6

Kvadratni koren: 2.23607
Kvadratni koren: 2
Kvadratni koren: 1.73205
Kvadratni koren: 9
Kvadratni koren: 2.44949
```

У примеру се сваки унети податак обрађује независно. Помоћу функције sqrt() израчунава се квадратни корен сваке вредности, а резултат се одмах приказује.

При обради података често је потребно применити исту математичку операцију на све елементе низа. На пример, може бити потребно израчунати квадрате свих измерених вредности или одредити апсолутну вредност сваког мерења. У таквим ситуацијама сваки елемент се редом обрађује и на њега се примењује одговарајућа математичка функција. Захваљујући овом приступу, велики број података може се брзо обрадити једноставним програмским поступком.

Пример 2: Квадрати измерених вредности

истраживачком центру уносе се пет измерених вредности. Израчунати квадрат сваке вредности.

main.cpp

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    double vrednost[5];
    for(int i=0;i<5;i++)
    {
        cout<<"Vrednost "<<i+1<<": ";
        cin>>vrednost[i];
    }
    cout << endl;
    for(int i=0;i<5;i++)
    {
        cout<<"Kvadrat: "<<pow(vrednost[i],2)<<endl;
    }
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Vrednost 1: 4
Vrednost 2: 2
Vrednost 3: 3
Vrednost 4: 4
Vrednost 5: 6

Kvadrat: 16
Kvadrat: 4
Kvadrat: 9
Kvadrat: 16
Kvadrat: 36
```

У примеру се сваки елемент низа обрађује појединачно. Помоћу функције pow() израчунава се квадрат сваке унете вредности.

При практичној обради података често се појављују и негативне вредности. У одређеним ситуацијама није важан њихов знак, већ само њихова величина. У ту сврху користи се функција abs(), којом се одређује апсолутна вредност броја. Након обраде свих елемената може се извршити и додатна анализа добијених резултата.

Пример 3: Највећа апсолутна вредност

У научној лабораторији уноси се пет мерења која могу бити позитивна или негативна. Одредити највећу апсолутну вредност.

main.cpp

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    int merenje[5];
    int najveci;
    for(int i=0;i<5;i++)
    {
        cout<<"Merenje "<<i+1<< ": ";
        cin>>merenje[i];
    }
    najveci =abs(merenje[0]);
    for(int i=1;i<5;i++)
    {
        if(abs(merenje[i])>najveci)
        {
            najveci =abs(merenje[i]);
        }
    }
    cout<<"Najveća apsolutna vrednost : "<<najveci <<endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Merenje 1: 45
Merenje 2: -23
Merenje 3: 48
Merenje 4: 65
Merenje 5: -87
Najveća apsolutna vrednost t: 87

Process returned 0 (0x0)   execution time : 19.213 s
Press any key to continue.
```

Logs & others

У примеру се на сваки елемент најпре примењује функција abs(). Затим се добијене вредности упоређују и одређује се највећа апсолутна вредност.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Зашто у програм треба укључити библиотеку cmath?
2. Како се математичка функција може применити на све елементе једног низа?
3. Зашто је корисно комбиновати математичке функције са низовима и циклусима?

ЗАДАЦИ

1. У геодетској служби уноси се пет растојања. Напиши програм који ће израчунати квадратни корен сваког растојања.
2. У физичкој лабораторији уноси се шест мерења. Напиши програм који ће израчунати квадрат сваке вредности.
3. У истраживачкој станици уноси се пет позитивних и негативних мерења. Напиши програм који ће одредити највећу апсолутну вредност.
4. У астрономској опсерваторији уноси се седам растојања. Напиши програм који ће израчунати куб сваког растојања.
5. У инжењерској лабораторији уноси се шест мерења. Напиши програм који ће израчунати апсолутну вредност сваког мерења и приказати добијене резултате.

ЗАПАМТИ ШТА СИ НАУЧИО

- Функције из библиотеке cmath могу се применити на елементе једног низа.
- Комбиновањем низова, циклуса и математичких функција могу се решавати различити практични задаци.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим информационим системима велики број података обрађује се применом математичких функција. У науци се могу анализирати резултати мерења, у инжењерству различите физичке величине, а у финансијским системима различити статистички показатељи. Иако савремени програми обрађују огромне количине информација, основна идеја остаје иста – сваки податак обрађује се одговарајућом математичком функцијом, а добијени резултати се даље могу анализирати и упоређивати.

2.24

КОМБИНОВАНА ПРИМЕНА ФУНКЦИЈА И ЈЕДНОДИМЕНЗИОНАЛНИХ НИЗОВА

ВЕЋ ЗНАШ ДА...

- користиш једнодимензионалне низове;
- уносиш и обрађујеш елементе;
- дефинишеш и позиваш функције;
- користиш циклус for;
- комбинујеш различите програмске конструкције.

ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представља аргумент функције?
2. Шта представља позив функције?
3. Да ли нека функција може да буде позвана више пута у једном програму?
4. Зашто при обради низова најчешће се користи циклус?
5. Какву предност доноси комбиновање функција и низова?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- Примењујеш функције на елементе низа;
- Комбинујеш циклусе и функције;
- обрађујеш податке корисничким функцијама;
- користиш повратне вредности при раду са низовима;
- решаваш практичне програмске задатке.

При обради података често је потребно извршити исти поступак над више елемената низа. Уместо да се исте програмске инструкције понављају више пута, оне се могу сместити у функцију. Затим се функција редом позива за сваки елемент низа.

Комбиновањем функција и једнодимензионалних низова програми постају прегледнији и лакши за разумевање. Иста функција може се употребити за обраду свих елемената, без потребе за поновним писањем истих програмских инструкција.

При комбиновању функција и низова функција најчешће обрађује један елемент, док циклус омогућава да она буде позвана за све елементе низа. На овај начин свака вредност се обрађује на исти начин, а програм остаје једноставан и прегледан. Овакав приступ има широку примену при решавању различитих програмских задатака и представља природан наставак изучавања функција и једнодимензионалних низова.

Пример 1: Квадрат елемената

У истраживачкој лабораторији уноси се пет вредности. Приказати квадрат сваке вредности помоћу функције.

```cpp
#include <iostream>

using namespace std;

int kvadrat(int x)
{
    return x * x;
}

int main()
{
    int broj[5];
    for(int i=0;i<5;i++)
    {
        cout<<"Broj "<<i+1<<": ";cin>>broj[i];
    }
    cout << endl;
    for(int i = 0; i < 5; i++)
    {
        cout <<kvadrat(broj[i])<<endl;
    }
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Broj 1: 2
Broj 2: 4
Broj 3: 3
Broj 4: 8
Broj 5: 15

4
16
9
64
225

Process returned 0 (0x0)   execution time : 7.610 s
Press any key to continue.
```

У примеру функција обрађује један елемент, док циклус омогућава да се иста функција позове за све елементе низа.

При комбиновању функција и низова није неопходно да функција израчунава само математичку вредност. У многим ситуацијама потребно је проверити да ли одређени елемент испуњава задати услов. У таквим случајевима функција може да враћа логичку вредност. Ако је услов испуњен, функција враћа true, а у супротном false. Помоћу циклуса иста функција може се позвати за све елементе низа, а добијени резултати могу се додатно обрађивати.

Пример 2: Провера парних бројева

У примеру функција проверава да ли је број паран. Затим се иста функција позива за сваки елемент низа и пребројавају се сви бројеви који испуњавају услов.

main.cpp

```
#include <iostream>

using namespace std;

bool paren(int x)
{
    if(x % 2 == 0)
    {
        return true;
    }
    return false;
}

int main()
{
    int broj[6];
    int ukupno = 0;
    for(int i=0;i<6;i++)
    {
        cout<<"Broj "<<i+1<<": ";
        cin >> broj[i];
    }
    for(int i = 0; i < 6; i++)
    {
        if(jParni brojevi ))
        {
            ukupno>++;
        }
    }
    cout<<"Parni brojevi : "<<ukupno<<endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug
Broj 1: 2
Broj 2: 5
Broj 3: 45
Broj 4: 62
Broj 5: 188
Broj 6: 145
Parni brojevi i: 3

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

У примеру функција проверава да ли је број паран. Затим се иста функција позива за сваки елемент низа и пребројавају се сви бројеви који испуњавају услов.

При раду са функцијама често је потребно израчунати одређену вредност за сваки елемент низа, а затим додатно обрадити добијене резултате. На овај начин могу се комбиновати различите програмске технике које су претходно изучаване одвојено. Захваљујући оваквом приступу, програми постају организованији и лакши за одржавање.

Пример 3: Запремина коцки и њихова сума

У техничкој лабораторији потребно је оптимизовати простор око пет објеката који ће бити послати у свемир. Свих пет објеката има облик коцке. Потребно је извршити пет мерења запремине тела облика коцке. Израчунати збир свих измерених запремина помоћу функције.

main.cpp

```
#include <iostream>

using namespace std;

int zapremina (int x)
{
    return x * x * x;
}
int main()
{
    int broj[5];
    int zbir=0;
    for(int i=0;i<5;i++)
    {
        cout<<"Kocka "<<i+1<<": ";
        cin>>broj[i];
    }
    for(int i=0;i<5;i++)
    {
        zbir += zapremina(broj[i]);
    }
    cout << "Zbir zapremina: " << zbir << endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Kocka 1: 2
Kocka 2: 3
Kocka 3: 11
Kocka 4: 5
Kocka 5: 9
Zbir zapremina      i: 2220

Process returned 0 (0x0)   execution time : 10.651 s
Press any key to continue.
```

Logs & others

main.cpp

```cpp
#include <iostream>

using namespace std;

int kub(int x)
{
    return x * x * x;
}
int main()
{
    int broj[5];
    int zbir=0;
    for(int i=0;i<5;i++)
    {
        cout<<"Merenje "<<i+1<<": ";
        cin>>broj[i];
    }
    for(int i=0;i<5;i++)
    {
        zbir +=kub(broj[i]);
    }
    cout<<"Zbir kubova     : "<<zbir<<endl;
    return 0;
}
```

```
D:\TD\Cpp\Vtora\bin\Debug\Vtora.exe
Merenje 1: 2
Merenje 2: 5
Merenje 3: 14
Merenje 4: 22
Merenje 5: 4
Zbir kubova  i: 13589

Process returned 0 (0x0)   execution time : 9.686 s
Press any key to continue.
```

У техничкој лабораторији потребно је оптимизовати простор око пет објеката који ће бити послати у свемир. Свих пет објеката има облик коцке. Потребно је извршити пет мерења запремине тела облика коцке. Израчунати збир свих измерених запремина помоћу функције.

ПИТАЊА И ЗАДАЦИ ЗА ПОНАВЉАЊЕ

ПИТАЊА

1. Какав задатак може да извршава функција која обрађује један елемент низа?
2. Како циклус омогућава да се иста функција примени на више елемената?
3. У којим ситуацијама функција може да враћа логичку вредност?
4. Зашто је корисно да обрада података буде смештена у функцију?
5. Које су предности комбиновања функција и једнодимензионалних низова?

ЗАДАЦИ

1. У метеоролошкој станици уноси се шест температура. Напиши програм у којем ћеш употребити функцију која израчунава троструку вредност температуре (помножену са 3) и приказати резултате.
2. У центру за роботику уноси се пет резултата. Напиши програм у којем ћеш употребити функцију која проверава да ли је број позитиван и одредити колико је позитивних вредности унето.
3. У научној лабораторији уноси се шест мерења. Напиши програм у којем ћеш употребити функцију која израчунава остатак при дељењу бројем 5 и приказати добијене вредности.
4. У спортском центру уноси се пет резултата. Напиши програм у којем ћеш употребити функцију која израчунава половину сваког резултата и приказати добијене вредности.
5. У енергетском центру уноси се шест мерења. Напиши програм у којем ћеш употребити функцију која одређује да ли је вредност већа од 100 и израчунати број таквих мерења. се внесуваат шест мерења. Напиши програма во која ќе употребиш функција што определува дали вредноста е поголема од 100 и ќе го пресмета бројот на такви мерења.

ЗАПАМТИ ШТА СИ НАУЧИО

- Функција може да обрађује појединачни елемент низа.
- Иста функција може да се позове за све елементе низа.
- Функција може да враћа бројчану или логичку вредност.
- Циклус омогућава да се функција примени на све елементе.
- Комбиновање функција и низова омогућава да програми буду прегледнији и лакши за одржавање.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременом програмирању функције представљају један од најважнијих начина за организовање програмског кода. Уместо да се исте инструкције понављају на више места, оне се смештају у функције које се могу позивати више пута. Када се овакав приступ комбинује са једнодимензионалним низовима, велике количине података могу се једноставно обрађивати. Овакав начин рада примењује се при обра-

2.25

ди научних мерења, финансијских анализа, статистичких података и у многим другим савременим информационим системима.

ЗАДАЦИ ЗА ВЕЖБАЊЕ СА НИЗОВИМА

Лаки задаци

1. У планинарском дому током једне седмице бележи се број посетилаца. Напиши програм који ће израчунати укупну посећеност.
2. У расаднику се евидентира висина десет младих садница. Напиши програм који ће одредити просечну висину.
3. У музеју се евидентира број посетилаца током осам узастопних дана. Напиши програм који ће одредити дан са највећом посећеношћу.
4. У ветроелектрани се евидентира дневна производња електричне енергије девет генератора. Напиши програм који ће одредити који генератор има најмању производњу.
5. У парку дивљих животиња бележи се број рођених младунаца у седам различитих станишта. Напиши програм који ће одредити у колико је станишта рођено више од три младунца.
6. У библиотеку пристижу књиге у различитом броју примерака. У десет испорука евидентира се број књига. Напиши програм који ће проверити да ли нека испорука садржи тачно 100 књига.
7. У фабрици сокова бележи се број произведених флаша током осам сати. Напиши програм који ће одредити разлику између најуспешнијег и најслабијег сата.
8. У националном парку бележи се број уочених јелена током девет посматрања. Напиши програм који ће одредити у колико је посматрања уочено најмање пет јелена.
9. У астрономском кампу бележи се број уочених метеора током десет ноћи. Напиши програм који ће израчунати укупан број метеора уочених током парних ноћи.
10. У акваријуму се прати број новорођених риба у осам базена. Напиши програм који ће одредити колико базена има број новорођених риба већи од просечног броја.

Средње тешки задаци

1. У руднику се евидентирају површине осам квадратних ископа. Напиши програм који ће одредити дужину странице сваког ископа.
2. У расаднику се бележе дужине страница седам квадратних цветних леја. Напиши програм који ће израчунати површину сваке леје.

3. У геолошкој лабораторији евидентирају се позитивна и негативна померања земљишта током девет мерења. Напиши програм који ће одредити које је померање највеће, без обзира на смер.
4. У центру за спортске припреме прати се растојање које је претрчало шест спортиста. Напиши програм који ће израчунати квадрат растојања сваког спортисте и приказати добијене вредности.
5. У архитектонском студију пројектују се кружне фонтане. Познати су радијуси осам фонтана. Напиши програм који ће израчунати површину сваке фонтане.
6. У грађевинској компанији уносе се дужине страница седам коцки које ће се користити као декоративни елементи. Напиши програм који ће израчунати запремину сваке коцке.
7. У астрономској опсерваторији евидентирају се позитивна и негативна одступања телескопа. Напиши програм који ће израчунати укупну величину свих одступања, без обзира на њихов смер.
8. У научној лабораторији уносе се површине квадратних узорака. Напиши програм који ће одредити који узорак има најдужу страницу.
9. У центру за обновљиву енергију познати су радијуси шест кружних соларних платформи. Напиши програм који ће израчунати укупну површину свих платформи.
10. У геодетској служби евидентирају се позитивне и негативне грешке при мерењу. Напиши програм који ће израчунати просечну величину грешака, без обзира на то да ли су позитивне или негативне.

Тешки задаци

1. У пчеларском центру свака кошница добија оцену за продуктивност. Кошница се сматра успешном ако произведе најмање 25 kg меда. Током сезоне евидентира се производња осам кошница. Напиши програм који ће одредити колико успешних кошница има.
2. У планинарском клубу бележе се висине десет врхова. Врх се сматра високим ако је његова надморска висина најмање 2000 метара. Напиши програм који ће одредити колико је високих врхова евидентирано.
3. У акваријуму се прати број новорођених риба у девет базена. Базен се сматра успешним ако је у њему рођено више од 50 риба. Напиши програм који ће одредити број успешних базена.
4. У астрономској опсерваторији бележе се растојања седам небеских тела. Небеско тело се сматра блиским ако је растојање мање од унапред задате вредности. Напиши програм који ће одредити број блиских тела.
5. У националном парку бележи се старост десет стабала. Стабло се сматра вековним ако је старо најмање 100 година. Напиши програм који ће одредити колико вековних стабала има.
6. У центру за спасавање животиња евидентира се телесна маса осам животиња. За сваку животињу треба одредити да ли припада групи лаких или тешких животиња према унапред задатој граници. Напиши програм који ће одредити број тешких животиња.
7. У спортском кампу бележе се дужине скокова девет такмичара. Скок се сматра успешним ако је дужи од задате границе. Напиши програм који ће одредити колико је успешних скокова реализовано.
8. У ветропарку бележи се производња седам турбина. Турбина се сматра ефикасном ако произведе више од задате количине енергије. Напиши програм који ће одредити број ефикасних турбина.
9. У археолошкој експедицији евидентира се старост осам пронађених предмета. Предмет се сматра историјски значајним ако је његова старост већа од унапред одређене вредности. Напиши програм који ће одредити број значајних предмета.
10. У центру за обновљиву енергију прати се дневна производња девет соларних система. Систем се сматра успешним ако произведе више од задате количине енергије. Напиши програм који ће одредити колико успешних система има.

11. У мото-клубу бележе се пређени километри осам возача. Возач добија авантуристички статус ако је прешао најмање 300 km. Напиши програм у којем ће обрада за једног возача бити издвојена у функцију, а програм ће одредити колико је возача добило авантуристички статус.
12. У пчелињаку се прати количина меда из седам кошница. Кошница се сматра продуктивном ако произведе најмање 25 kg меда. Напиши програм у којем ће провера за једну кошницу бити реализована помоћу функције, а подаци ће бити обрађени преко низа.
13. У свемирској мисији бележе се величине девет астероида. Астероид се сматра ризичним ако је његова величина већа од задате границе. Напиши програм који ће обрадити све астероиде у низу и помоћу функције утврђивати да ли је неки астероид ризичан.
14. У археолошкој експедицији уноси се старост осам предмета. Предмет се сматра значајним ако је старији од 1000 година. Напиши програм у којем ће функција одређивати значајност једног предмета, а програм ће приказати број значајних предмета.
15. У фабрици се бележи број израђених делова шест робота. Робот се сматра ефикасним ако је израдио више од 120 делова. Напиши програм у којем ће функција проверавати ефикасност једног робота, а низ ће чувати резултате свих робота.
16. У центру за спасавање уноси се телесна маса седам животиња. Животиња се сматра тешком ако је њена маса већа од унапред задате границе. Напиши програм у којем ће функција вршити проверу за једну животињу, а програм ће израчунати колико је животиња тешко.
17. У паметном граду прати се дневна потрошња електричне енергије осам објеката. Објекат се сматра великим потрошачем ако је потрошња већа од 500 јединица. Напиши програм у којем ће функција одређивати да ли је један објекат велики потрошач.
18. У спортском кампу уносе се резултати девет скокова удаљ. Скок се сматра успешним ако је дужи од задате границе. Напиши програм у којем ће функција проверавати да ли је један скок успешан, а програм ће приказати број успешних скокова.
19. Еколошка патрола бележи количину прикупљеног отпада на осам локација. Локација се сматра критичном ако је количина отпада већа од задате границе. Напиши програм у којем ће функција проверавати стање једне локације, а програм ће одредити колико критичних локација има.
20. У подводној мисији уносе се дубине седам роњења. Роњење се сматра дубоким ако је дубина већа од 40 метара. Напиши програм у којем ће функција проверавати да ли је неко роњење дубоко, а низ ће чувати сва мерења.

О теми

У савременом друштву свакодневно се стварају и обрађују велике количине података. Подаци се користе у школама, банкама, болницама, библиотекама, продавницама, државним институцијама и у многим другим областима. За њихово ефикасно чување, организовање и обраду примењују се базе података, које омогућавају брзо проналажење, ажурирање и управљање информацијама. Захваљујући њиховој примени, рад различитих организација постаје ефикаснији, а приступ потребним информацијама бржи и поузданији.

У овој тематској целини обрађују се основни појмови повезани са базама података и системима за управљање базама података. Посебна пажња посвећује се организовању података у табеле, записима и пољима, успостављању веза између података, као и претраживању, уношењу и ажурирању информација. Кроз практичне активности стичу се основна знања и вештине за коришћење база података у различитим областима свакодневног живота и савременог пословања.

НОВИ ПОЈМОВИ

- база података;
- систем за управљање базом података;
- примарни кључ;
- веза;
- претраживање;
- ажурирање;
- сортирање;
- филтрирање;
- упит (query);
- извештај.
-

ВЕЋ ЗНАШ ДА...

- унесеш и уређујеш податке;
- организујеш информације у табели;
- користиш различите компјутерске програме;
- претражујеш информације;
- анализираш и упоређујеш податке;
- примењујеш дигиталне технологије при решавању практичних задатака;

TEMA 3

3.1

ОСНОВИ БАЗЕ ПОДАТАКА

ВЕЋ ЗНАШ ДА...

- користиш разне компјутерске програме;
- уносиш и обрађујеш податке;
- организујеш информације у табеларном облику;
- претражујеш информације на интернету;
- користиш разне дигиталне услуге;
- анализираш и упоређујеш податке.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Како банке чувају податке својих корисника?
2. Како школа води евиденцију о ученицима и наставницима?
3. Како интернет-продавнице знају који производи су доступни?
4. Како хиљаде корисника може истовремено да користи исте информације?
5. Зашто је важно да се подаци лако налазе и ажурирају?

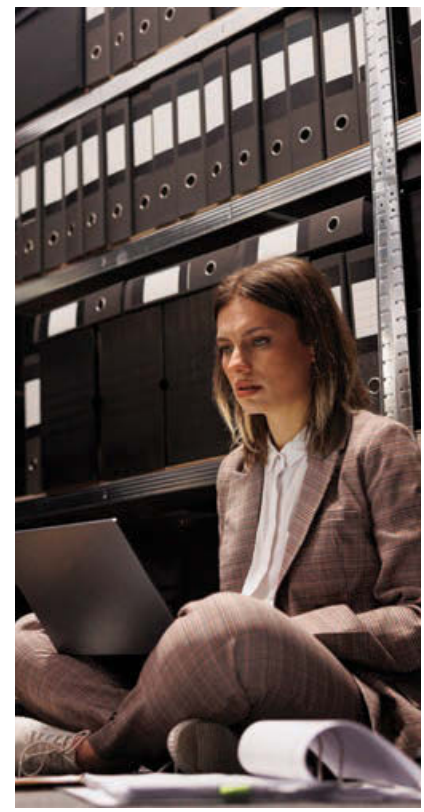
У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш шта представља базу података;
- наводиш примере примене базе података;
- објашњаваш зашто се користе базе података;
- анализираш предности њихове примене;
- препознајеш њихово значење у савременим информатичким системима.

Савремено друштво свакодневно ствара огромне количине информација. Свака куповина у продавници, свако подизање новца на банкомату, свако претраживање интернета и свака порука послата преко друштвених мрежа представљају податке које треба сачувати и обрадити. Поред појединаца, велике количине информација стварају школе, универзитети, банке, болнице, државне институције и многе друге организације. За њихово правилно функционисање потребно је да подаци буду доступни у сваком тренутку и да могу брзо да се пронађу и обраде.

Са развојем информационих технологија знатно су порасли и захтеви у погледу управљања информацијама. Није довољно да подаци само буду сачувани, већ треба да буду безбедни, тачни и лако доступни корисницима који имају право да их користе. Поред тога, потребно је да се нове информације лако додају, постојеће мењају, а непотребни подаци уклањају. Из ових разлога, савремени информациони системи користе посебна решења за организовање великих количина података и управљање њима.

3.1.1 ОД ДАТОТЕКА ДО САВРЕМЕНИХ СИСТЕМА



У почетној фази развоја информационих система подаци су се најчешће чували у засебним датотекама. Сваки програм користио је сопствени начин организовања информација, а подаци потребни за рад записивали су се у различите документе. Овакав начин рада био је сасвим довољан док је количина информација била релативно мала и док је информационе системе користио ограничен број корисника.

Временом се број података знатно повећао. Исте информације почеле су да се користе у више различитих програма, а често је било потребно да се неки податак унесе на више места. На пример, ако би одређени корисник променио адресу или број телефона, промену је требало извршити у свим датотекама у којима су се налазиле те информације. Уколико нека датотека није била ажурирана, могле су да се појаве различите вредности за исти податак, што је доводило до грешака и непоузданости у раду.

Проблем је представљало и претраживање информација. Са повећањем броја података, њихово проналажење захтевало је све више времена и ресурса. Поред тога, често понављање истих информација непотребно је заузимало додатни простор за складиштење и отежавало њихову обраду. Одржавање оваквих система постајало је све сложеније и скупљеРазвој информационих технологија наметнуо је потребу за другачијим начином организовања података. Уместо да буду распоређени у великом броју независних датотека, подаци су почели да се организују у јединственим системима који омогућавају централизовано чување, лакше ажурирање и брже претраживање информација. Овакав приступ представља основу за развој савремених база података.

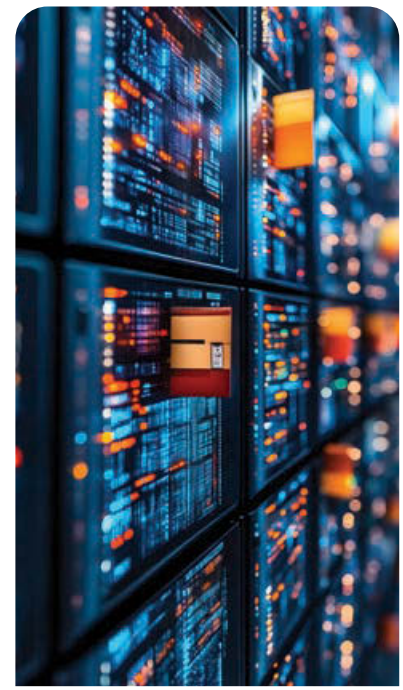
3.1.2 ШТА ПРЕДСТАВЉА БАЗУ ПОДАТАКА?

База података представља организовану збирку података која омогућава њихово безбедно чување, лако претраживање и ефикасну обраду. Њен основни задатак је да обезбеди доступност информација онда када су потребне и омогући њихово једноставно додавање, мењање и коришћење. За разлику од класичног чувања података у независним датотекама, базе података омогућавају да информације буду организоване на начин који побољшава њихову тачност и доступност.

Једна од најважнијих предности база података јесте могућност да више корисника и различити информациони системи користе исте информације. Уколико се дода нови податак или измени постојећа информација, промена одмах постаје доступна свим корисницима који имају право приступа. На овај начин избегава се непотребно понављање података и смањује могућност појаве грешака.

Базе података не користе се само за чување информација. Оне омогућавају и њихово претраживање, упоређивање, сортирање и анализу. Захваљујући овим могућностима, велике количине података могу брзо да се обрађују и користе за доношење различитих одлука. Због тога базе података представљају један од најважнијих делова савремених информационих система.

3.1.3 ПРИМЕНА БАЗА ПОДАТАКА



Базе података данас се примењују у готово свим областима људске делатности. У образовању се користе за вођење евиденције о ученицима, наставницима, предметима и наставним активностима. У здравственим установама примењују се за организовање медицинских информација и података о пацијентима, док у банкама омогућавају безбедно управљање финансијским услугама и трансакцијама.

Базе података имају широку примену и у савременим информационим системима. Користе се у интернет продавницама, на друштвеним мрежама, у системима електронске поште, на платформама за учење на даљину и за многе друге дигиталне услуге. Сваки пут када се корисник пријављује на неку веб-страницу, претражује информације или обавља електронско плаћање, у позадини најчешће ради нека база података.

Развој технологија у облаку, мобилних апликација и вештачке интелигенције додатно повећава значај база података. Савремени информациони системи обрађују милијарде података у секунди, при чему њихова поузданост, тачност и брзина представљају кључне факторе за успешно функционисање дигиталних услуга које свакодневно користимо.

3.1.4 ПРЕДНОСТИ БАЗА ПОДАТАКА

Примена база података пружа многобројне предности у односу на класичан начин чувања информација. Подаци су организовани на систематичан начин, што омогућава њихово брже проналажење и лакше ажурирање. Истовремено се смањује могућност појаве грешака и непотребног понављања информација.

Друга важна предност је могућност да више корисника истовремено ради са истим подацима. Док једни корисници уносе нове информације, други могу да претражују или анализирају постојеће податке. Ово је посебно важно за савремене информационе системе који опслужују велики број корисника.

Базе података омогућавају и већу безбедност информација. Приступ подацима може бити ограничен у складу са потребама корисника, а савремени системи обезбеђују механизме за заштиту и чување информација. Захваљујући овим карактеристикама, базе података представљају незаменљив део савремених информационих технологија и зато се њихова примена стално проширује. Размотримо једну средњу школу у коју се сваке школске године уписују нови ученици. Поред личних података ученика, потребно је евидентирати одељења, наставнике, наставне предмете, распоред часова, успех и многе друге информације. Током године неки подаци се мењају, уписују се нови ученици, а други завршавају своје школовање. Ови унети подаци треба да буду доступни и у будућности. Свако ко је завршио школовање после извесног времена треба да приложи податак у будућим системима или компанијама. Ако се неки документ изгуби, треба да постоји могућност његовог поновног издавања, без обзира на то када је школовање завршено.

Претпоставимо да се све ове информације чувају у великом броју различитих докумената. Ако је потребно пронаћи број телефона неког ученика, списак ученика у одређеном одељењу или податак о одређеном наставнику, биће потребно претражити више различитих докумената. Осим тога, сваку промену биће потребно унети на више места, што повећава могућност појаве грешака.

Коришћењем базе података све информације могу бити организоване у јединственом систему. Нови подаци се лако уносе, постојеће информације ажурирају, а потребни подаци могу брзо да се пронађу. На овај начин знатно се побољшава организација рада и смањује време потребно за обраду информација.

Иста логика примењује се и у многим другим областима. Болнице воде евиденцију о пацијентима, банке о својим корисницима и трансакцијама, библиотеке о књигама, а савремени интернет сервиси о милионима корисничких налога и различитим дигиталним садржајима. Без база података функционисање ових система било би много теже.

Базе података омогућавају да информације буду централизоване и доступне корисницима којима су потребне. Промене се могу лако извршити, а подаци брзо пронаћи и обрадити. Захваљујући овим могућностима, базе података представљају основни део савремених информационих система. Организовано чување информација доприноси не само бржем раду него и већој безбедности и тачности података. Савремене информационе технологије ослањају се на овакве системе како би обезбедиле несметано функционисање различитих дигиталних услуга које се свакодневно користе.

Развој информационих технологија довео је до стварања и обраде огромних количина података. Класичан начин њиховог чувања у независним датотекама временом је показао многобројне недостатке, као што су понављање информација, сложено ажурирање и тешкоће при претраживању података. Због тога се појавила потреба за ефикаснијим начином организовања података и управљања њима.

Базе података представљају савремено решење за безбедно чување и обраду информација. Оне омогућавају да се подаци лако додају и мењају, као и да их користи више корисника, при чему се побољшавају тачност и ефикасност рада. Захваљујући својим карактеристикама, базе података данас имају широку примену у различитим областима савременог друштва. Стицање основних знања о базама података представља важан део информатичке писмености. Разумевање њиховог значаја и начина примене представља основу за изучавање савремених информационих система и њихове практичне примене.

ДА ЛИ ЗНАШ?

Да ли знаш да се сваког дана у свету стварају огромне количине нових података? Велики део њих настаје коришћењем мобилних телефона, интернет сервиса, друштвених мрежа и различитих дигиталних уређаја. За њихово организовање и обраду користе се савремене базе података које могу да складиште огромне количине информација и управљају њима.

ЗАПАМТИ ШТА СИ НАУЧИО

База података представља организовану збирку података. Савремени информациони системи обрађују велике количине информација. Класично чување података у независним датотекама има одређене недостатке. Базе података омогућавају лакше чување, претраживање и ажурирање информација. Оне имају широку примену у различитим областима савременог друштва. Базе података представљају важан део савремених информационих технологија.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Савремени информациони системи могу да обрађују огромне количине података. Велике интернет компаније, научни центри и различите дигиталне платформе свакодневно раде са милијардама информација које се стално стварају и ажурирају. Управљање оваквим подацима представља посебну област у информатици, а развој нових технологија непрестано омогућава бржу и ефикаснију обраду информација.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

1. Одговори на питања.

а) Шта представља базу података?

б) Зашто се користе базе података?

в) Наведи три области у којима се примењују базе података.

2. Тачно или нетачно.

а) Базе података служе само за чување информација.

б) Подаци у бази података могу да се ажурирају.

в) Савремени информатички системи не користе базе података.

3. Размисли и образложи

Који би били највећи проблеми ако банка, болница или школа не би користила базу података за организовање својих информација?

3.2

КРЕИРАЊЕ ТАБЕЛА У БАЗИ ПОДАТАКА

ВЕЋ ЗНАШ ДА...

- објашњаваш шта представља базу података;
- наводиш примере примене базе података;
- објашњаваш зашто се користе базе података;
- анализираш предности од њихове примене;
- користиш табеле за организовање информација.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

- Како најлакше може да се организује велики број информација?
- Зашто подаци треба да буду систематски подређени?
- Како се организују информације у једној бази података?
- Шта се догађа ако подаци нису организовани на прегледан начин?
- Да ли све информације треба да буду смештене у једној табели?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш шта представља релациону базу података;
- разликујеш табелу, поље и запис;
- анализираш организацију података у табели;
- планираш структуру једноставне табеле;
- користиш програме за рад са базама података;
- креираш једноставну табелу.

У свакодневном животу информације се често организују у табеле. Школе воде табеле о ученицима, библиотеке о књигама, спортски клубови о такмичењима, а различите институције о својим корисницима и услугама. Табеларни приказ омогућава да информације буду прегледне, да се лако упоређују и да се потребни подаци брзо проналазе.

Иста идеја примењује се и код база података. Иако савремени информациони системи обрађују огромне количине информација, њихово организовање најчешће почиње табелама. Захваљујући оваквом начину рада, подаци могу систематски да се уносе, ажурирају и користе за различите намене.

Занимљиво је да табеларни приказ представља везу између свакодневног рада и база података. Велики број програма омогућава да се подаци извезу у табеларном облику, а истовремено се табеле креиране у различитим програмима могу унети у базу података. На пример, подаци организовани у табели могу да се увезу у базу података, а подаци из базе могу поново да се извезу ради даље

обраде и анализе. Ова могућност знатно олакшава размену информација између различитих информационих система.

3.2.1 ТАБЕЛА КАО НАЧИН ЗА ОРГАНИЗОВАЊЕ ПОДАТАКА

Када је потребно чувати већи број информација, неопходно је да оне буду организоване на логичан начин. Један од најједноставнијих и најпрегледнијих начина њиховог организовања јесте табеларни приказ. У табели су подаци распоређени тако да могу лако да се уносе, прегледају и упоређују.

Претпоставимо да је потребно водити евиденцију о рачунарима у једној школи. За сваки рачунар треба знати више информација, као што су инвентарни број, произвођач, модел, оперативни систем и просторија у којој се користи. Организовање ових информација у табели омогућава да сваки податак има своје место и знатно олакшава њихово коришћење.

У савременим базама података табеле представљају основни начин организовања информација. Оне омогућавају да подаци буду систематизовани и лако доступни, а истовремено стварају добру основу за даљу обраду и анализу.

Практичан пример

Разгледај следећу табелу

Инвентарни број	Произвођач	Модел	Оперативни систем	Локација
1001	Lenovo	ThinkCentre	Windows 11	Кабинет 1
1002	Dell	OptiPlex	Windows 11	Кабинет 2
1003	HP	ProDesk	Linux	Кабинет 3
1004	Lenovo	ThinkCentre	Windows 11	Кабинет 4

Из табеле се лако може уочити да је сваки рачунар описан помоћу више различитих информација. Ако се набави нов рачунар, подаци о њему могу се једноставно додати. Ако се неки рачунар премести у другу просторију, потребно је променити само одговарајућу информацију.

Овакав начин организације омогућава да се подаци лако прегледају и упоређују. На пример, брзо се може утврдити колико рачунара користи одређени оперативни систем или који се уређаји налазе у одређеној просторији.

Како да анализирамо табелу?

Пажљивим разматрањем претходног примера може се уочити да је табела састављена од колона и редова. Свака колона садржи одређену врсту информација, док сваки ред представља потпун опис једног рачунара. Захваљујући оваквој организацији, подаци су прегледни и лако разумљиви.

Добра организација информација омогућава њихов бржи унос, лакше одржавање и ефикасније коришћење. Због тога правилно креирање табела представља један од најважнијих корака при изради базе података.

3.2.2 РЕЛАЦИОНА БАЗА ПОДАТАКА

У пракси се сви подаци веома ретко смештају у једну једину табелу. Ако би се у једну табелу уписивале све информације о ученицима, наставницима, предметима, оценама, распореду и изостанцима, она би постала превелика, сложена за одржавање и подложна грешкама. Због тога савремене базе података организују информације у више међусобно повезаних табела. Овај начин организације назива се релационим моделом, а базе које га користе називају се релационим базама података.

Основна идеја релационог модела јесте да свака табела садржи податке о једној одређеној целини. На пример, у једном школском информационом систему може да постоји посебна табела за ученике, посебна табела за наставнике, посебна табела за предмете и посебна табела за оцене. Уместо да се исти подаци понављају на више места, табеле се међусобно повезују преко заједничких поља. На тај начин смањује се могућност појаве грешака, а подаци се чувају организованије и ефикасније.

Релационе базе података данас су најраспрострањенија врста база података. Оне се користе у образовним институцијама, банкама, здравственим установама, државним институцијама, интернет продавницама и многобројним другим информационим системима. Захваљујући њиховој структури, велике количине података могу брзо да се уносе, претражују, ажурирају и анализирају.

Пример организације података у више табела

Претпоставимо да је потребно водити евиденцију о ученицима и њиховим оценама. Уместо да се све информације сместе у једну табелу, могу се креирати две табеле.

Табела: Ученици

ID	Име	Презиме
1	Ана	Петрова
2	Марко	Стојанов
3	Елена	Николовска

Табела: Оцене

ID	УченикID	Предмет	Оценка
1	1	Информатика	5
2	1	Математика	4
3	2	Информатика	5
4	3	Математика	5

У првој табели чувају се основни подаци о ученицима, а у другој њихове оцене. Уместо да се сваки пут понављају име и презиме ученика, користи се идентификациони број који омогућава да се две табеле повежу. На тај начин смањује се понављање података и побољшава организација информација.

Зашто се користи више табела?

Једна од најважнијих предности релационих база података јесте смањење броја дуплираних података. Уколико се подаци о ученику чувају само једном, сваку промену биће потребно унети на једном месту. Ако ученик промени презиме или ако је потребно исправити неки податак, измена ће бити једноставна и неће постојати опасност да део записа остане непромењен.

Организовање података у више табела омогућава и брже претраживање информација. Уместо да програм претражује огромну табелу са хиљадама колона и редова, он ради са мањим и боље организованим табелама. Ово је посебно важно за савремене информационе системе који обрађују милионе записа.

Поред тога, овакав начин рада омогућава лакше проширивање базе података. Ако се у будућности појави потреба за новим информацијама, може се додати нова табела без значајних измена постојеће структуре. Због тога релационе базе података представљају стандард у савременим информационим системима.

3.2.3 ПРОГРАМИ ЗА РАД СА БАЗАМА ПОДАТАКА

За креирање, организовање и обраду података користе се специјализовани програми познати као системи за управљање базама података. Ови програми обезбеђују алате за унос, уређивање, претраживање и приказивање података, при чему корисник не мора непосредно да води рачуна о начину на који се информације физички чувају у рачунару.

У образовању и при изучавању основа база података често се користе програми Microsoft Access и LibreOffice Base. Они обезбеђују графичко радно окружење у којем корисник може да креира табеле, уноси податке, успоставља везе између табела и израђује обрасце, упите и извештаје. Захваљујући визуелном приступу, ови програми су погодни за изучавање основних принципа рада са базама података.

Иако постоје и многи други системи за управљање базама података који се користе у великим компанијама и институцијама, принципи рада су слични. Без обзира на то да ли се користи једноставан образовни алат или сложен корпоративни систем, подаци се организују у табеле, успостављају се везе између њих и обезбеђује се могућност њиховог претраживања и обраде.

Важно је разумети да програм представља алат за рад са базом података, а не саму базу података. База података је организована збирка информација, док програм обезбеђује средства помоћу којих корисник ради са тим информацијама.

Креирање табеле

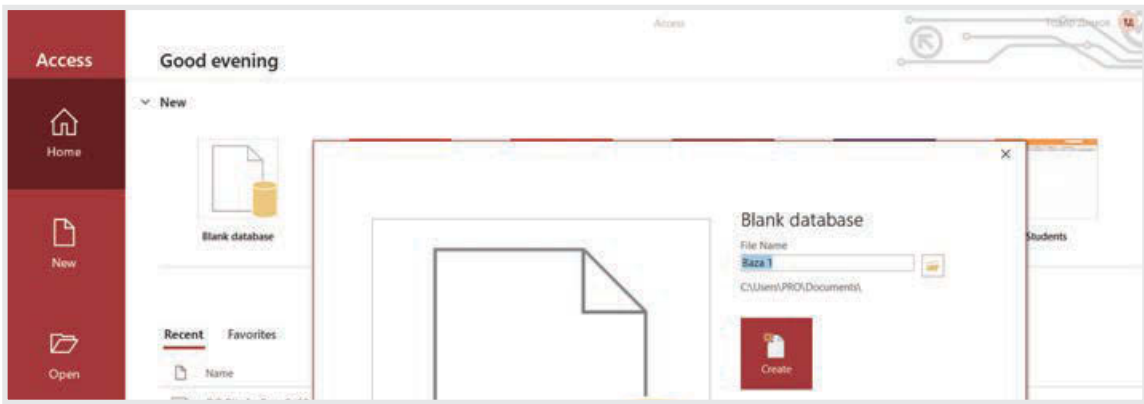
Пре него што започне унос података, потребно је пажљиво испланирати које ће се информације чувати у бази података. Овај корак је важан зато што квалитет организације директно утиче на ефикасност рада. Ако је табела лоше осмишљена, касније се могу појавити проблеми при уносу, претраживању и ажурирању података.

При креирању табеле најпре се одређују поља која ће она садржати. Свако поље намењено је одређеној врсти информација. На пример, уколико се креира табела за ученике, потребно је одредити поља за име, презиме, датум рођења, адресу, број телефона и друге потребне податке.

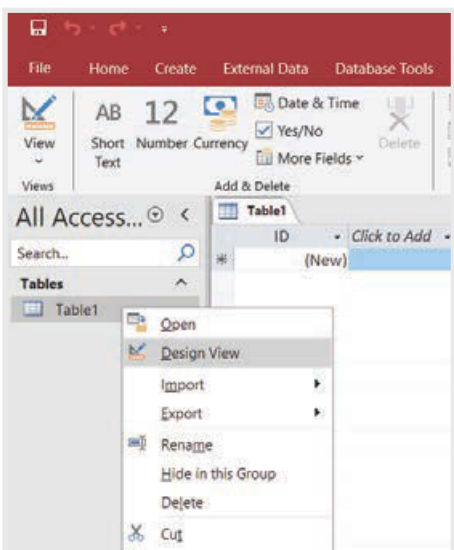
Добар избор поља омогућава да подаци буду организовани, прегледни и лаки за обраду. Због тога креирање табеле не представља само технички поступак него и процес пажљивог планирања информација које ће се чувати у бази података.

Пример: Планирање табела за ученике

MS Access, Након покретања програма потребно је изабрати празну базу података. Новој бази додељује се име и бира се локација на којој ће бити сачувана целокупна база података.



Претпоставимо да школа жели да води електронску евиденцију о ученицима. Пре креирања табеле потребно је размислити о томе које ће информације бити потребне за сваког ученика. Може се утврдити да је потребно чувати име, презиме, датум рођења, адресу, број телефона и адресу електронске поште.

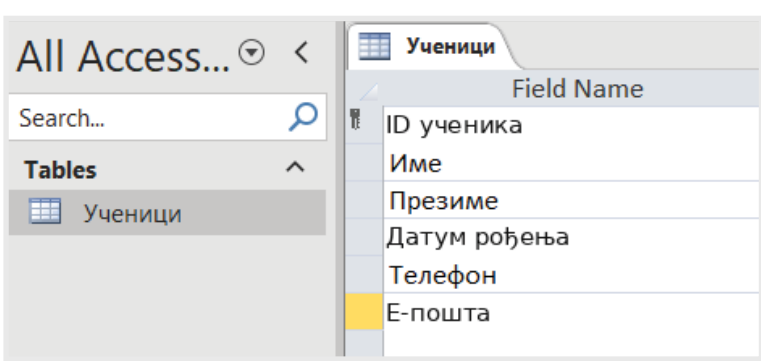


Да би се подесила поља са потребним елементима, приступа се приказу дизајна табеле.

Одмах по избору табела се чува под именом које јој додељује корисник. Пожељно је додељивати имена која указују на врсту података које ће табела садржати.

На основу ове анализе може да се испланира следећа структура:

ID ученика, Име,	Презиме,	Датум рођења,	Адреса,	Телефон,	Е-пошта



Из примера се може уочити да свака колона представља одређену врсту података који ће се уносити за све ученике. Овакво планирање представља први корак у процесу креирања табеле.

Анализа примера

При креирању табеле није довољно само унети колоне са различитим именима. Потребно је пажљиво одредити које су информације заиста потребне. Ако се додају непотребна поља, табела ће постати сложена и тежа за одржавање. С друге стране, уколико недостају важна поља, касније ће бити потребне додатне измене структуре.

Планирање табела представља један од најважнијих корака при пројектовању базе података. У пракси се, пре него што започне израда система, често врши детаљна анализа информација које треба чувати и начина на који ће се користити. На тај начин обезбеђују се ефикаснија организација и лакши рад са подацима.

ЗАКЉУЧАК:

Табеле представљају основни начин организовања података у релационим базама података. Оне омогућавају да информације буду прегледне, лаке за унос и једноставне за обраду. Правилна организација табела представља први корак ка стварању квалитетне базе података.

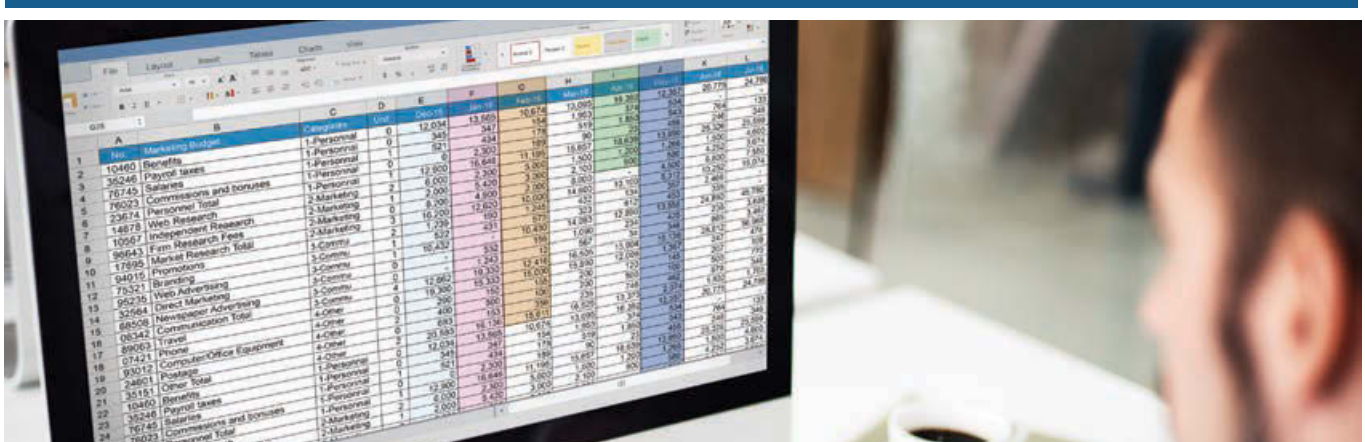
Свака табела садржи поља и записе који омогућавају систематско организовање информација. Захваљујући оваквој структури, подаци се могу лакше претраживати, ажурирати и користити у различитим информационим системима.

ДА ЛИ ЗНАШ?

Да ли знаш да је највећи део података које свакодневно користимо организован у табеле? Без обзира на то да ли је реч о школској евиденци банкарском систему или интернет продавници, основу организације најчешће чини табеларни приказ информација.

ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представља релациона база података?
2. Шта представља табела?
3. Која је улога поља у табели?
4. Шта представља запис?
5. Зашто се подаци организују у више табела?
6. Који програми могу да се користе за рад са базама података?
7. Зашто је важно планирање табеле пре њеног креирања?



ПРАКТИЧНИ ЗАДАЦИ

Задатак 1

Направи план табеле у којој ћеш чувати податке о књигама у библиотеци.

Задатак 2

Одреди која од следећих информација представљају поља:

- Име ученика
- Марко Петров
- Телефон
- Кабинет 12
- Датум рођења

Задатак 3

У следећој табели означи колико поља и колико записа има.

Име	Презиме	Клас
Ана	Петрова	I-1
Марко	Стојанов	I-2
Елена	Николовска	I-1

Задатак 4

Објасни зашто није добра идеја да се сви подаци неке школе чувају у једној јединој табели.

ИСТРАЖИВАЧКИ ЗАДАТАК

Истражи који програми за рад са базама података су данас најчешће коришћени. За сваку пронађи:

- име;
- произвођача;
- да ли је бесплатна или је комерцијална;
- област примене.

ЗАПАМТИ ШТА СИ НАУЧИО

Табела је основни елемент релационе база података.

Поље представља колону у табели. Запис представља ред у табели.

Релационе базе података садрже више међусобно повезаних табела.

Подаци могу да се увозе и да се извозе у табеларном облику.

Добро планирана табела омогућава лакши рад са подацима.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У великим компанијама и институцијама базе података садрже хиљаде табела и милионе записа. Упркос томе, захваљујући доброј организацији табела, потребне информације могу се пронаћи за делић секунде.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

Заокружи тачан одговор

Поље представља:

а) ред
б) колона
в) табела
г) база података

Запис представља:

а) колону
б) поље
в) ред
г) базу података

Тачно или нетачно

1. Једна база података може да са-држи више табела.
2. Поље и запис су исти појмови.
3. Табеле се користе за организовање података.

3.3

ТИПОВИ ПОДАТАКА

ВЕЋ ЗНАШ ДА...

- објашњаваш шта представља табела;
- разликујеш поље и запис;
- планираш структуру једноставне табеле;
- креираш табелу у бази података;
- организујеш информације у табеларном облику.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

- Да ли у свим пољима могу да се уносе исте врсте података?
- Како компјутер препознава да ли је одређена вредност текст, број или датум?
- Зашто је важно да постоје правила за уношење података?
- Шта може да се догоди ако корисник унесе нетачну вредност?
- Како може да се побољша тачност података у бази?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- разликујеш разне типове података;
- бираш одговарајући тип података;
- објашњаваш својства поља;
- примењујеш правила за валидацију;
- анализираш начине за контролу унетих података;
- препознајеш значење квалитетних и тачних информација.

При креирању табеле није довољно само одредити поља која ће она садржати. За свако поље потребно је одредити и коју врсту података ће садржати. Ово је важно зато што се различите информације записују на различите начине. На пример, име и презиме састоје се од знакова и слова, датум рођења уноси се у формату датума, а број изостанака или оцене најчешће се уносе као бројчане вредности. Одређивање одговарајућег типа податка омогућава програму да правилно обрађује информације. Ако се у поље намењено бројевима унесе текст или се у поље за датум унесе неодговарајућа вредност, може доћи до грешака при обради података. Због тога се већ приликом креирања табеле дефинише које вредности свако поље може да садржи. У савременим системима за рад са базама података постоји више типова података. Иако се њихови називи могу мало разликовати од једног програма до другог, њихова намена је слична. Најчешће се користе текстуални подаци, бројчани подаци, датуми и времена, логичке вредности и поља за дужи текст.

3.3.1 НАЈЧЕШЋЕ КОРИШЋЕНИ ТИПОВИ ПОДАТАКА

Текстуални подаци

Текстуални тип података користи се за унос речи, имена, адреса, телефонских бројева и других информација које се не користе у математичким прорачунима. Иако телефонски број садржи цифре, он се најчешће чува као текст зато што се не сабира, не одузима и не множи.

Примери:

- Тодор Димов
- Свети Николе
- II-3
- 071123456

Бројчани подаци

Бројчани тип података користи се када податке треба математички обрађивати. Над њима се могу вршити прорачуни, одређивати просечне, максималне и минималне вредности и спроводити друге статистичке анализе.

Примери:

- 5
- 18
- 125
- 98.5

Датум и време

Овај тип користи се за унос датума и временских информација. Захваљујући њему може да се израчуна узраст ученика, трајање одређених активности или временска разлика између два догађаја.

Примери:

- 15.09.2025
- 21.05.2026
- 08:30

Логички подаци

Овај тип користи се за унос датума и временских информација. Захваљујући њему може да се израчуна узраст ученика, трајање одређених активности или временска разлика између два догађаја.

Примери:

- Да
- Не

или

- True
- False

Дужи текст

Логички тип омогућава унос само две вредности. У различитим програмима оне могу бити представљене као Да/Не, Тачно/Нетачно или Укључено/Искључено.

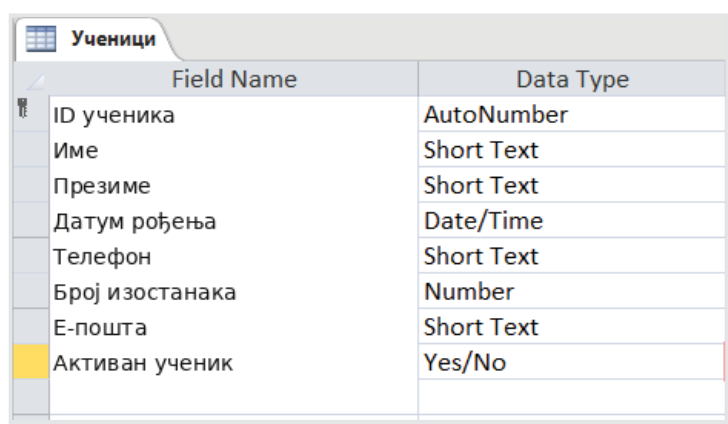
Пример:

„Ученик активно учествује на такмичењима из информатике и освојио је више награда на регионалним и државним такмичењима."

Практичан пример

Разгледати табелу за ученике.

Поље	Тип података
ID ученика	Број
Име	Текст
Презиме	Текст
Датум рођења	Датум/Време
Број изостанака	Број
Е-пошта	Текст
Активан ученик	Логички



Слика 1 Тип података

Из примера се може уочити да свако поље има одређени тип података. Захваљујући овој организацији, програм зна какве вредности могу да се уносе и како треба да се обрађују. На пример, број изостанака може да се сабира и анализира, док код имена то нема смисла.

Зашто је важно правилно изабрати тип података?

Избор одговарајућег типа података утиче на тачност и ефикасност рада са базом података. Када би се све информације чувале као обичан текст, програм не би могао правилно да врши прорачуне, сортира податке или проверава да ли су унете вредности исправне.

Поред тога, правилан избор доприноси бољем искоришћењу меморије и бржој обради информација. Код великих база података које садрже милионе записа овакве оптимизације имају значајан утицај на рад система.

Пажљивим одређивањем типа података за свако поље ствара се добра основа за поуздану и ефикасну базу података. Овај корак представља један од најважнијих делова процеса креирања табела.

Својства поља

При креирању табеле за свако поље не одређују се само његово име и тип података. Додатно се могу подесити различита својства која одређују начин уноса, приказивања и обраде информација. Захваљујући овим својствима, подаци се уносе на контролисанији начин и смањује се могућност појаве грешака.

Претпоставимо да у једној табели постоји поље за име ученика. Ако се не поставе никаква ограничења, корисник може да унесе празну вредност, предугачак текст или насумичне знакове који немају никакво значење. Коришћењем својстава поља могу се унапред одредити правила која ће омогућити квалитетнији унос података.

Својства поља представљају важан механизам за контролу информација у бази података. Она омогућавају стандардизацију уноса, бољу организацију података и већу поузданост при њиховој обради.

Најчешће коришћена својства

Величина поља

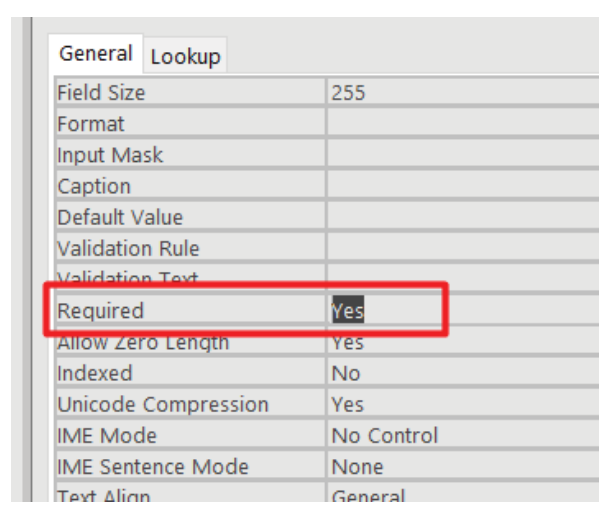
Код текстуалних података често се одређује максималан број знакова који се могу унети у поље. На пример, ако се зна да шифра ученика увек садржи највише десет знакова, нема потребе да се дозвољава унос много дужег текста.

Ограничавањем величине поља смањује се могућност уноса неодговарајућих података и обезбеђује ефикасније коришћење меморијског простора.

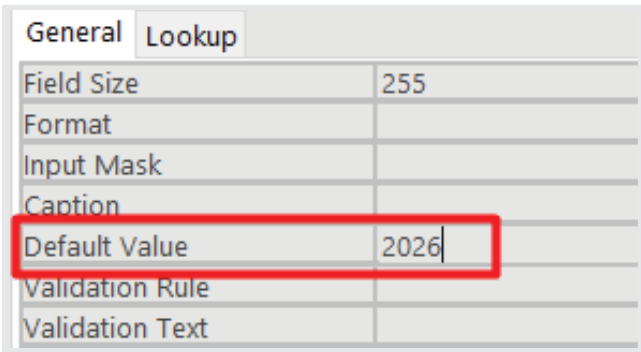
Обавезно поље

Понекад је унос одређених информација обавезан. На пример, у школској евиденцији не би било прихватљиво да постоји ученик/ученица без имена или презимена. У таквим ситуацијама користи се својство које не дозвољава чување записа ако је поље празно.

Ово својство доприноси побољшању квалитета података и спречава појаву непотпуних записа. У програму MS Access ово се подешава у одељку General табеле, где се, међу бројним могућностима, може изабрати „Да“ за својство Обавезно (Required).



Слика 2 Обавезно поље



Слика 3 Вредност која се појављa автоматски

Подразумевана вредност

У одређеним ситуацијама велики број записа има исту почетну вредност. Уместо да корисник стално уноси исту информацију, може се унапред поставити подразумевана вредност.

На пример, уколико већина ученика припада текућој школској години, та вредност може аутоматски да се појављује при уносу новог записа. Корисник ће је променити само ако је потребно.

Формат података

Одређене информације треба да буду приказане на стандардан начин. Датуми, бројеви и новчане вредности најчешће имају унапред одређен формат.

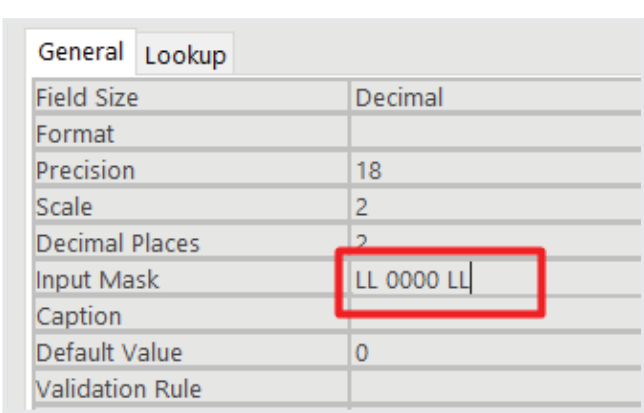
Захваљујући овом својству, сви подаци истог типа приказују се на уједначен и прегледан начин, што побољшава читљивост и разумљивост информација.

Валидација података

У свакој бази података постоји могућност да корисници грешком унесу нетачне информације. Може се унети текст уместо броја, насумична вредност уместо датума или податак који није у складу са правилима система. Да би се спречиле овакве ситуације, користи се валидација података.

Валидација представља поступак којим се проверава да ли унета вредност испуњава унапред одређене услове. Ако вредност није у складу са правилима, систем упозорава корисника и не дозвољава чување нетачних информација.

Уколико се пројектује систем за обрачун наплате паркирања путем SMS поруке, потребно је проверити да ли је унета регистарска ознака у исправном формату. Људи често греше при уносу регистарских ознака. У пољу „Input mask“ на картици „General“ може се подесити валидација. Уколико очекујемо да се у систем уносе слова и цифре у формату регистарских ознака, потребно је задати да ознака садржи два слова, затим четири цифре и на крају два слова. При томе „L“ означава да се на тој позицији може унети искључиво слово, док „0“ означава позицију за цифру. Овако подешена маска проверавање да ли је регистарска ознака исправно унета.



Слика 4 постављање маске за унос

Овај механизам има велики значај у информационим системима који обрађују велике количине података. Чак и мала грешка при уносу може изазвати проблеме приликом претраживања, анализе или израде извештаја. Због тога савремене базе података користе различите технике за проверу унетих информација.

Примери валидације

Разгледати следеће ситуације:

Пример 1

Поље: Узраст

Правило: Вредност мора да буде између 14 и 19.

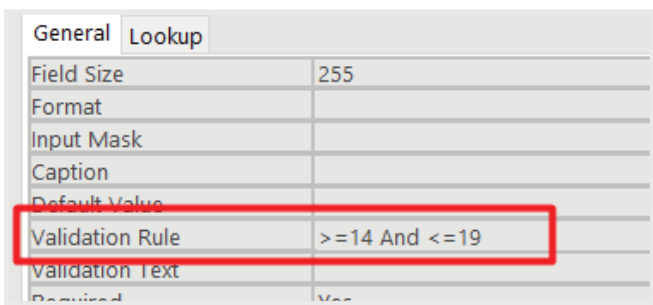
√ 16

√ 18

x -5

x 120

У овом случају систем ће прихватити само реалне вредности за узраст ученика. Овако треба да изгледа валидација података.



Слика 5 постављање правила за валидацију

Пример 2

Поље: Оцена

Правило: Дозвољене су само вредности од 1 до 5

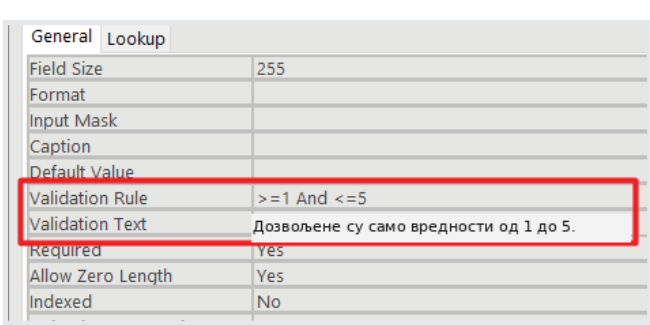
√ 1

√ 5

x 7

x 0

Овим правилом се спречава унос оцена које нису унете у систем за оцењивање.



Слика 6 Постављање на правила и текст за валидација

Текст за валидацију додатно објашњава шта поље треба да садржи. Добра је пракса да се при коришћењу валидације постави и текст за валидацију. У том случају, када програм упозори на грешку, поред звучног сигнала приказује се и порука о грешци.

Пример 3

Поле: Е-пошта

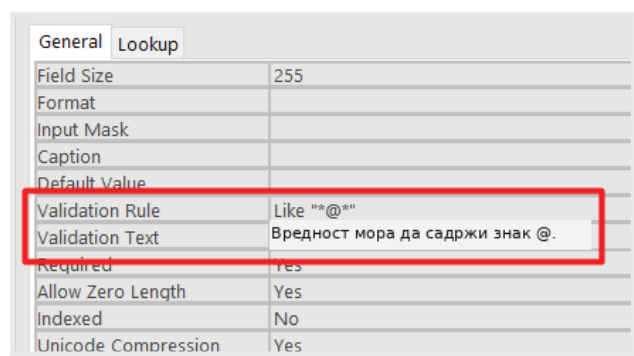
Правило: Вредност мора да го содржи знакот @.

√ učenik@uciliste.mk

x ucenik.gmail.com

Валидација помаже да се открију очигледне грешке још при уношењу података.

Слика 7 Валидација са обавезним знаком



Звездице испред и иза траженог знака означавају да испред и иза тог знака обавезно мора да постоји текст.

Анализа примера

Из претходних примера може се уочити да валидација не служи за улепшавање података, већ за заштиту базе података од нетачних информација. Што је већи број корисника који уносе податке, то је већа потреба за аутоматском контролом вредности.

Комбинација типова података, својстава поља и правила за валидацију ствара поуздану основу за рад са информацијама. Захваљујући овим механизмима, подаци постају конзистентнији, прецизнији и лакши за обраду.

ЗАКЉУЧАК:

Тип података и својства поља имају велики утицај на квалитет информација у бази података. Правилан избор омогућава да се подаци лакше обрађују, претражују и анализирају.

Валидација и својства поља представљају важне механизме за контролу унетих информација. Они помажу да се смањи могућност појаве грешака и обезбеђују већу поузданост и тачност података.

ДА ЛИ ЗНАШ?

Да ли знаш да рачунар различито обрађује текст, бројеве и датуме? Захваљујући типовима података, систем зна да ли може да врши прорачуне са одређеном вредношћу или треба само да је приказује.

ПИТАЊА ЗА ПОНАВЉАЊЕ

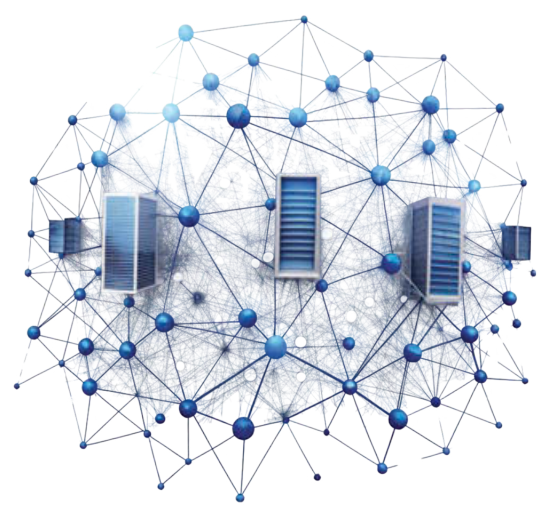
1. Шта представља тип податка?
2. Који типови података се најчешће користе?
3. Када се користи текстуални тип?
4. Када се користи бројчани тип?
5. Шта представља валидација?
6. Зашто се користе својства поља?
7. Шта је подразумевана вредност?

ПРАКТИЧНИ ЗАДАЦИ

Задатак 1

За свако поље одреди тип података:

Поље
Име
Датум рођења
Изостанци
Активан ученик
Напомена



Задатак 2

Наведи три поља за која би поставио обавезно уношење података.

Задатак 3

Наведи пример за поље које би имало подразумевану вредност.

Задатак 4

Објасни зашто узраст треба да буде бројчани, а не текстуални податак.

ИСТРАЖИВАЧКИ ЗАДАТАК

Истражи који типови података користе Microsoft Access и LibreOffice Base. Направи поређење и пронађи који типови су заједнички за оба програма.

ЗАПАМТИ ШТА СИ НАУЧИО

Свако поље има одређени тип података.

Тип података одређује какве вредности могу да се уносе.

Својства поља омогућавају контролу над подацима.

Валидација спречава унос нетачних вредности.

Правилан избор типова података побољшава ефикасност базе података.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У професионалним системима постоје десетине различитих типова података. Неки од њих користе се за складиштење фотографија, видео-записа, географских координата или сложених докумената.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

Заокружи тачан одговор

1. За датум рођења најбољи тип је:

а) текст

б) број

в) датум/време

г) логички

2. Валидација служи за:

а) брисање записа

б) проверу унетих података

в) штампање извештаја

г) креирање табела

Тачно или нетачно

1. У поље за узраст може да се уноси било који текст.

2. Логички тип најчешће има две могуће вредности.

3. Валидација побољшава квалитет података.

3.4

ПРИМАРНИ КЉУЧ И ПОВЕЗИВАЊЕ ТАБЕЛА

ВЕЋ ЗНАШ ДА...

- креираш табеле у бази података;
- користиш различите типове података;
- примењујеш својства поља
- користиш правила за валидацију;
- организујеш информације у табеларном облику.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

- Шта ће се десити ако у два записа постоје исти подаци?
- Како могу са сигурношћу да се разликују два ученика са истим именом и презименом?
- Зашто сваки запис треба да има јединствен идентификатор?
- Како компјутер брзо проналази потребан запис?
- Како може у различитим табелама да се користе исте информације без понављања?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш шта представља примарни кључ;
- препознајеш поља погодна за примарни кључ;
- анализираш значење јединствене идентификације;
- објашњаваш предности коришћења примарног кључа;
- примењујеш примарни кључ при креирању табеле;
- анализираш организацију података у релационој бази.

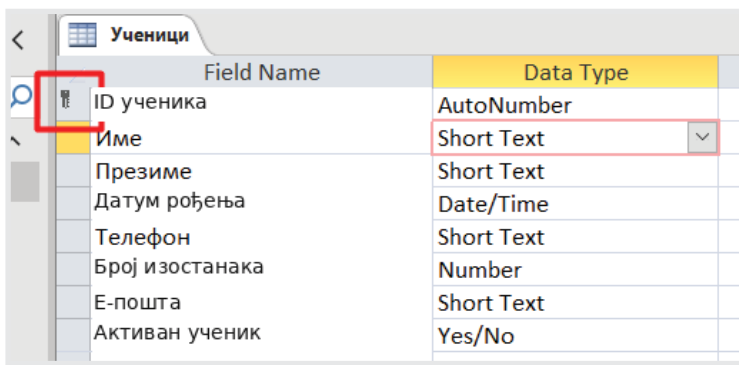
При раду са већим бројем записа често се јавља потреба да сваки запис може недвосмислено да се препозна. Претпоставимо да се у једној табели чувају подаци о ученицима. Могуће је да два ученика имају исто име и презиме, па се само на основу тих података не може са сигурношћу утврдити о ком ученику је реч. Због тога се у базама података користи посебно поље које омогућава јединствену идентификацију сваког записа.

Поље које садржи јединствену вредност за сваки запис назива се примарним кључем. Његов основни задатак је да обезбеди да сваки запис у табели буде јединствен и лако препознатљив. Вредност примарног кључа не сме да се понавља код два различита записа и не сме да остане DENIED празна.

Примарни кључ има велики значај за правилно функционисање базе података. Осим што обезбеђује јединствену идентификацију записа, он представља основу за повезивање табела и организовање сложених информационих система. Из тог разлога, већ при креирању табеле потребно је пажљиво изабрати поље које ће имати ову улогу.

3.4.1 КАРАКТЕРИСТИКЕ ПРИМАРНОГ КЉУЧА

Да би неко поље могло да се користи као примарни кључ, потребно је да испуњава неколико услова. Прво, вредност мора да буде јединствена за сваки запис. Уколико две особе имају исти идентификациони број, систем неће моћи правилно да их разликује. Друго, вредност не сме да буде празна, јер сваки запис мора да има свој јединствени идентификатор. Треће, пожељно је да вредност буде стабилна и да се ретко мења, јер честе промене могу изазвати проблеме при организовању података.



Слика 1 Постављен примарни кључ

У многим случајевима као примарни кључ користи се посебно поље које аутоматски добија нову вредност при уносу сваког новог записа. На овај начин избегава се могућност да корисник случајно унесе већ постојећу вредност.

Пример

Разгледати следећу табелу.

Ученик ID	Име	Презиме
1	Ана	Петрова
2	Ajet	Rexhepi
3	Ана	Петрова
4	John	Wayne

Из табеле се може уочити да се име и презиме „Ана Петрова“ појављују два пута. Када би се идентификација вршила само на основу имена и презимена, било би тешко разликовати та два записа.

Поље УченикID садржи различиту вредност за сваког ученика и због тога се може користити као примарни кључ. Без обзира на то колико ученика има исто име или презиме, њихови идентификациони бројеви биће различити. У овом случају као идентификациони број може се користити неки јединствен податак о ученику, на пример, број у дневнику. У једном одељењу може бити само један ученик са бројем 3. У образовном систему користи се ЕИБУ (јединствени идентификациони број ученика), у Министарству унутрашњих послова број личне карте, у Министарству спољних послова број пасоша, а у Министарству здравља број здравствене књижице. Сви ови записи повезани су са ЈМБГ-ом (јединственим матичним бројем грађанина). Због законских прописа не може свако

да има приступ ЈМБГ-у неке особе, па свака институција осмишљава посебан начин за разликовање грађана са истим личним подацима. Уколико нека организација или компанија издаје картице својим клијентима и жели да ти бројевибуду узастопни, као тип података користи се AutoNumber. Ово је веома корисна опција и код фискалних каса у продавницама, које дневно издају и по стотине фискалних рачуна.

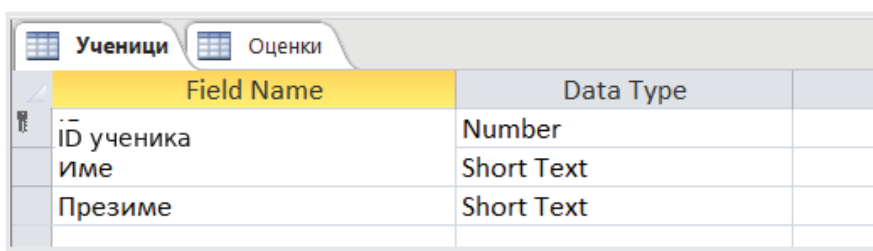
Анализа примера

Из претходног примера може се уочити да примарни кључ не мора да има значење за корисника. Његова улога је да обезбеди јединственост записа и правилну организацију података. У пракси корисници најчешће виде имена, адресе или друге информације, док систем у позадини користи примарне кључеве за поуздано управљање подацима.

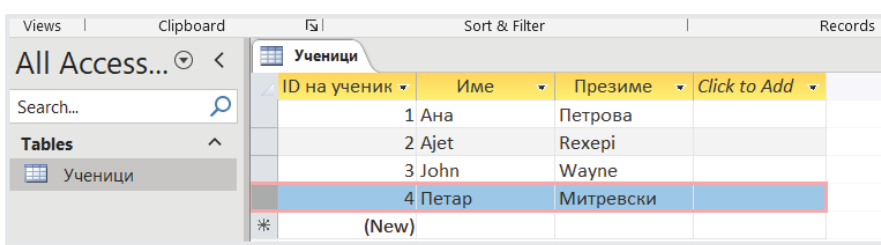
Употреба примарног кључа знатно олакшава претраживање, ажурирање и одржавање информација. Захваљујући њему, сваки запис може брзо да се пронађе и обради без опасности од мешања са другим записима који садрже сличне податке.

3.4.2 ПОВЕЗИВАЊЕ ТАБЕЛА

У већим базама података информације се најчешће организују у више табела. Да би се те табеле могле заједнички користити, потребно је да између њих постоји веза. Повезивање табела омогућава да се информације ефикасније организују и да се избегне непотребно понављање података.

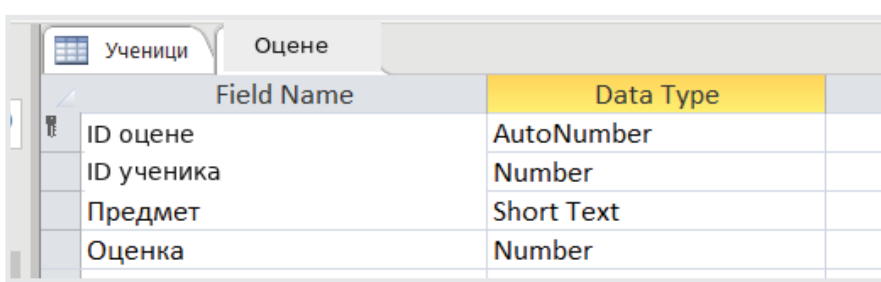


Слика 2 Уређивање табеле са подацима о ученицима

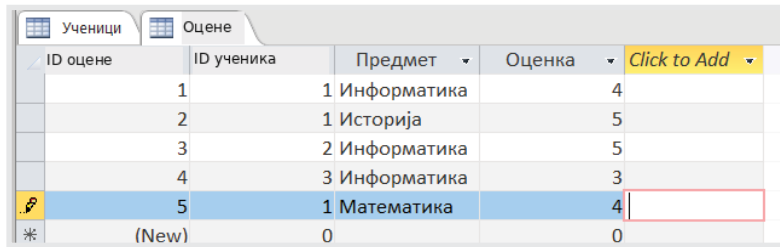


Слика 3 Табела са подацима о ученицима

Претпоставимо да постоји табела са ученицима и посебна табела са оценама. Уместо да се у сваки запис о оцени поново уписују име и презиме ученика, може се користити идентификациони број из табеле са ученицима. На тај начин информације се чувају само једном, а ипак се могу лако повезати када је потребно.

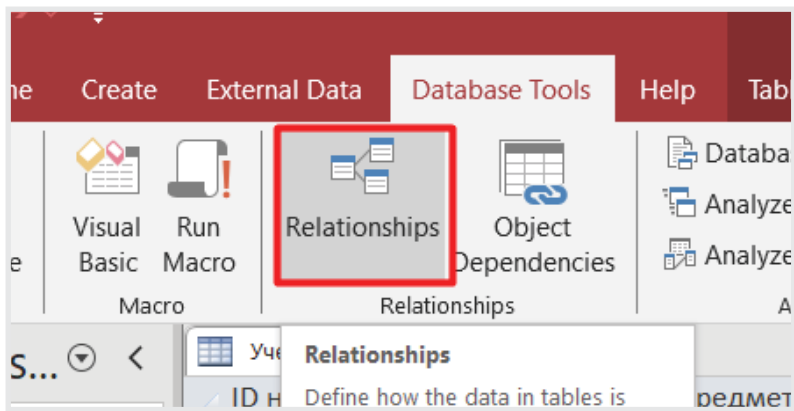


Слика 4 Табела са подацима о ученицима



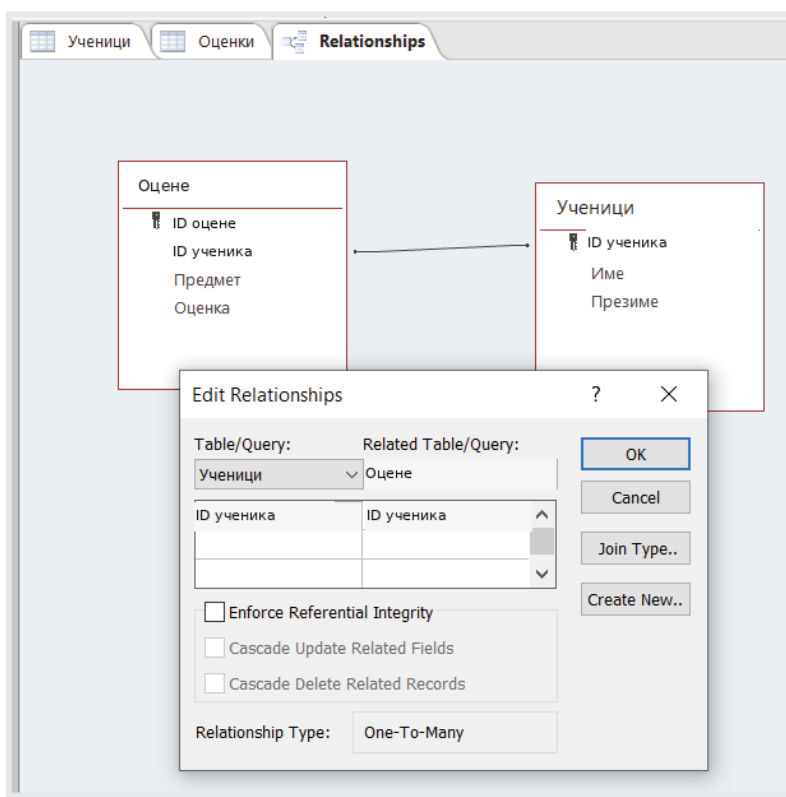
Слика 5 Попуњавање оцена у табели

Повезивање табела представља једну од најважнијих карактеристика релационих база података. Захваљујући њему, подаци остају организовани, а систем може ефикасно да ради и са великим бројем информација.



Слика 6 Постављање веза у бази података

При успостављању веза потребно је да елементи који се повезују буду истог типа. У овом случају две табеле биће повезане преко поља ID ученика.



Слика 7 Постављање веза између табела

Уређивање и ажурирање табела

Базе података ретко остају непромењене. Временом се јавља потреба за додавањем нових поља, мењем постојећих својстава или уносом нових записа. Због тога савремени системи омогућавају лако уређивање и ажурирање табела.

Уређивање табеле може да обухвати додавање новог поља, промену имена постојећег поља или промену његовог типа података. Овакве промене најчешће се врше када се појаве нови захтеви или када је потребно боље организовати информације.

Ажурирање табела односи се на измену података. На пример, може се променити број телефона, адреса или друга информација која већ постоји у бази података. Редовно ажурирање неопходно је да би подаци остали тачни и корисни.

Добро организоване табеле, правилно изабрана поља и пажљиво ажурирање података представљају основу за поуздану и ефикасну базу података. Захваљујући овим активностима, информације остају тачне, организоване и лако доступне корисницима.

ЗАКЉУЧАК

Примарни кључ представља један од најважнијих елемената релационих база података. Његов основни задатак је да обезбеди да сваки запис буде јединствен и лако препознатљив. Захваљујући примарном кључу, подаци се могу поуздано организовати и одржавати.

Коришћење примарног кључа повећава ефикасност претраживања и ажурирања информација, као и управљања њима. Он представља основу за стварање већих и сложенијих база података.

ДА ЛИ ЗНАШ?

Да ли знаш да две особе могу да имају исто име и презиме, али не и исти матични број? Сличан принцип примењује се и у базама података помоћу примарног кључа.

ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представља примарни кључ?
2. Које услове треба да испуњава примарни кључ?
3. Зашто вредности примарног кључа не смеју да се понављају?
4. Да ли примарни кључ може бити празан?
5. Које су предности коришћења примарног кључа?

ПРАКТИЧНИ ЗАДАЦИ

Задатак 1

Одреди које поље би било најбоље за примарни кључ!

| Шифра | Производ | Количина |

Задатак 2

Наведи три примера поља која могу да се користе као примарни кључ!

Задатак 3

Објасни зашто име и презиме нису увек добар избор за примарни кључ!

Задатак 4

Размисли шта би се догодило ако два записа имају исту вредност примарног кључа!

ИСТРАЖИВАЧКИ ЗАДАТАК

Истражи како различите институције обезбеђују јединствену идентификацију лица, производа или докумената. Пронађи најмање три примера и објасни како функционишу.

ЗАПАМТИ ШТА СИ НАУЧИО

- Примарни кључ омогућава јединствену идентификацију записа.
- Његове вредности не смеју да се понављају.
- Не сме да садржи празне вредности.
- Олакшава претраживање и ажурирање података.
- Представља важан елемент у организацији релационих база података.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У великим базама података примарни кључеви често се генеришу аутоматски. На тај начин систем гарантује да ће сваки нови запис добити јединствени идентификациони број.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

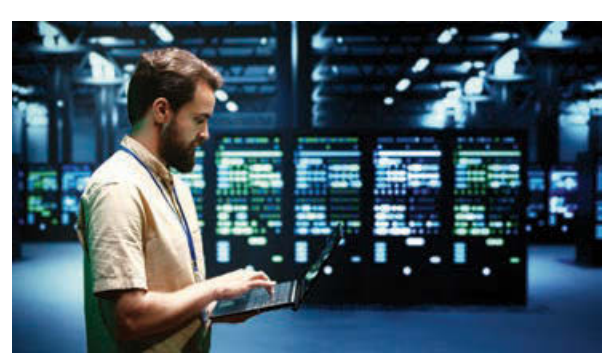
Заокружи тачан одговор

1. Примарни кључ:
а) може да се понавља
б) мора да буде јединствен
в) није важан
г) служи за форматирање

2. Примарни кључ се најчешће користи за:
а) идентификација записа
б) штампање извештаја
в) брисање табела
г) уношење текста

Тачно или нетачно

1. Два различита записа могу да имају исти примарни кључ.
2. Примарни кључ може да буде празан.
3. Примарни кључ помаже при организацији података.



3.5

ОБРАСЦИ ЗА УНОС И ПРЕГЛЕД ПОДАТАКА

ВЕЋ ЗНАШ ДА...

- креираш табеле у бази података;
- користиш различите типове података;
- подешаваш својства поља;
- користиш примарни кључ;
- уносиш и ажурираш податке у табели.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Да ли је практично да корисник увек директно уноси податке у табелама?
2. Како може да се поједностави уношење информација?
3. Да ли сви корисници мора да виде структуру табеле?
4. Како може да се направи прегледнији приказ података?
5. Да ли је могуће уношење података да изгледа као попуњавање електронског обрасца?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш шта представља образац;
- наводиш предности коришћења образаца;
- креираш једноставан образац;
- уносиш податке преко обрасца;
- уређујеш сталне записе;
- анализираш примену образаца у информатичким системима.

У претходним лекцијама подаци су уношени директно у табеле. Иако је овај приступ једноставан, није увек најпрактичнији. Код већих база података корисници често немају потребу да гледају целу табелу нити да раде са свим пољима истовремено. Уместо тога, много је практичније да се подаци уносе преко посебно припремљеног екрана који приказује само потребне информације.

Управо из овог разлога у базама података користе се обрасци. Обрасци омогућавају унос, преглед и ажурирање података на прегледнији и контролисанији начин. Уместо да ради са редовима и колонама, корисник користи поља, дугмад, листе и друге елементе који подсећају на електронски образац.

У свакодневном животу често се срећу различити обрасци. Попуњавање онлајн пријаве, регистрација корисничког налога, унос података за поруџбину или електронско пријављивање испита представљају примере образаца који у позадини комуницирају са базом података.

Од табеле до обрасца

Разгледати табелу за ученике.

УченикID	Име	Презиме	Телефон	Е-пошта
1	Ана	Петрова	071123456	ana@email.com
2	Марко	Стојанов	075654321	marko@email.com

На први поглед табела садржи све потребне информације. Међутим, ако корисник треба свакодневно да уноси податке о новим ученицима, овакав приказ није увек најпрактичнији, посебно ако табела садржи десетак или двадесет поља.

Због тога се може креирати образац који ће податке приказивати на једноставнији начин.

Предности коришћења образаца

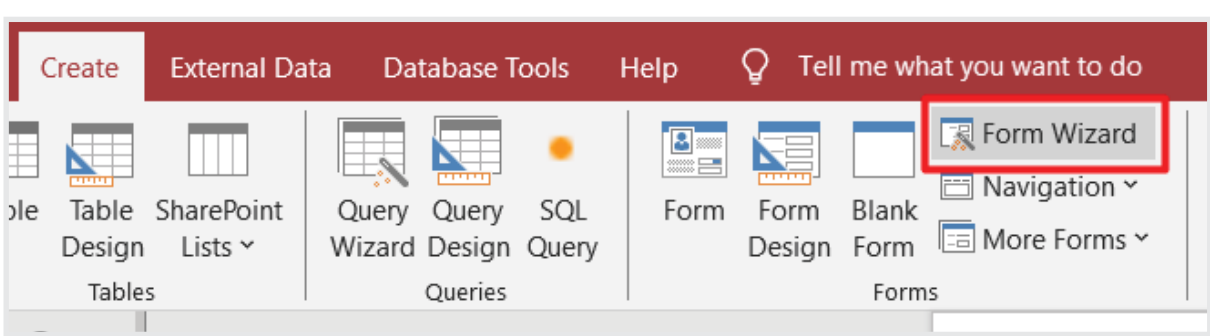
Обрасци знатно побољшавају комуникацију између корисника и базе података. Они омогућавају једноставнији унос информација и смањују могућност појаве грешака. Уместо да се корисник креће кроз велику табелу, његова пажња усмерена је само на податке које треба унети.

Додатна предност је могућност боље организације елемената. Повезане информације могу да се групишу, а могу се користити и дугмад за избор, падајуће листе или поља за потврду. На овај начин образац постаје прегледнији и лакши за коришћење.

Обрасци омогућавају и ограничавање приступа одређеним информацијама. Корисник може да ради са обрасцем без директног отварања табеле. Ово доприноси већој безбедности и бољој контроли над подацима.

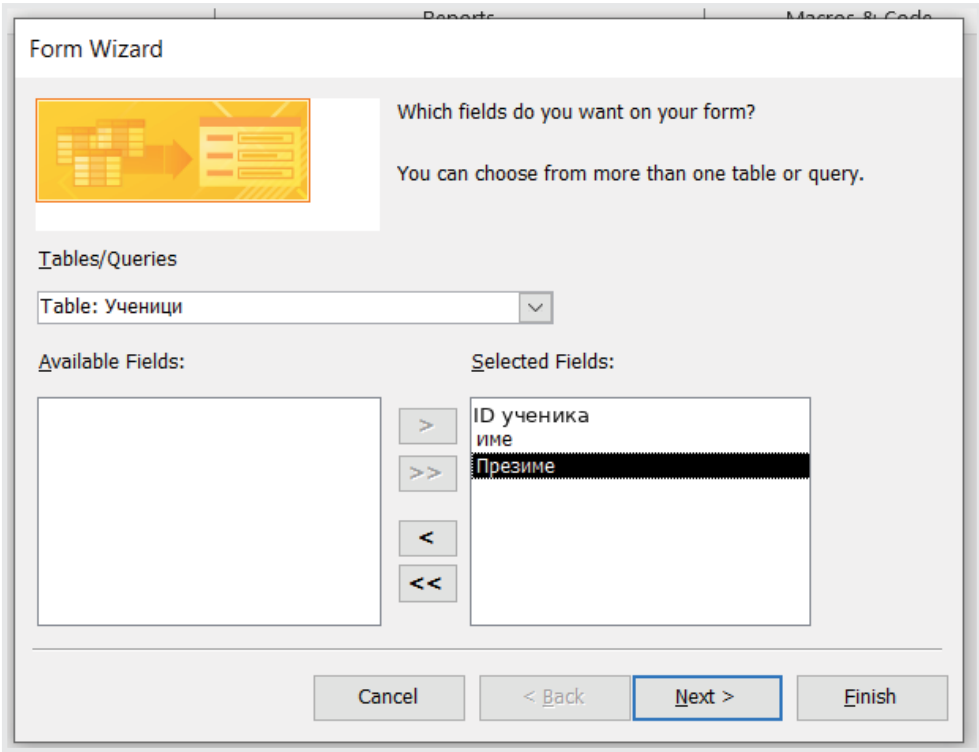
Креирање обрасца

У програмима Microsoft Access и LibreOffice Base постоје алати који омогућавају аутоматско креирање образаца на основу постојеће табеле. Корисник најпре бира табелу која ће бити извор података, а затим програм аутоматски поставља поља у образац.



Слика 1 Укључивање обрасца

Након што се образац креира, његов изглед може се додатно уређивати. Могу се променити наслови, величина елемената, њихов положај и начин приказивања података. На овај начин образац се може прилагодити конкретним потребама корисника.



Слика 2 Избор табеле и полиња од табела

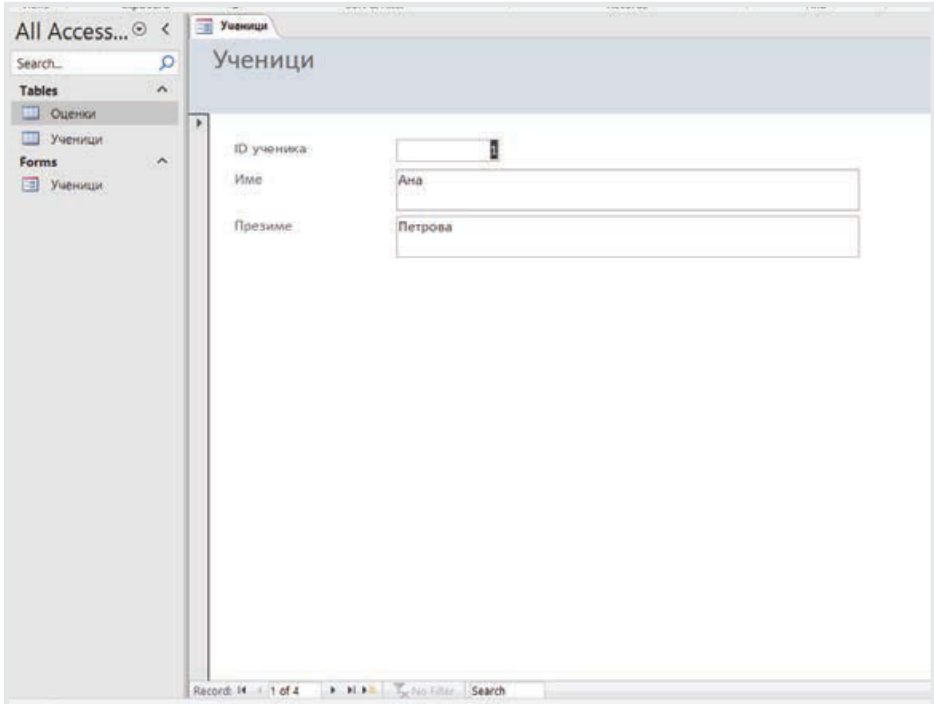
Сва потребна поља додају се у образац из одговарајуће табеле. Ако у бази постоји више табела, најпре је потребно изабрати табелу, а затим и поља која треба да се појаве у обрасцу.

Уношење података преко обрасца

Након што се образац креира, може се користити за унос нових података у базу података. Корисник више нема потребу да директно отвара табелу и тражи одговарајући ред за унос информација. Уместо тога, потребно је само да попуни поља приказана на обрасцу.

При уносу новог записа свака информација уноси се у одговарајуће поље. Ако је за неко поље подешена валидација или постављено ограничење, програм ће аутоматски проверити да ли унета вредност испуњава задате услове. На тај начин смањује се могућност уноса нетачних или непотпуних података.

Обрасци омогућавају да процес уноса информација више личи на попуњавање електронског обрасца него на рад са класичном табелом. Ово је посебно корисно када базу података користе особе које немају напредна информатичка знања.



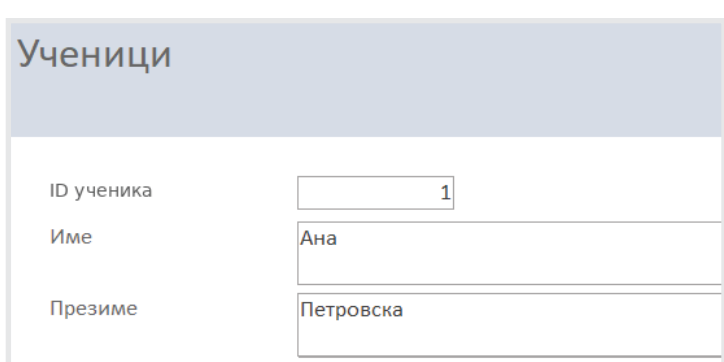
Слика 3 Образац у MS Access

Након креирања обрасца помоћу Чаробњака за обрасце приказују се сви записи, уколико их има. Из претходног примера може се уочити да корисник више не види редове и колоне. Уместо тога, сваки податак уноси се у посебно поље. Овакав начин рада сличнији је попуњавању електронских образаца које свакодневно користимо на интернету.

Уређивање сталних записа

Подаци унети у базу података не остају увек непромењени. Бројеви телефона се мењају, корисници добијају нове адресе електронске поште, а одређене информације треба редовно ажурирати. Обрасци омогућавају да се овакве промене изврше једноставно и брзо.

Када корисник отвори постојећи запис помоћу обрасца, све информације приказују се у пољима која се могу директно изменити. Након измене, нови подаци аутоматски се уписују у базу података. На овај начин корисник не мора да тражи одговарајућу информацију у табели.



Слика 4 Промена постојећег записа

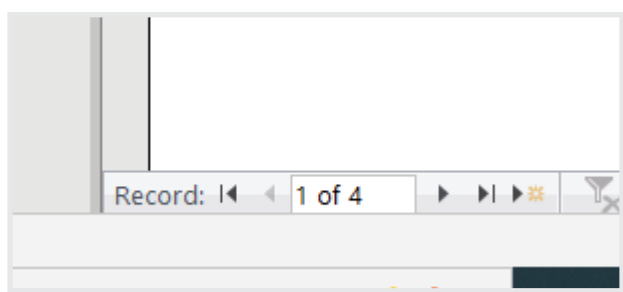
Уређивање помоћу обрасца омогућава већу прегледност и бољу контролу над информацијама. Корисник се усредсређује само на запис који обрађује, а да притом није оптерећен осталим подацима који се налазе у табели.

Навигација кроз записе

Образац обично не приказује само један запис. Он омогућава кретање кроз све записе који се налазе у табели. У ту сврху користе се навигациона дугмад помоћу којих корисник може да пређе на претходни, следећи, први или последњи запис.

Ова могућност је посебно корисна када табела садржи велики број записа. Уместо да корисник ручно претражује податке, довољно је да користи навигационе елементе обрасца. На тај начин прегледање информација постаје брже и једноставније.

У многим савременим информационим системима обрасци садрже и додатне могућности за претраживање и филтрирање података. Иако ће ове функције бити детаљније обрађене у наредним лекцијама, важно је разумети да обрасци не служе само за унос информација него и за њихово ефикасно коришћење.



Слика 5 Навигација кроз записе

рактичан пример: Евиденција ученика

Претпоставимо да је потребно израдити једноставан систем за евиденцију ученика у једној школи. У бази података већ постоји табела која садржи потребне информације о њима. Уместо да наставник или административни радник ради директно са табелом, може се креирати образац који ће омогућити лакши унос и преглед података.

Образац може да садржи поља за име, презиме, број телефона, адресу електронске поште и одељење. При уносу података о новом ученику корисник попуњава поља, а систем аутоматски чува запис у бази података. Ако је касније потребно променити неки податак, корисник једноставно отвара одговарајући запис помоћу обрасца и уноси потребну измену.

Овакав приступ знатно побољшава ефикасност рада. Уместо сложене табеле са много редова и колона, корисник ради са једноставним и прегледним екраном који приказује само информације које су му потребне. Због тога обрасци представљају један од најчешће коришћених елемената у савременим базама података.

Анализа примера

Из претходног примера може се уочити да обрасци служе као посредници између корисника и базе података. Они не замењују табеле, већ омогућавају лакши рад са њима. Сви подаци се и даље чувају у табелама, али корисник најчешће комуницира преко образаца.

Коришћењем образаца побољшава се прегледност, смањује се могућност појаве грешака и олакшавају се унос и ажурирање података. Због ових предности обрасци имају важну улогу у развоју информационих система заснованих на базама података.

ЗАКЉУЧАК:

Обрасци представљају важан елемент у базама података и омогућавају једноставнију комуникацију између корисника и података. Уместо да директно ради са табелама, корисник користи прегледан интерфејс преко којег може да уноси, прегледа и ажурира информације.

Коришћењем образаца смањује се могућност појаве грешака при уносу података, а истовремено се побољшавају прегледност и ефикасност рада. Из ових разлога обрасци представљају један од најчешће коришћених елемената у савременим базама података.

ДА ЛИ ЗНАШ?

Да ли знаш да свака веб-страница за регистрацију, пријављивање или онлајн наручивање заправо користи образац? Иако корисник види само поља за унос података, те информације се у позадини уписују у базу података.

ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представља образац у бази података?
2. Које су предности образаца у односу на директан рад са табелама?
3. За шта се обрасци најчешће користе?
4. Како се уносе подаци преко обрасца?
5. Како се врши ажурирање постојећих записа?
6. Шта представља навигација кроз записе?
7. Да ли обрасци замењују табеле?

ПРАКТИЧНИ ЗАДАЦИ

Задатак 1
Отвори постојећу табелу и креирај образац који ће приказивати сва поља!

Задатак 2
Отвори постојећу табелу и креирај образац који ће приказивати сва поља!

Задатак 3
Промени два постојећа записа користећи образац!

Задатак 4
Прегледај све записе користећи навигацију!

Задатак 5
Предложи три поједностављивања образаца због прегледности!

ИСТРАЖИВАЧКИ ЗАДАТАК

Истражи примере електронских образаца који се свакодневно користе на интернету. Пронађи најмање три примера и анализирај:

- која је њихова намена;
- који подаци се уносе;
- како су организована поља;
- која правила за унос се примењују.

ИСТРАЖИ

- Образац представља кориснички интерфејс за рад са подацима.
- Обрасци омогућавају унос, преглед и ажурирање записа.
- Подаци се и даље чувају у табелама.
- Обрасци побољшавају прегледност и олакшавају рад.
- Навигацијски елементи омогућавају кретање кроз записе.
- Обрасци су широко коришћени у савременим информатичким системима.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Поред једноставних образаца, савремени системи омогућавају креирање сложених образаца који садрже дугмад, листе за избор, слике, аутоматске прорачуне и повезане податке из више табела. У професионалним апликацијама обрасци су често главни начин комуникације између корисника и базе података.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

Заокружи тачан одговор

1. Образац служи за:
а) брисање база података
б) уношење и прегледање података
в) форматирање оперативног система
г) компресирање датотека
2. Подаци унети преко обрасца чувају се у:
а) оперативна меморија
б) образац
в) табела
г) извештај

Тачно или нетачно

1. Обрасци омогућавају лакши рад са подацима.
2. Подаци се чувају у самом обрасцу.
3. Обрасци могу да се користе за ажурирање записа.
4. Навигацијске команде омогућавају кретање кроз записе.

АНАЛИЗА

У једној школској бази података постоји табела са 500 ученика.

Објасни:

- зашто је коришћење обрасца практичније од директног рада са табелом;
- које предности добија корисник;
- како образац може да помогне при уносу новог ученика.

3.6

УПИТИ (QUERIES) – ПРЕТРАЖИВАЊЕ И СЕЛЕКТИРАЊЕ ПОДАТАКА

ВЕЋ ЗНАШ ДА...

- креираш табеле у бази података;
- користиш разне типове података;
- примењујеш примарни кључ;
- уносиш податке преко образаца;
- прегледаш и ажурираш записе.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Шта ако у бази има хиљаду записа, а треба ти само један?
2. Како могу да се пронађу сви ученици једног одељења?
3. Како могу да се прикажу само ученици са одличним успехом?
4. Да ли је потребно да се увек прегледају сви записи?
5. Како може компјутер брзо да пронађе потребне информације?

- **У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:**
- • објашњаваш појам и улогу упита у бази података;
- • креираш упите за издвајање потребних података;
- • постављаш један или више критеријума у упиту;
- • сортираш и анализираш резултате упита;
- • примењујеш упите у практичним ситуацијама.

Увод

У стварним базама података често се чувају стотине, хиљаде или чак милиони записа. Иако су сви подаци важни, у пракси корисницима најчешће треба само одређени део информација. На пример, наставник може желети да види само ученике из одређеног одељења, библиотекар само књиге које су тренутно позајмљене, а администрација само ученике који нису измирили трошкове екскурзије.

Прегледање целе табеле сваки пут када је потребна одређена информација било би споро и неефикасно. Због тога системи за управљање базама података омогућавају коришћење посебних механизама за претраживање и издвајање информација. Ови механизми називају се упитима.

Упити представљају алат који омогућава да се из базе података издвоје само информације које испуњавају задате услове. Уместо да корисник ручно претражује све записе, систем аутоматски проналази и приказује само потребне резултате.

Шта представља упит?

Упит представља захтев који се упућује систему за управљање базама података. Помоћу упита корисник одређује које податке жели да добије и под којим условима треба да буду изабрани.

За разлику од табеле, која садржи све податке, упит најчешће приказује само један њихов део. На пример, могу се приказати само ученици из одређеног одељења, само ученици са просеком већим од 4,50 или само ученици који учествују на такмичењима.

Важно је нагласити да упит не мења податке у табели. Он само приказује информације које испуњавају задате услове. Након затварања упита подаци у табели остају непромењени.

Практичан пример

Разгледати следећу табелу

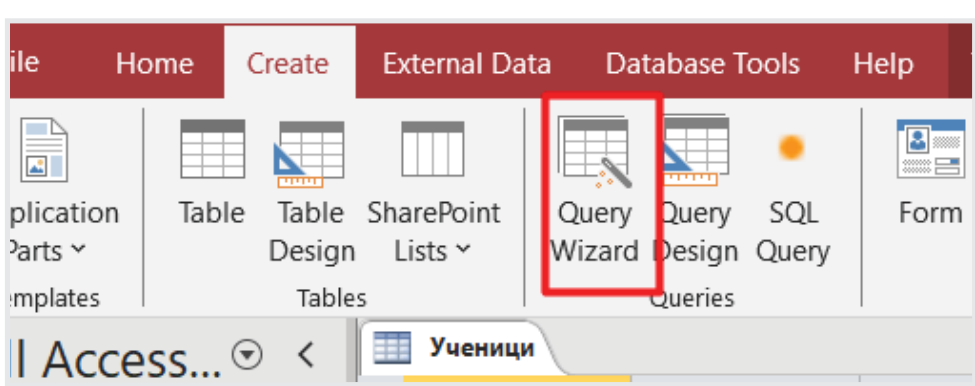
Име	Презиме	Паралелка	Просек
Ана	Петрова	I-1	5.00
Марко	Стојанов	I-2	4.20
Елена	Николовска	I-1	4.80
Merita	Rexhepi	I-3	3.90

Претпоставимо да је потребно приказати само ученике из одељења I-1.
Уместо да се табела ручно претражује, може се креирати упит којим ће се задати услов.

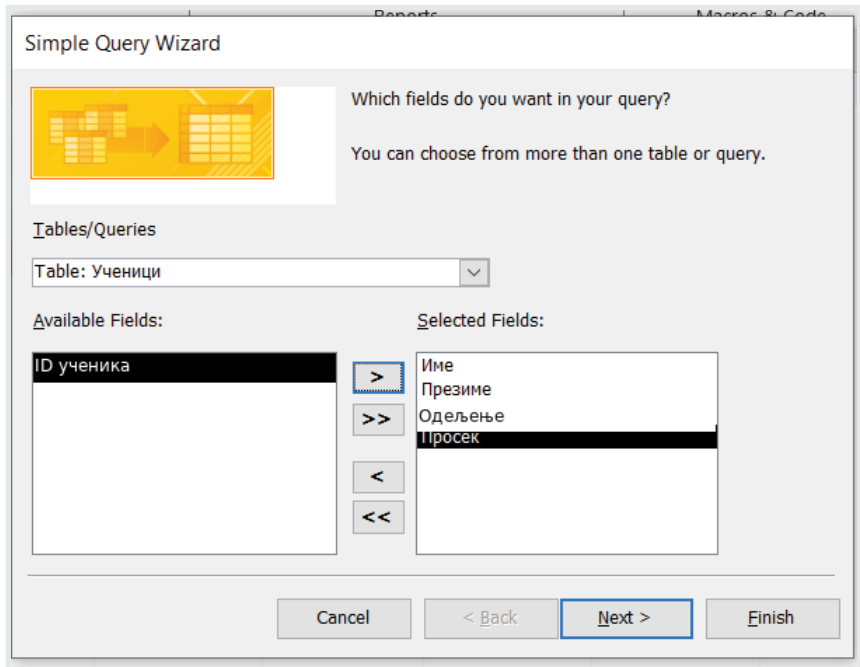
Одељење = I-1

Као резултат добиће се:

Име	Презиме	Паралелка	Просек
Ана	Петрова	I-1	5.00
Елена	Николовска	I-1	4.80

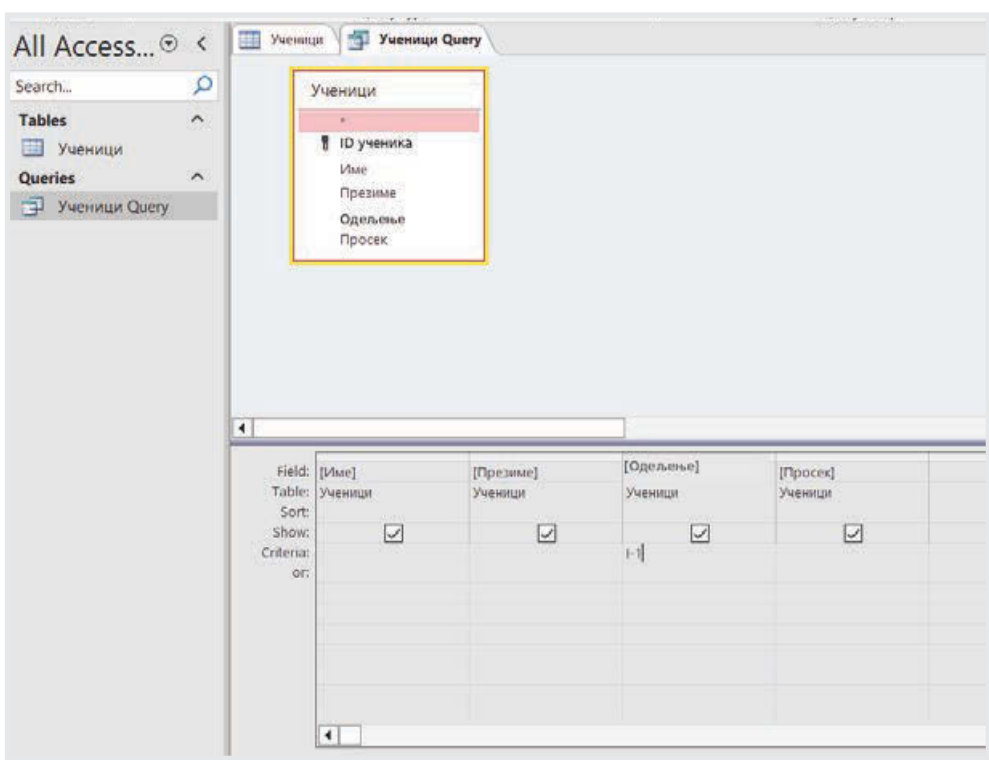


Слика 1 Упит у MS Access-у



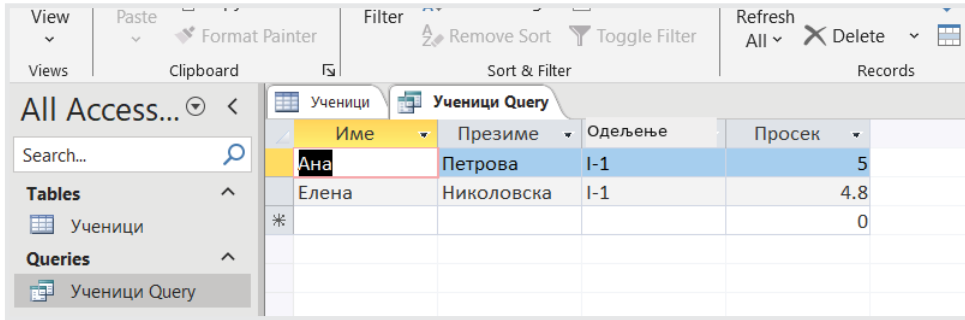
Слика 2 Постављање параметара у упиту

У већ креираном извештају који садржи све елементе, потребни критеријуми могу се подесити у приказу дизајна.



Слика 3 Постављање критеријума у упиту

У овом случају реч је о ученицима из одеље I-1, па се та вредност уписује у поље за критеријум у колони „Клас“. Након извршавања упита избором „Run“, приказују се само ученици из задатог одељења.



Слика 4 Ефекат постављеног критеријума у упиту

Овако припремљен упит приказује тражене податке из базе. У једној бази може се креирати неограничен број упитника за различите карактеристике базе.

ашто се користе упитници?

Упити знатно олакшавају коришћење база података. Уместо да корисник ради са великим бројем записа, систем аутоматски издваја потребне информације. Ово не само што штеди време него и смањује могућност појаве грешака.

Поред тога, упити омогућавају да се подаци анализирају на различите начине. Иста табела може се користити за добијање различитих информација у зависности од постављених услова. Тако се једном могу претраживати ученици према одељењу, други пут према успеху, а трећи пут према броју изостанака.

Због ових могућности упити су један од најкориснијих алата за рад са базама података и имају широку примену у различитим информационим системима.

Услови у упитима

Једна од највећих предности упита јесте могућност постављања услова према којима ће се подаци бирати. Уместо да се приказују сви записи из табеле, корисник може да одреди које информације жели да добије. Ово омогућава брзо проналажење потребних података чак и када база садржи велики број записа.

Услов представља правило које запис мора да испуни како би био приказан у резултатима. На пример, могу се затражити сви ученици из одређеног одељења, све књиге одређеног аутора или сви производи чија је количина мања од задате вредности. Систем аутоматски проверава записе и приказује само оне који испуњавају услов.

У пракси се могу користити различите врсте услова. Неки услови проверавају да ли је вредност једнака одређеној вредности, док други проверавају да ли је вредност већа, мања или се налази у одређеном опсегу. Захваљујући овој флексибилности, упити се могу прилагодити различитим потребама корисника.

Пример за коришћење услова

Име	Презиме	Просек
Ана	Петрова	5.00
Марко	Стојанов	4.20
Елена	Николовска	4.80
Merita	Rexhepi	3.90

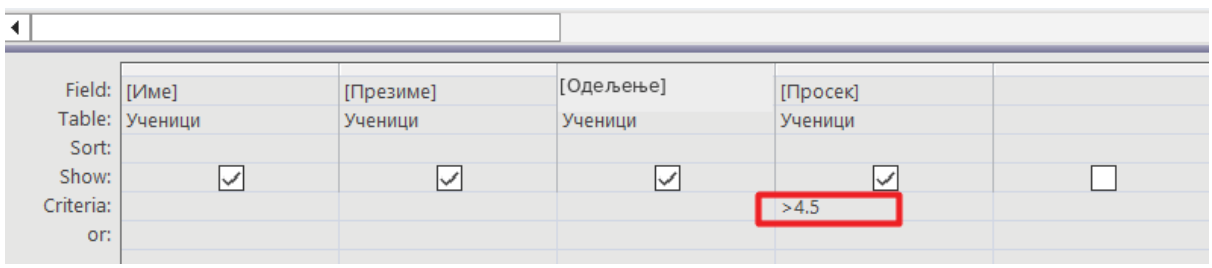
Претпоставимо да је потребно приказати само ученике са просеком већим од 4,50. Услов ће бити:

Просек > 4.50

Резултат ће садржати следеће записе:

Име	Презиме	Просек
Ана	Петрова	5.00
Елена	Николовска	4.80

Из примера се може уочити да је систем аутоматски издвојио само записе који испуњавају постављени услов.



Слика 5 Постављање услова за просек

Сортирање резултата

Поред претраживања података, упити омогућавају и њихово сортирање. Сортирање представља организовање резултата према одређеном редоследу. Најчешће се користи ради лакшег прегледа и анализе информација.

Подаци се могу сортирати по азбучном реду, бројчаним вредностима или датумима. На пример, списак ученика може се сортирати према презимену, списак производа према цени, а списак књига према години издања.

Сортирање не мења податке у табели. Оно утиче само на начин приказивања резултата у упиту. На тај начин корисник може брзо да добије јаснији преглед информација.

Пример за сортирање

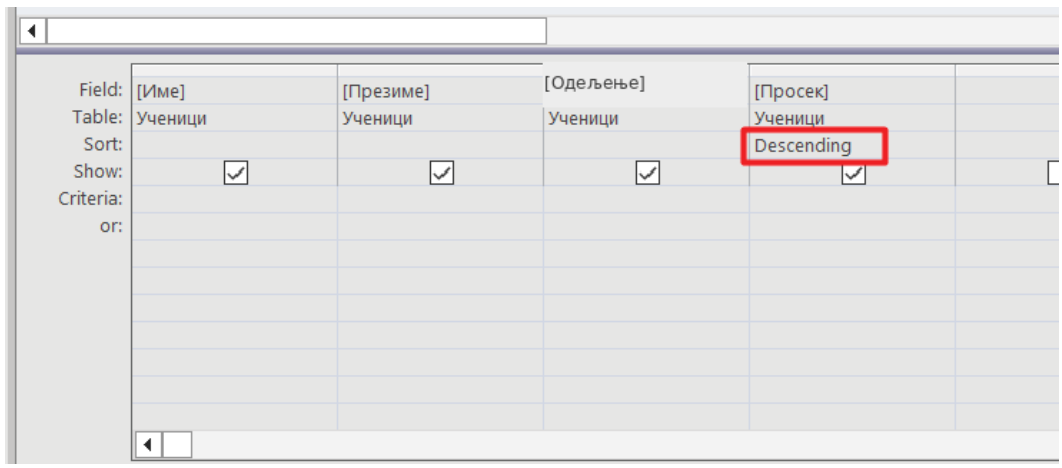
Разгледати следећи резултат.

Име	Просек
Марко	4.20
Ана	5.00
Merita	3.90
Елена	4.80

Ако се примени сортирање према просеку опадајућим редоследом, резултат ће бити:

Име	Просек
Ана	5.00
Елена	4.80
Марко	4.20
Merita	3.90

На овај начин највише вредности се приказују на почетку списка.



Слика 6 Сортирање резултата

После сваке промене која се направи у дизајнерском приказу упита потребно је активирати команду „Run" да би се добило ново стање упита.

Комбиновање више услова

У многим ситуацијама један услов није довољан за добијање потребних информација. Због тога упити омогућавају коришћење више услова истовремено. На пример, могу се тражити ученици који припадају одређеном одељењу и истовремено имају просек већи од одређене вредности.

Претпоставимо да треба пронаћи ученике из одељења I-1 са просеком већим од 4,50.

Потребни се два услови:

- Одељење = I-1
- Просек > 4.50

Само записи који испуњавају оба услова биће приказани у резултатима.

Комбиновање услова омогућава много прецизније претраживање информација. Што су услови прецизније постављени, то ће добијени резултати бити релевантнији.

Практичан пример: Школска база података

Претпоставимо да школа води евиденцију о свим ученицима првог разреда. Директор жели да добије списак ученика који имају просек већи од 4,50 и истовремено немају више од три неоправдана изостанка.

Уместо да се ручно прегледају стотине записа, може се креирати упит који ће аутоматски издвојити само ученике који испуњавају оба услова. Резултат ће бити добијен за неколико секунди, без потребе за додатном обрадом.

Овакве ситуације свакодневно се јављају у образовању, здравству, банкарству и администрацији. Захваљујући упитницима, корисници могу брзо да добију тачне информације из великих количина података. Због тога су упити један од најмоћнијих механизама за рад са базама података.

Анализа примера

Из претходних примера може се закључити да упити не служе само за проналажење података већ и за њихову анализу. Они омогућавају издвајање потребних информација, њихово сортирање и комбиновање према различитим критеријумима. На тај начин корисник добија само релевантне податке, уместо да ради са целокупним садржајем табеле.

Захваљујући могућности постављања услова и сортирања резултата, упити значајно побољшавају ефикасност рада са базама података. Они представљају незаменљиву алатку у савременим информационим системима и свакодневно се користе за претраживање и анализу информација.

Screenshot 5 – Упит са више услова

Screenshot 6 – Резултати упита

ЗАКЉУЧАК:

Упити представљају важан алат за рад са базама података јер омогућавају брзо проналажење, издвајање и анализу информација. Уместо да корисник ручно претражује све записе у табели, систем аутоматски издваја податке који испуњавају задате услове.

Коришћењем упита могу се применити различити критеријуми претраживања, комбиновати више услова и сортирати резултати према потребама корисника. На тај начин побољшава се ефикасност рада и олакшава доношење одлука на основу тачних информација.

Због своје практичне примене, упити представљају један од најважнијих механизама у савременим базама података и користе се у готово свим информационим системима.

ДА ЛИ ЗНАШ?

Да ли знаш да интернет претраживачи користе принцип сличан упитима? Када унесеш појам за претрагу, систем аутоматски претражује огромну количину података и приказује само резултате који одговарају постављеном захтеву.

ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представља упит у бази података?
2. Која је разлика између табеле и упита?
3. Шта представља услов у упиту?
4. Зашто се користи сортирање резултата?
5. Да ли упит мења податке у табели?
6. Зашто је корисно комбиновање више услова?
7. У којим ситуацијама се најчешће користе упити?

ПРАКТИЧНИ ЗАДАЦИ

Задатак 1

Креирај упит који ће приказивати све ученике изабраног одељења!

Задатак 2

Креирај упит који ће приказивати ученике са просеком већим од 4,50!

Задатак 3

Сортирај резултате из претходног упита према просеку опадајућим редоследом!

Задатак 4

Креирај упит који ће приказивати ученике из одељења I-1 са просеком већим од 4,50!

Задатак 5

Направи упит који ће приказивати само ученике који имају више од пет изостанака!

ИСТРАЖИВАЧКИ ЗАДАТАК

У школи постоје подаци о ученицима, наставницима, предметима и оценама.

Предложи најмање пет питања на која би директор могао да добије одговор коришћењем упита.

За свако питање наведи:
- који подаци су потребни;
- који услови треба да се поставе;
- какав резултат се очекује.

ЗАПАМТИ ШТА СИ НАУЧИО

- Упит представља алатку за претраживање и селектирање података.
- Упити омогућавају приказивање само потребних информација.
- Услови одређују који записи ће бити приказани.
- Могуће је коришћење више услова истовремено.
- Резултати могу да се сортирају према различитим критеријумима.
- Упити не мењају податке у табели.
- Упити значајно побољшавају ефикасност рада са базама података.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У савременим системима за управљање базама података постоје напредни упити који могу да врше статистичке прорачуне, груписање података и анализу информација из више повезаних табела. У великим информационим системима ови упити се често користе за израду извештаја, анализу продаје, праћење финансијских података и подршку при доношењу пословних одлука.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

Заокружи тачан одговор

1. Упит служи за:

а) форматирање табела

б) претраживање и селектирање података

в) инсталацију програма

г) креирање оперативног система

2. Услов у упиту:

а) брише записе

б) одређује који записи ће бити приказани

в) мења структуру табеле

г) затвара базу података

3. Сортирање се користи за:

а) сортирање резултата

б) брисање резултата

в) креирање образаца

г) додавање нових поља

Тачно или нетачно

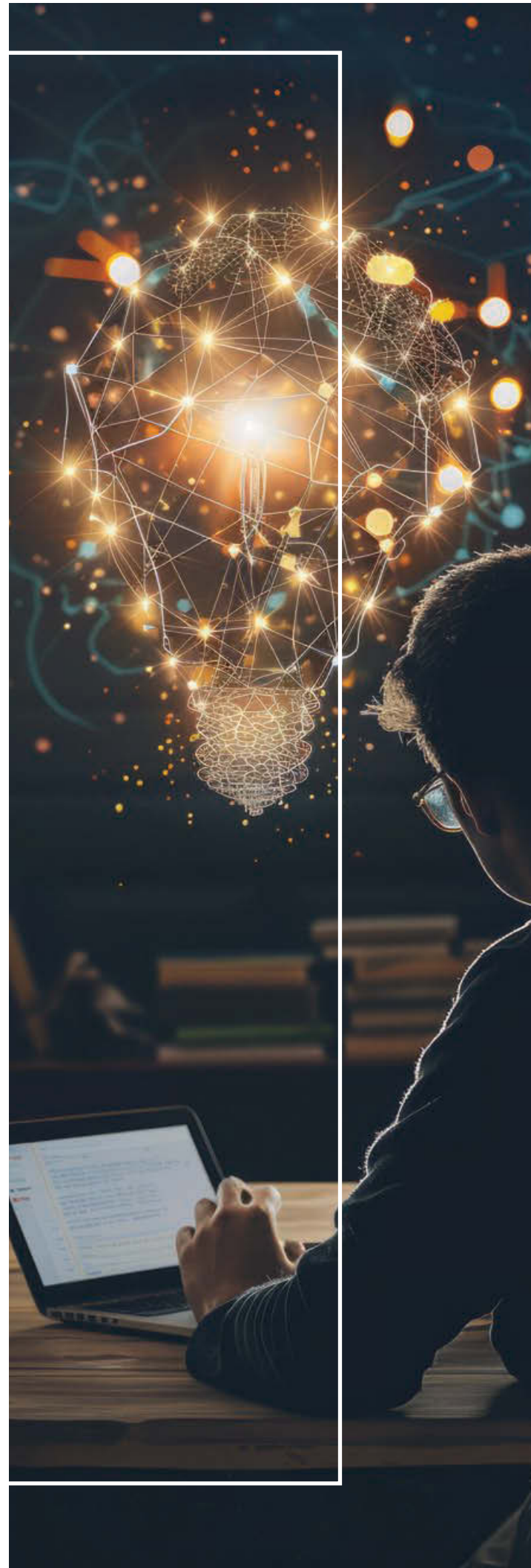
1. Упити могу да користе више услова истовремено.
2. Резултати упита могу да се сортирају.
3. Упит увек приказује све записе из табеле.
4. Упит мења податке у табели.

АНАЛИЗА

У бази података за библиотеке чувају се информације о књигама, ауторима и корисницима.

Објасни:

- какав упит би креирао да прикажеш све књиге одређеног аутора;
- какав услов би поставио;
- како би сортирао резултате да би били прегледнији.



3.7

ИЗВЕШТАЈИ (REPORTS) – ПРИКАЗИВАЊЕ И ШТАМПАЊЕ ПОДАТАКА

ВЕЋ ЗНАШ ДА...

- креираш табеле у бази података;
- користиш разне типове података;
- уносиш податке преко обрасца;
- креираш упите за претраживање информација;
- користиш услове и сортирање резултата;
- анализираш податке добијене из базе података.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Како могу да се припреме за штампу подаци из базе података?
2. Да ли је табела увек најбољи начин за приказивање информација?
3. Како може да се добије списак ученика спреман за штампање?
4. Како институције израђују извештаје о свом раду?
5. На који начин резултати упита могу да се представе прегледније?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш шта представља извештај;
- наводиш предности коришћења извештаја;
- креираш једноставан извештај;
- користиш податке из табела и упита у извештајима;
- уређујеш изглед извештаја;
- анализираш примену извештаја у информатичким системима.

При раду са подацима, они се организују у табеле, могу се уносити преко образаца и претраживати помоћу упита. Међутим, у многим ситуацијама потребно је да добијене информације буду приказане на прегледнији начин или припремљене за штампање. На пример, наставник можда жели да одштампа списак ученика, библиотекар извештај о задуженим књигама, а директор преглед успеха по одељењима.

Иако табеле и упити садрже потребне податке, њихов изглед није увек погодан за представљање или штампање. Због тога системи за управљање базама података омогућавају коришћење извештаја. Извештаји служе за организовано и професионално приказивање информација.

Извештај представља посебан објекат у бази података који приказује податке у облику припремљеном за прегледање, штампање или дељење. Он може да садржи наслове, поднаслове, груписане информације, бројеве страница и друге елементе који побољшавају прегледност.

Шта представља извештај?

Извештај је организован приказ података који се налазе у бази података. За разлику од образаца, који се најчешће користе за унос и уређивање информација, извештаји су намењени приказивању и представљању података.

Извештаји могу да користе информације директно из табела или из претходно креираних упита. На тај начин корисник може да прикаже само одређени део података који су потребни за конкретну анализу или презентацију.

У пракси се извештаји често користе за израду спискова, статистичких прегледа, резултата анализа, финансијских извештаја и других докумената који треба да буду јасно организовани и лаки за читање.

Практичан пример

Претпоставимо да школа жели да припреми списак ученика одељења I-1.

Подаци се најпре чувају у табели, а затим се помоћу упита издвајају само ученици из изабраног одељења.

Добијени резултати могу да изгледају овако:

Име	Презиме	Паралелка
Ана	Петрова	I-1
Елена	Николовска	I-1
Merita	Rexhepi	I-1

Иако су информације тачне, овакав приказ није најпогоднији за штампање и архивирање. Због тога се креира извештај који ће садржRexhepiслов, датум израде и уредно сортиране податке.

Ученици

ID ученика	Име	Презиме	Одељење	Просек	Click to Add
1	Ана	Петрова	I-1	5	
2	Марко	Стојанов	I-2	4.2	
3	Елена	Николовска	I-1	4.8	
4	Merita	Rexepi	I-1	5	
(New)				0	

Слика 1 Изворни подаци о извештају

Предности извештаја

Извештаји омогућавају да информације буду приказане на прегледнији и професионалнији начин. Они могу да садрже различите елементе који побољшавају читљивост, као што су наслови, логотипи, бројеви страница и груписани подаци.

Додатна предност је могућност штампања информација. Уместо да корисник штампа табеле или резултате упита, може да изради извештај који је посебно припремљен за ту намену.

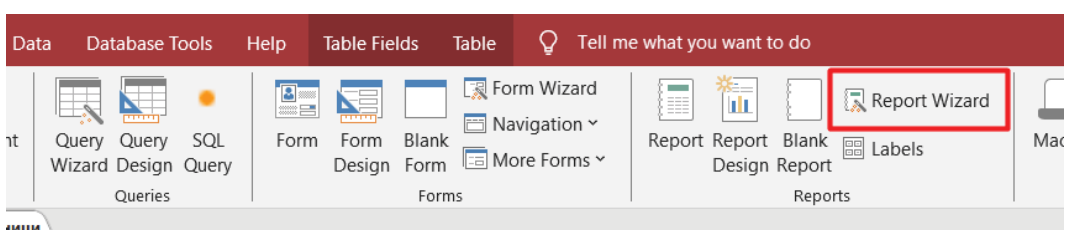
Извештаји омогућавају и боље представљање информација. Подаци могу бити организовани према различитим критеријумима, а корисник може лако да пронађе потребне информације.

Креирање извештаја

У програмима Microsoft Access и LibreOffice Base постоје алати који омогућавају аутоматско креирање извештаја. Корисник најпре бира табелу или упит као извор података, а затим програм генерише основни извештај.

Након креирања, извештај се може додатно уређивати. Могуће је променити распоред података, додати наслове и изабрати начин приказивања информација. На тај начин извештај се прилагођава конкретним потребама корисника.

У образовним установама извештаји се често користе за израду спискова ученика, извештаја о успеху, статистичких прегледа изостанака и других података који се редовно користе у раду.



Слика 2 Креирање извештаја помоћу Report Wizard

Уређивање изгледа извештаја

Након креирања извештаја, његов изглед се може додатно побољшати. Корисник може да промени распоред поља, дода наслове и поднаслове, постави бројеве страница и изабере одговарајући распоред за штампање.

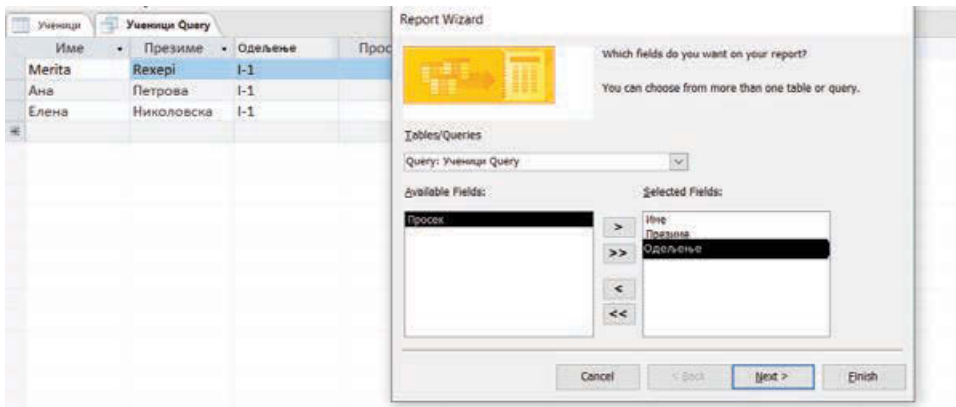
Добро организован извештај омогућава лакше читање и анализирање података. Правилно уређивање посебно је значајно за употребљивост извештаја који садрже велики број записа.

У многим случајевима извештаји садрже и додатне информације, као што су датум израде, назив установе или кратак опис садржаја. Ови елементи доприносе бољој организацији и професионалном изгледу документа.

Практичан пример: Извештај о успеху ученика

Претпоставимо да је потребно израдити извештај о ученицима који имају просек већи од 4,50.

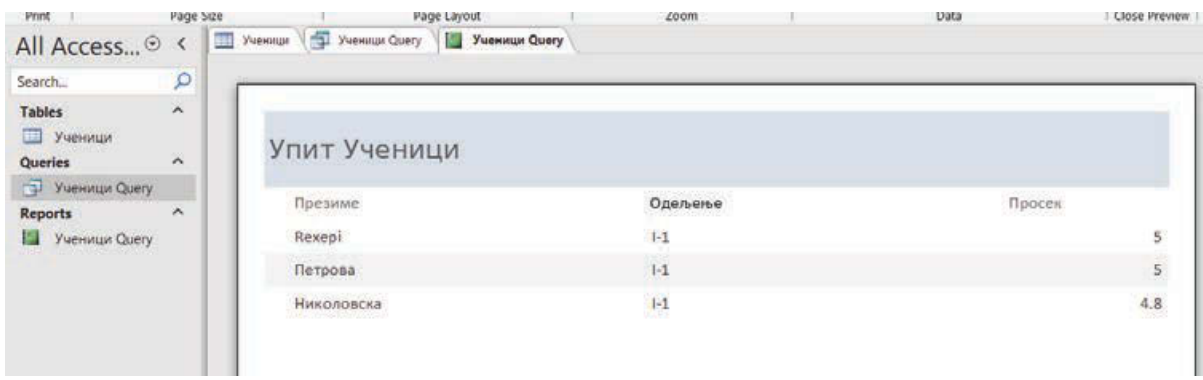
Најпре се креира упит који издваја ученике који испуњавају услов. Затим се добијени резултати користе као основа за извештај. Прво се поставља упит у којем је критеријум да просек буде већи од 4,50. Такав упит се затим бира приликом израде извештаја.



Слика 2 Припрема извештаја

У извештају могу бити приказани име, презиме, одељење и просечан успех сваког ученика. Додатно се могу унети наслов „Ученици са одличним успехом“ и датум израде документа.

На овај начин добијене информације постају прегледније и припремљене за штампање или представљање наставничком већу, одељењским старешинама или руководству школе.



Слика 3 Извештај припремљен за штампање

Анализа на примерот

Из претходног примера може се уочити да извештај не представља нови извор података, већ организован приказ већ постојећих информација. Подаци остају у табелама и упитима, док извештај служи за њихову презентацију. Коришћењем извештаја побољшава се читљивост информација и олакшава њихово коришћење у свакодневном раду. Уместо да корисник ради са сложеним табелама и упитима, он добија документ који је припремљен за преглед и штампање.

ЗАКЉУЧАК:

Извештаји представљају важан елемент база података и омогућавају приказивање информација на организован и прегледан начин. За разлику од табела и упита, који се користе за чување и претраживање података, извештаји су намењени њиховом представљању и штампању. Коришћењем извештаја могу се припремити различити спискови, анализе и прегледи који се користе у свакодневном раду установа и организација. Захваљујући могућности уређивања изгледа, извештаји омогућавају да информације буду приказане јасно и професионално. Због своје практичне примене, извештаји су важан део сваке базе података и често се користе за припремуд кумената које треба прегледати, архивирати или одштампати.

ДА ЛИ ЗНАШ?

Да ли знаш да се већина докумената које добијају директори, менаџери и руководиоци у различитим установама најчешће аутоматски генерише из база података? Уместо да се припремају ручно, извештаји се креирају за неколико секунди на основу података који се већ налазе у систему.

ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представља извештај у бази података?
2. Која је разлика између обрасца и извештаја?
3. Из којих извора се могу добијати подаци за извештај?
4. Зашто се користе извештаји?
5. Које су предности извештаја при штампању података?
6. Како се може побољшати изглед извештаја?

ПРАКТИЧНИ ЗАДАЦИ

Задатак 1

Креирај извештај на основу табеле са ученицима.

У извештају прикажи:

- име;
- презиме;
- одељење.

Задатак 2

Креирај упит који ће приказати ученике са просеком већим од 4,50 и на основу њега изради извештај.

Задатак 3

Додај наслов извештаја:

„Списак ученика са одличним успехом“

Задатак 4

Додај датум израде и бројеве страница у извештај.

Задатак 5

Одштампај или извези извештај у PDF формат.

ИСТРАЖИВАЧКИ ЗАДАТАК

Истражи у којим ситуацијама школе користе извештаје.

Наведи најмање пет примера и објасни:

- која је њихова намена;
- који се подаци користе;
- ко би користио те извештаје.

ЗАПАМТИ ШТА СИ НАУЧИО

- Извештај служи за организован приказ података.
- Извештаји могу да користе податке из табела и упита.
- Намењени су прегледању, штампању и презентацији информација.
- Изглед извештаја може додатно да се уређује.
- Извештаји не представљају нове податке, већ приказ постојећих информација.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

У професионалним информационим системима извештаји често садрже графиконе, статистичке анализе и аутоматске прорачуне. На пример, финансијски извештај може аутоматски да израчунава приходе и расходе, док извештај о продаји може да приказује трендове и поређења различитих периода.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

Заокружи тачан одговор

1. Извештај се најчешће користи за:
а) унос података
б) приказивање и штампање података
в) креирање оперативног система
г) брисање табела

2. Извештај може да користи податке из:
а) табеле
б) упита
в) табеле и упита
г) ниједног од наведеног

3. Извештаји се користе када:
а) податке треба организовати за презентацију
б) треба инсталирати програм
в) треба креирати оперативни систем
г) треба форматирати диск

Тачно или нетачно

1. Извештаји могу да се штампају.
2. Извештаји служе за унос нових записа.
3. Извештаји могу да садрже наслове и бројеве страница.
4. Извештаји могу да користе податке из упита.

АНАЛИЗА

У школској бази података потребно је припремити списак ученика са одличним успехом.
Објасни:

- који ће подаци бити потребни;
- да ли најпре треба креирати упит;
- зашто је извештај боље решење од обичне табеле за штампање списка.

3.8

ИЗРАДА ЦИРКУЛАРНИХ ПИСАМА СА БАЗОМ ПОДАТАКА

ВЕЋ ЗНАШ ДА...

- креираш табеле у бази података;
- уносиш и ажурираш податке;
- користиш обрасце за рад са подацима;
- креираш упите за претраживање информација;
- припремаш извештаје;
- организујеш податке у бази података.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

1. Како може исти документ да се пошаље великом броју разних лица?
2. Да ли је практично да се ручно уноси име и презиме у свакој позивници?
3. Како може школа да отштампа стотине потврда за ученике?
4. Како компаније шаљу персонализована писма својим корисницима?
5. Да ли је могуће да се неки документ аутоматски приспособљава сваком примачу?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш намену циркуларних писама;
- повезујеш документ са извором података;
- уносиш поља за спајање у шаблон документа;
- прегледаш и провераваш персонализоване записе;
- креираш већи број персонализованих докумената.

У многим ситуацијама потребно је исти документ послати великом броју прималаца. На пример, школа може да шаље позиве за родитељски састанак, потврде за ученике или захвалнице учесницима такмичења. Иако је садржај документа готово исти, одређени подаци, као што су име, презиме или адреса, разликују се за сваког примаоца.

Када би се сваки документ израђивао засебно, то би захтевало много времена и повећало могућност појаве грешака. Због тога савремени програми омогућавају аутоматско повезивање документа са подацима из базе података или табеле. На тај начин један шаблон може да се употреби за креирање великог броја персонализованих докумената.

Овај процес назива се циркуларно писмо. Он омогућава аутоматско уметање података из базе података у унапред припремљен документ. Као резултат добија се више докумената који имају исти садржај, али садрже различите личне податке.

Шта представља циркуларно писмо?

Циркуларно писмо је документ повезан са извором података. Извор података најчешће је база података, табела или листа са подацима о примаоцима. У документ се постављају посебна поља која се касније аутоматски замењују стварним подацима.

На пример, уместо да у документу буде написано конкретно име, може да се постави поље „Име“. При генерисању докумената програм ће аутоматски заменити ово поље именом сваког примаоца.

Захваљујући овом приступу, један документ може да се искористи за креирање десетина, стотина или чак хиљада персонализованих писама. То значајно повећава ефикасност рада и омогућава брзу припрему великог броја докумената.

Практичан пример

Претпоставимо да школа организује такмичење из информатике и треба да пошаље позивнице ученицима.

У бази података постоји табела:

Име	Презиме	Паралелка
Ана	Петрова	II-1
Марко	Стојанов	II-2
Елена	Николовска	II-1

У програму за обраду текста припрема се следећи шаблон:

Поштовани/а <<Име>> <<Презиме>>,

Позивамо вас да учествујете на школском такмичењу из информатике.

С поштовањем,

Организациони одбор

Након спајања документа са базом података аутоматски ће бити креиране посебне позивнице за сваког ученика.

Ученици

ID ученика	Име	Презиме	Одељење	Click to Add
1	Ана	Петрова	II-1	
2	Марко	Стојанов	II-2	
3	Елена	Николовска	II-1	
(New)				

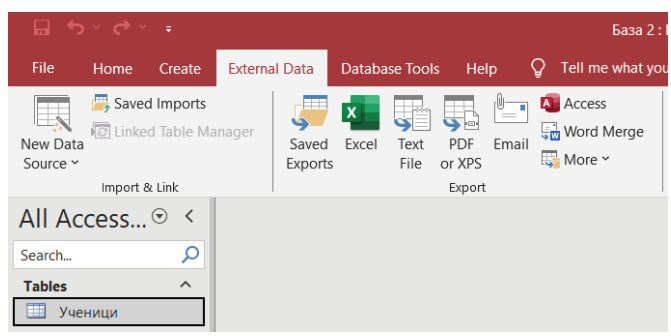
Слика 1: Табела са подацима о примаоцима

Повезивање документа са базом података

Да би се креирало циркуларно писмо, документ најпре треба повезати са изворoм података. Извор може бити база података, табела или друга структура која садржи податке о примаоцима.

Након успостављања везе, програм добија приступ свим записима који се налазе у бази. На тај начин подаци могу аутоматски да се користе при креирању докумената.

У савременим програмима овај поступак се најчешће изводи помоћу посебног, тзв. чаробњака (Wizard), који води корисника кроз процес повезивања и избора потребних поља. Пре почетка поступка табеле треба да буду затворене. Заправо, овај ексклузивни режим базе омогућава креирање осталих компоненти „пасивних" докумената. Не треба креирати писма из база у којима се у том тренутку ради. Ако је табела активна (отворена), чаробњак неће моћи да креира писмо. Потребно је само да табела буде означена (кликнута), али не и отворена. У неким верзијама софтвера потребно је најпре затворити целу базу како би могло да се креира спајање са програмом за обраду текста.



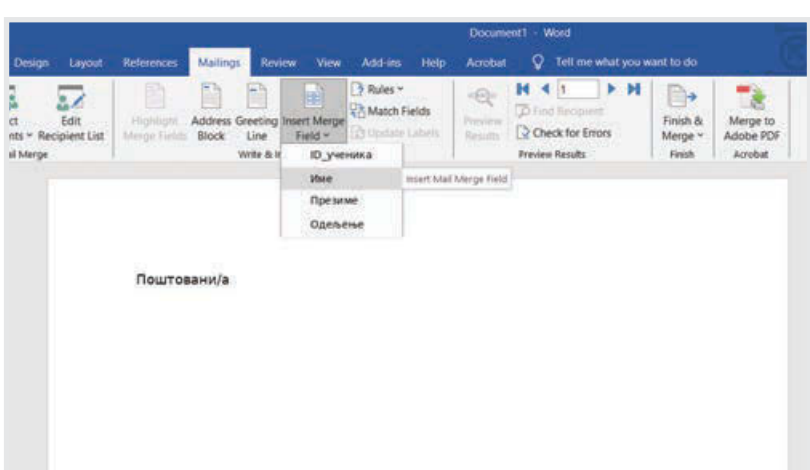
Слика 2 Повезивање са текстуалним документом

Поља за спајање

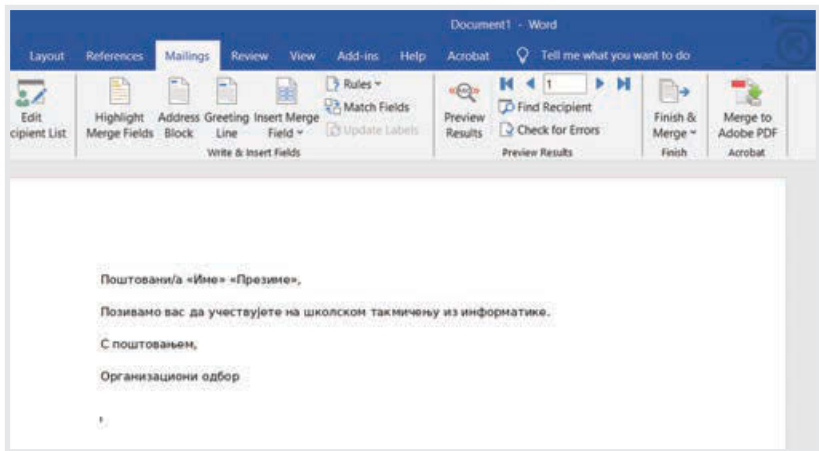
Након што документ буде повезан са базом података, потребно је одредити места на којима ће се приказивати подаци из записа. За ту сврху користе се поља за спајање.

Поља за спајање представљају ознаке које одређују место на којем ће програм уметнути податке из базе података. Свако поље је повезано са одређеном колоном табеле.

На пример, поље <<Име>> повезано је са колоном Име, а поље <<Презиме>> са колоном Презиме. При генерисању докумената програм аутоматски замењује ове ознаке одговарајућим вредностима из сваког записа.



Слика 3 Уметање Merge Fields



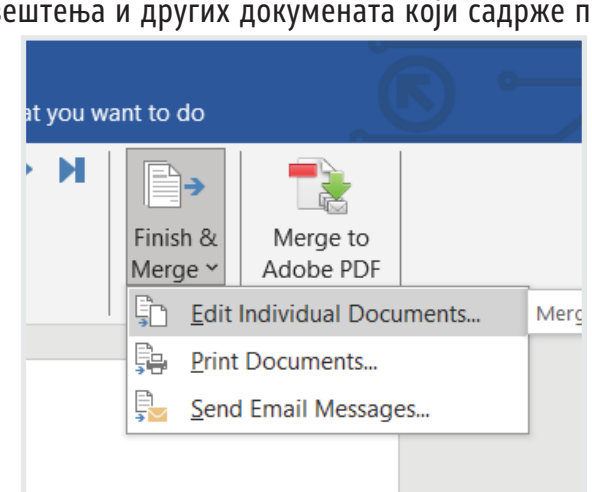
Слика 4 Комплетирано писмо

Генерисање циркуларних писама

Након постављања свих поља за спајање, програм може аутоматски да генерише документе. За сваки запис из базе података креира се посебна верзија документа.

Ако база садржи десет записа, биће креирано десет докумената. Ако садржи сто записа, биће креирано сто докумената. Корисник не мора ручно да мења имена и друге податке.

Ова могућност значајно убрзава рад и омогућава брзу припрему персонализованих позивница, потврда, сертификата, обавештења и других докумената који садрже податке о више особа.



Слика 5 Генерисање докумената

Да би се поступак завршио, потребно је извршити спајање помоћу иконе са слике 5 и применити га на све потребне записе.

Практичан пример: Сертификати за такмичење

Претпоставимо да школа организује такмичење у програмирању и да је потребно израдити сертификате за све учеснике.

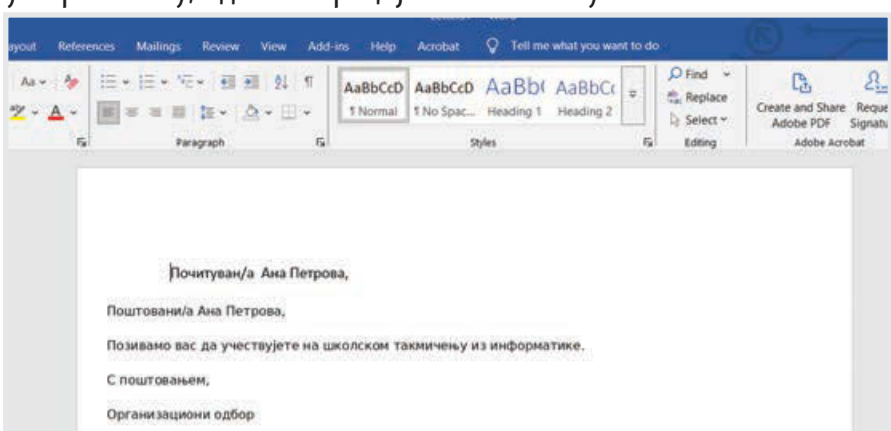
У бази података чувају се имена и презимена ученика. Уместо да се сваки сертификат уређује засебно, може се припремити један шаблон у који ће бити постављена поља за спајање.

При генерисању циркуларног писма програм ће аутоматски креирати сертификат за сваког ученика. На тај начин се за неколико секунди може припремити велики број докумената професионалног изгледа.

Анализа примера

Из претходног примера може се уочити да циркуларна писма представљају везу између база података и програма за обраду текста. Подаци остају у бази података, док документ представља шаблон који се аутоматски попуњава потребним информацијама.

Коришћењем циркуларних писама смањује се време потребно за израду докумената, умањује се могућност грешака и обезбеђује јединствен изглед свих докумената. Због тога ова техника има широку примену у образовању, администрацији и пословању.



Слика 6: Завршни персонализовани документ

ЗАКЉУЧАК:

Циркуларна писма представљају ефикасан начин за аутоматско креирање великог броја докумената који имају исти садржај, али различите податке о примаоцима. Уместо да се сваки документ уређује посебно, корисник припрема један шаблон који се повезује са базом података или табелом.

Коришћењем поља за спајање програм аутоматски умеће потребне информације у документ. Тако се штеди време, смањује могућност грешака и обезбеђује јединствен изглед свих докумената.

Циркуларна писма имају широку примену у образовању, администрацији и пословању, где је често потребно креирати велики број персонализованих докумената.

ДА ЛИ ЗНАШ?

Да ли знаш да се велики део сертификата, диплома, позивница и обавештења које данас добијају ученици и запослени креира аутоматски помоћу циркуларних писама? Уместо да се сваки документ ручно попуњава, подаци се преузимају из базе података и убацују у припремљени шаблон.

ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представља циркуларно писмо?
2. Које су предности циркуларних писама?
3. Шта представља извор података?
4. Како се документ повезује са базом података?
5. Шта су поља за спајање?
6. Зашто се користе поља за спајање?

ПРАКТИЧНИ ЗАДАЦИ

Задатак 1

Креирај табелу са следећим пољима:

- Име
- Презиме
- Одељење

Унеси податке за најмање пет ученика.

Задатак 2

Припреми шаблон позивнице за школско такмичење.

У шаблону предвиди места за:

- име;
- презиме;
- одељење.

Задатак 3

Повежи документ са табелом и уметни одговарајућа поља за спајање.

Задатак 4

Генериши циркуларно писмо за све ученике из табеле.

Задатак 5

Припреми шаблон сертификата који ће аутоматски користити податке из базе података.

ИСТРАЖИВАЧКИ ЗАДАТАК

Истражи у којим областима се циркуларна писма најчешће користе.

За сваку област наведи:

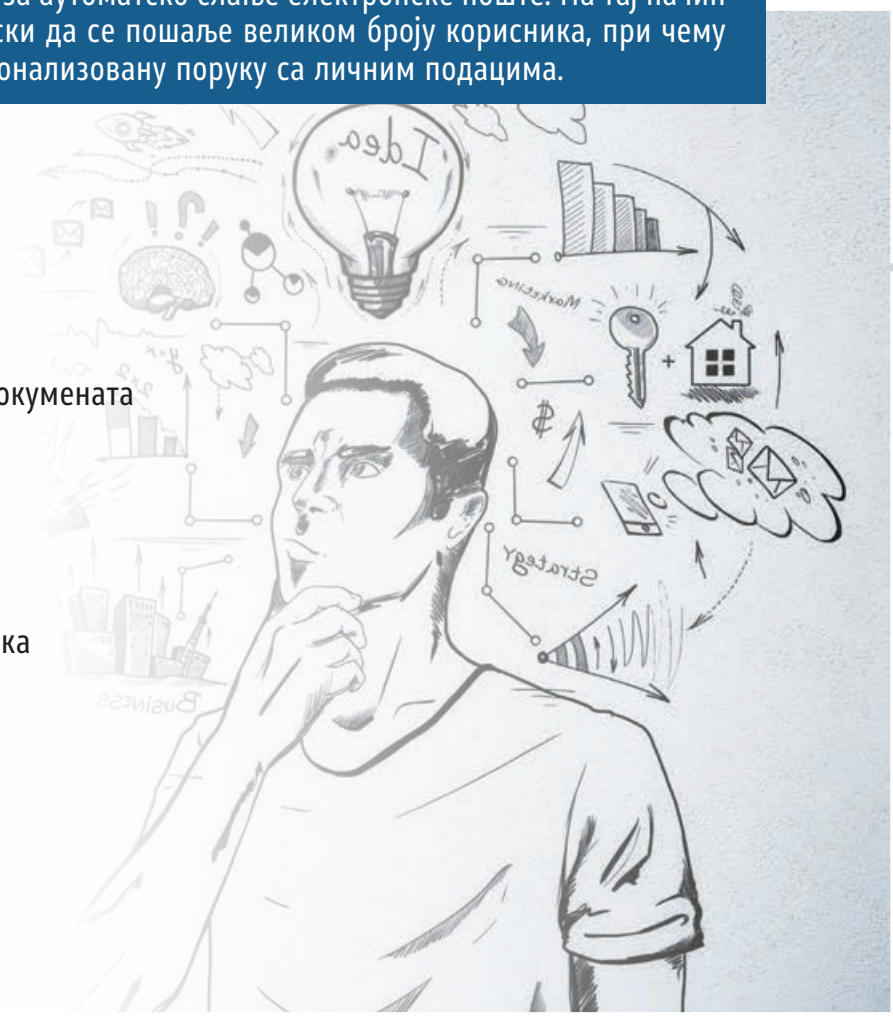
- врсту документа;
- извор података;
- предности коришћења циркуларних писама.

ЗАПАМТИ ШТА СИ НАУЧИО

- Циркуларно писмо је документ повезан са извором података.
- Извор података може бити база података или табела.
- Поља за спајање служе за аутоматско уметање података.
- Један шаблон може да креира велики број различитих докумената.
- Циркуларна писма штеде време и смањују могућност грешака.
- Ова техника се користи за позивнице, сертификате, потврде и обавештења

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Савремени системи омогућавају да се циркуларна писма користе не само за штампање докумената већ и за аутоматско слање електронске поште. На тај начин један шаблон може аутоматски да се пошаље великом броју корисника, при чему сваки прималац добија персонализовану поруку са личним подацима.



ПРОВЕРИ СВОЈЕ ЗНАЊЕ

Заокружи тачан одговор

1. Циркуларно писмо служи за:

а) инсталирање програма
б) креирање више персонализованих докумената
в) форматирање диска
г) креирање базе података

2. Поља за спајање:

а) бришу податке
б) служе за аутоматско уметање података
в) креирају нову табелу
г) штампају извештаје

3. Извор података може бити:

а) база података
б) табела
в) база података или табела
г) оперативни систем

Тачно или нетачно

1. Циркуларно писмо користи један шаблон за више докумената.
2. Сваки документ мора ручно да се попуњава.
3. Подаци могу аутоматски да се преузимају из базе података.
4. Поља за спајање замењују се стварним подацима при генерисању докумената.

АНАЛИЗА

Школа треба да изради 300 сертификата за учеснике такмичења.

Објасни:

- зашто је циркуларно писмо боље решење од ручног попуњавања;
- који подаци би били потребни у бази података;
- која поља за спајање би се користила у сертификату.

3.9

SQL – ОСНОВНИ ЈЕЗИК ЗА РАД СА БАЗАМА ПОДАТАКА

ВЕЋ ЗНАШ ДА...

- креираш табеле у бази података;
- одређујеш поља и типове података;
- користиш примарни кључ;
- уносиш податке преко форми;
- претражујеш информације помоћу упита;
- припремаш извештаје;
- користиш податке из базе података при изради циркуларних писама.

ПИТАЊА ЗА РАЗМИШЉАЊЕ

- Како компјутер комуницира са базом података?
- Како се могу затражити само одређене информације из табеле?
- Како се могу сортирати резултати претраге?
- Да ли постоји стандардни језик који се користи у различитим базама података?
- Како се извршавају претраге у великим базама података?

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш шта представља SQL;
- користиш основне SQL наредбе;
- извршаваш једноставне претраге;
- постављаш услове при претраживању;
- сортираш резултате;
- анализираш резултате добијене помоћу SQL-а.

У претходним лекцијама приказано је како се подаци могу уносити преко форми, претраживати упитима и приказивати кроз извештаје. У већини програма ове активности могу да се извршавају графичким алаткама и менијима. Међутим, у позадини постоји стандардни језик који омогућава директну комуникацију са базом података.

Овај језик назива се SQL (Structured Query Language). Он представља међународни стандард за рад са релационим базама података и користи се у великом броју система за управљање подацима. Захваљујући SQL-у, корисници и програмери могу да креирају табеле, уносе податке, претражују информације и управљају садржајем базе података.

Иако савремени програми нуде графичке алатке за рад са базама података, многе њихове функције заправо генеришу SQL наредбе. Зато познавање основа SQL-а омогућава боље разумевање начина на који базе података функционишу.

Шта представља SQL?

SQL је специјализован језик који се користи за рад са релационим базама података. Његов назив потиче од енглеског израза Structured Query Language, што у преводу значи „структурирани језик за постављање упита" бази података.

Помоћу SQL-а могу да се траже информације из табела, додају нови записи, мењају постојећи подаци и бришу информације које више нису потребне. Због ове разноврсности SQL представља један од најважнијих језика у области база података.

У овој лекцији биће обрађене само основне SQL наредбе које омогућавају претраживање и селектовање података. Операције се извршавају над упитом. За сваку од следећих наредби најпре се креира табела, а затим упит како би SQL могао да се користи.

Наредба SELECT

Најчешће коришћена SQL наредба је SELECT. Она служи за приказ података из једне или више табела.

Размотримо табелу Ученици.

Име	Презиме	Одељење
Ана	Петрова	II-1
Марко	Стојанов	II-2
Јелена	Николовска	II-1

За приказ свих записа користи се:

```
SELECT * FROM Ucenici;
```

Звездица (*) означава да треба приказати сва поља из табеле.

Ако је потребно приказати само имена и презимена, може се написати:

```
SELECT Ime, Prezime
FROM Ucenici;
```

Наредба WHERE

У многим ситуацијама потребно је приказати само записе који испуњавају одређени услов. За ту сврху користи се наредба WHERE.

Претпоставимо да треба приказати само ученике из одељења II-1.

```
SELECT *
FROM Ucenici
WHERE Paralelka = 'II-1';
```

Систем ће приказати само записе који испуњавају наведени услов.

Услови омогућавају знатно брже претраживање података и користе се у готово свим стварним базама података.

Наредба ORDER BY

Понекад податке треба приказати одређеним редоследом. За ту сврху користи се наредба ORDER BY.

Претпоставимо да ученике треба сортирати по презимену.

```
SELECT *
FROM Ucenici
ORDER BY Prezime;
```

Резултати ће бити приказани азбучним редом.

Могуће је и сортирање опадајућим редоследом:

```
SELECT *
FROM Ucenici
ORDER BY Prezime DESC;
```

Кључна реч DESC означава опадајући редослед.

Наредба DELETE

Понекад је потребно уклонити одређене записе из табеле. За ту сврху користи се наредба DELETE.

На пример:

```
DELETE FROM Ucenici
WHERE Paralelka = 'IV-4';
```

Ова наредба ће обрисати све записе који испуњавају услов.

Због могућности губитка података, DELETE треба користити пажљиво и увек са јасно дефинисаним условом.

Практичан пример

Размотримо базу података која садржи информације о ученицима.

Потребно је приказати све ученике са просеком већим од 4.50 и сортирати резултате по презимену.

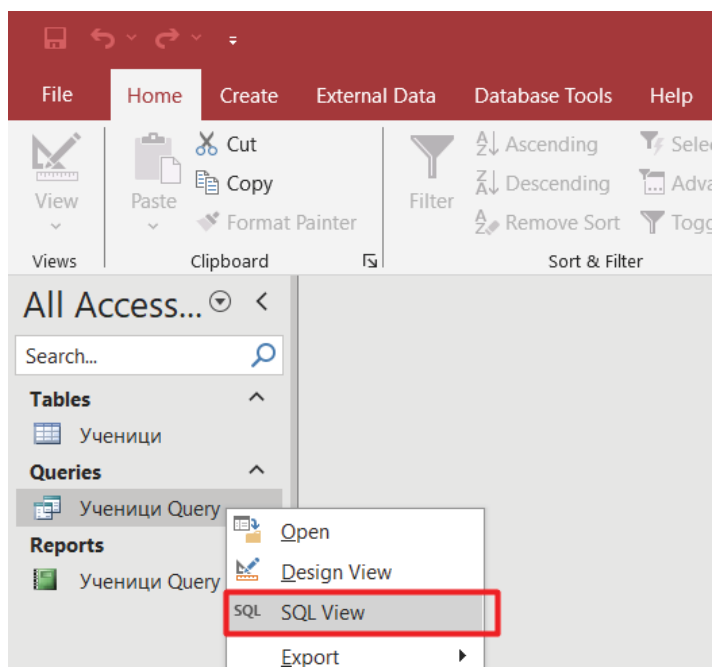
```
SELECT *
FROM Ucenici
WHERE Prosek > 4.50
ORDER BY Prezime;
```

У једној наредби комбиновани су претраживање, услов и сортирање резултата. То је једна од највећих предности SQL-а.

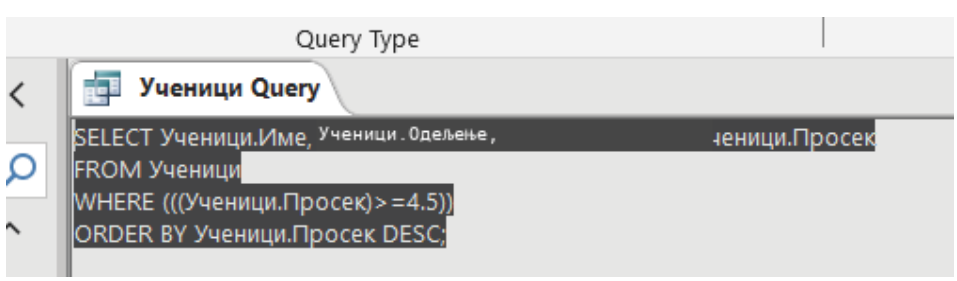
Анализа примера

Из претходног примера може се уочити да SQL омогућава једноставно и прецизно управљање подацима. Уместо да корисник ручно претражује записе, довољно је да зада одговарајућу SQL наредбу и систем ће аутоматски пронаћи потребне информације.

Захваљујући једноставној структури и великим могућностима, SQL данас представља стандард за рад са релационим базама података и присутан је у готово свим савременим информационим системима.



Слика 1 SQL приказ у MS Access-у



Слика 2 Наредбе у SQL-у

SQL представља стандардни језик за рад са релационим базама података. Помоћу њега могу да се претражују, сортирају и обрађују подаци који се чувају у табелама. Иако многи програми нуде графичке алатке за рад са базама података, у њиховој основи најчешће се користе SQL наредбе.

Наредбе SELECT, WHERE и ORDER BY омогућавају брзо проналажење и организовање информација, док се DELETE користи за уклањање непотребних записа. Захваљујући овим могућностима, SQL је важна алатка за рад са подацима у савременим информационим системима.

ДА ЛИ ЗНАШ?

Да ли знаш да се SQL користи више од четири деценије и да данас представља један од најраспрострањенијих програмских језика на свету? Готово сви велики информациони системи, веб-апликације и мобилни сервиси користе SQL за рад са подацима.

ПИТАЊА ЗА ПОНАВЉАЊЕ

1. Шта представља SQL?
2. Од којих речи је састављена скраћеница SQL?
3. Чему служи команда SELECT?
4. Чему служи команда WHERE?
5. Какву улогу има ORDER BY?
6. Када се користи команда DELETE?
7. Зашто се сматра SQL стандардним језиком рада са базом података

ПРАКТИЧНИ ЗАДАЧИ

Задатак 1

Напиши SQL команду којом ће се приказати сви записи из табеле Ucenici.

Задатак 2

Напиши SQL команду којом ће се приказивати само поља:

- Ime
- Prezime

из табеле Ucenici.

Задатак 3

Напиши SQL команду која ће приказивати само ученике одељења II-1.

Задатак 4

Напиши SQL команду која ће приказивати све ученике сортиране по презимену.

Задатак 5

Напиши SQL команду која ће приказивати ученике с просеком већим од 4.50.

Задатак 6

Објасни шта ће урадити следећа SQL команда:

```
SELECT *
FROM Ucenici
WHERE Prosek > 4.50
ORDER BY Prezime;
```

ИСТРАЖУВАЧКИ ЗАДАТАК

Истражи који системи за управљање са базама података користе SQL.

За најмање три система пронађи:

- име система;
- произвођач;
- област примене;
- занимљив податак о систему. Припреми кратак извештај или презентацију.

ЗАПАМТИ ШТА СИ НАУЧИО

- SQL је стандардни језик за рад са релационим базама података.
- SELECT служи за приказивање података.
- WHERE омогућава постављање услова при претраживању.
- ORDER BY се користи за сортирање резултата.
- DELETE служи за брисање записа.
- SQL омогућава брз и ефикасан рад са великим количинама података.
- Велики број савремених система користи SQL у свом раду.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Поред основних наредби, SQL садржи велики број напредних могућности. Њиме се могу повезивати табеле, извршавати статистички прорачуни, груписати подаци и креирати сложени извештаји. У професионалној примени SQL представља једну од најважнијих технологија у развоју информационих система и веб-апликација.

ПРОВЕРИ СВОЈЕ ЗНАЊЕ

Заокружи тачан одговор

1. SQL представља:

а) оперативни систем

б) програмски језик за графику

в) језик за рад са базама података

г) антивирусни програм

2. Команда SELECT служи за:

а) брисање записа

б) приказивање података в) штампање докумената

г) креирање образаца

3. Команда WHERE служи за:

а) сортирање података

б) постављање услова

в) брисање табела

г) креирање извештаја

4. ORDER BY се користи за:

а) сортирање резултата

б) брисање података

в) креирање поља

г) повезивање табела

Тачно или нетачно

1. SQL се користи за рад са базама података.
2. SELECT брише записе из табеле.
3. WHERE омогућава селектирање записа према услову.
4. ORDER BY омогућава сортирање резултата.
5. DELETE се користи за уклањање записа.

ДОПУНИ

1. Команда_____служи за приказивање података.
2. Команда_____служи за сортирање резултата.
3. Команда_____служи за брисање записа.

АНАЛИЗА

Разгледати табелу:

Име	Презиме	Просек
Ана	Петрова	5.00
Марко	Стојанов	4.20
Елена	Николовска	4.80

Напиши SQL команду којом ће бити приказани ученици са просеком већим од 4,50, сортирани према презимену.

Објасни шта ће бити приказано као резултат.

ЗАВРШНА МИСАО

Базе података представљају основу савремених информационих система. У оквиру ове теме приказано је како се подаци организују у табелама, уносе преко образаца, претражују помоћу упита, приказују у извештајима, користе у циркуларним писмима и обрађују помоћу SQL-а. Ова знања представљају важну основу за даље изучавање информатике и развој савремених софтверских решења.

TEMA 4

ТЕМА 4: МУЛТИМЕДИЈА И КОМПЈУТЕРСКА ГРАФИКА

Мултимедија и рачунарска графика представљају важан део савремене технологије и свакодневног живота. Оне обухватају коришћење текста, звука, слика, видео-записа и анимације, али и креирање и обраду визуелних елемената помоћу рачунара. Захваљујући рачунарској графици, можемо да стварамо реалистичне слике, игре, филмове и дигиталне дизајне.

Мултимедија чини информације занимљивијим и лакшим за разумевање, нарочито у образовању и области забаве. Данас се ове технологије користе у многим областима, као што су маркетинг, медицина, архитектура и видео-игре. Због тога је познавање мултимедије и рачунарске графике важно за младе који одрастају у дигиталном свету

НОВИ ПОЈМОВИ:

мултимедија, елементи – текст, слика, звук, видео, графика, растерска графика, векторска графика, пиксел, резолуција, селекција, слојеви, видљивост.

ВЕЋ ЗНАШ ДА:

- Уређујеш слике помоћу основних алатки;
- копираш и премешташ делове слике, бришеш делове слике, додајеш ефекте слици.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ

- Набroj најмање три алатке за уређивање слика.
- Шта значи „копирање дела слике“?
- Који су најчешћи ефекти који могу да се примене на слици?
- Објасни разлику између „копирања“ и „премештања“ дела слике.
- Објасни како визуелни ефекти мењају појаву слике.
- Копирај објекат са једне слике и постави га у другу.
- Додај ефекат „црно-бело“ датој слици.
- Упореди оригиналну и уређену слику – које су разлике?
- Анализирај како премештање објеката мења композицију слике.
- Објасни зашто брисање одређених елемената може да побољша слику.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- Објашњаваш о мултимедији, њеним основним елементима и примени, као и о карактеристикама растерске и векторске графике;
- примењујеш основне алатке и технике за обраду слика, укључујући корекције, слојеве и комбиновање графичких елемената;
- користиш алатке са вештачком интелигенцијом за генерисање и трансформацију слика, да формулишеш једноставне инструкције (prompt) и да анализираш добијене резултате;
- креираш једноставна графичка решења (постер, банер или лого) са применом изучених техника и ВИ алатки.

4.1

МУЛТИМЕДИЈА

У савременом дигиталном свету информације се ретко преносе само на један начин. Када читамо електронску књигу, гледамо образовни видео, пратимо презентацију, користимо апликацију или претражујемо садржаје на интернету, најчешће се истовремено сусрећемо са текстом, сликама, звуком, видео-записима и интерактивним елементима. Такав начин представљања садржаја близак је човеку јер ни у свакодневном животу појаве не доживљавамо изоловано, већ истовремено помоћу више чула и искустава. Овакав начин преношења информација представља основу за разумевање појма мултимедије.

Појам мултимедија може се објаснити као начин представљања информација комбиновањем више елемената, као што су текст, слика, звук, видео и анимација. Ови елементи користе се заједно како би се одређени садржај објаснио, приказао или пренео на потпунији начин.

Реч мултимедија састоји се од два дела: „мулти“, што значи „више“, и „медија“, што се односи на медије, односно начине или средства помоћу којих се преносе информације.

У информатици се мултимедија повезује са дигиталним садржајима и апликацијама у којима се комбинују различити елементи, као што су текст, слика, звук, видео, анимација и интерактивни делови. Када су ови елементи повезани у једну целину, настаје мултимедијални садржај или мултимедијална апликација. Примери мултимедијалних садржаја су презентација, веб-страница и електронски уџбеник, док су образовна апликација и рачунарска игра примери мултимедијалних апликација.

Делови од којих се састоји мултимедијални садржај називају се мултимедијални елементи. Основни мултимедијални елементи су текст, слика, звук, видео, анимација и интерактивни елементи. Сваки од њих има своју улогу: текст пружа информације, објашњења или упутства; слика визуелно приказује појам, предмет или појаву; звук преноси говор, музику или звучне сигнале; видео приказује кретање, поступак или догађај, док интерактивни елементи омогућавају кориснику да бира, одговара или управља деловима садржаја.

Важно је знати да мултимедија не подразумева само постављање више елемената на једно место. Они треба да буду повезани са темом и да имају заједнички циљ. На пример, презентација са текстом, сликама и кратким видео-записом о једној наставној теми представља мултимедијални садржај. Међутим, ако су текст, слике и звук постављени без реда и нису повезани са темом, неће допринети разумевању информација.

Мултимедија има широку примену у свакодневном животу јер омогућава да се информације представе на разумљив, занимљив и практичан начин. Она се користи у различитим областима, као на пример:

- У образовању мултимедија помаже да се наставни садржаји објасне и прикажу на начин који је лакши за разумевање. На пример, уместо да ученик само чита о Сунчевом систему, може да гледа анимацију о кретању планета, слуша објашњење и одговара на интерактивна питања. Тако учење постаје једноставније и активније.

- У медицини мултимедија може да помогне при објашњавању различитих здравствених стања, функционисања органа или одређених медицинских поступака. На пример, лекар може да користи снимке, 3Д приказе, видео-записе и кратка објашњења како би пацијент јасније разумео шта се дешава у његовом телу или како ће одређени поступак бити изведен.

- У туризму, мултимедија помаже људима да упознају неко место пре него што га посете. На пример, туристичка веб-страница или апликација може да садржи фотографије, мапе, видео-записе, виртуелне обиласке и кратке информације о знаменитостима, хотелима или музејима.

- У инжењерству се мултимедија користи за једноставније представљање сложених пројеката и техничких система. На пример, рад машине може се приказати помоћу дигиталне симулације која садржи анимацију кретања, текстуална објашњења, звучна упутства и могућност да корисник изабере део машине који жели да погледа.

- У спорту се мултимедија може користити за анализу спортских догађаја. На пример, анализа спортског догађаја може да садржи видео-снимке, успорене снимке, графиконе, статистичке податке и објашњења кретања играча.

Пример примене мултимедије су и рачунарске игре. Рачунарска игра може да садржи слику, звук, кретање, текстуална упутства и интерактивне елементе помоћу којих играч управља радњом.

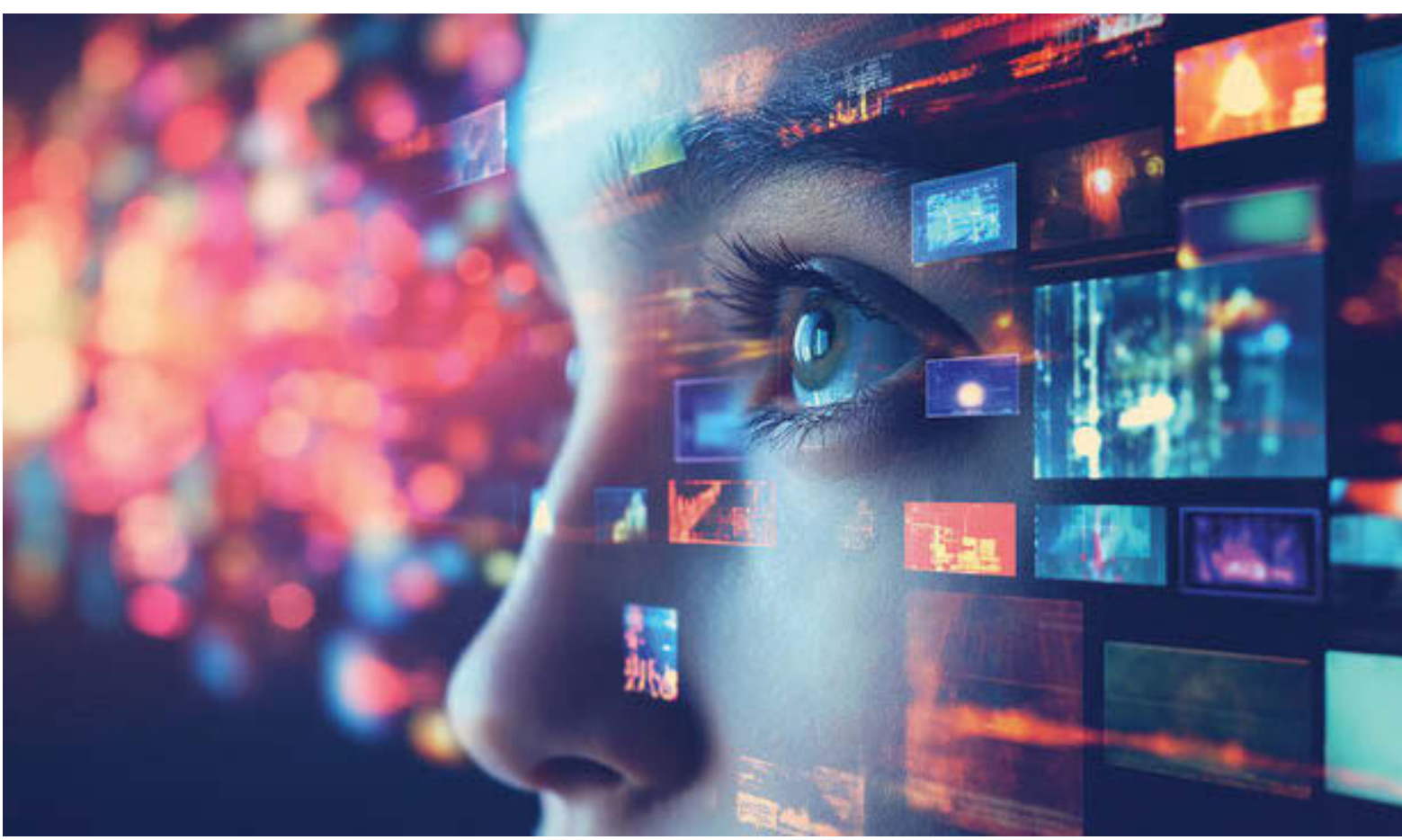
Мултимедија је део савременог начина учења, рада, информисања и решавања практичних задатака. Она није повезана само са рачунарима и мобилним уређајима већ и са начином на који се данас различите информације стварају, приказују и користе.

ЗАКЉУЧАК:

Мултимедија представља важан део савременог дигиталног света јер омогућава повезивање различитих врста садржаја. Захваљујући њој, информације постају јасније, занимљивије и лакше за разумевање. Препознавањем њених елемената и применом у стварним ситуацијама развијају се креативност и дигиталне вештине. Ова знања помажу у ефикаснијој комуникацији и учењу у савременом друштву

ПИТАЊА ЗА ПОНАВЉАЊЕ ЗНАЊА

1. Шта је мултимедија?
2. Наброј најмање четири елемента мултимедије.
3. Објасни зашто је мултимедија важна у свакодневном животу.
4. Објасни како слике и звук заједно побољшавају презентацију.
5. Наведи два конкретна примера где се мултимедија користи у образовању.
6. Одреди који се елементи мултимедије користе у једном YouTube видеу.
7. Анализирај који су мултимедијални елементи присутни у једној онлајн игри.
8. Упореди два различита примера мултимедије (нпр. филм и веб-страницу) према њиховим елементима.
9. Процени који је најважнији елемент мултимедије и објасни зашто.
10. Наведи пример из свакодневног живота у којем ти је мултимедија помогла да боље разумеш неку информацију. Објасни који су елементи били употребљени.
11. Изабери једну област, на пример медицину, туризам, спорт или инжењерство, и предложи како се мултимедија може употребити за решавање конкретног проблема у тој области.



4.2

ВЕКТОРСКА И РАСТЕРСКА ГРАФИКА

У свакодневном животу често се сусрећемо са фотографијама, цртежима, иконама, мапама, шемама, логотипима и другим визуелним приказима. Они могу да пренесу поруку, представе предмет, прикажу податак или помогну да се одређено упутство јасније прати.

Представљање информација, идеја или предмета помоћу визуелних елемената, као што су слике, линије, облици, боје и симболи, назива се графика.

Када се графика креира, обрађује и приказује помоћу компјутера, реч је о компјутерској графици.

У зависности од основних елемената од којих је слика састављена, најчешће разликујемо два основна типа компјутерске графике: растерску и векторску графику.

У векторској графици слика је представљена линијама, кривама, облицима и другим графичким објектима. Ови елементи се не записују као мрежа пиксела, већ се описују математичким правилима. Компјутер користи та правила како би приказао слику на екрану.

На пример, кружница у векторској графици није записана као велики број малих тачака, већ као облик са одређеним положајем, величином, линијом и бојом. Због тога се векторска слика може једноставно увећавати или умањивати, а да не постане замућена или састављена од малих квадратића. Линије и облици остају јасни и оштри чак и када се слика прикаже увећана.

Векторска графика најчешће се користи за логотипе, иконе, знакове, симболе, техничке цртеже, илустрације и дизајне који треба да се употребљавају у различитим величинама. На пример, исти логотип може бити мали на визиткарти, али и веома велики на плакату или билборду, а да не изгледа замућено.

Предност векторске графике је у томе што се може увећавати и умањивати без губитка јасноће линија и облика. Такође, појединачни елементи слике, као што су линије, облици и боје, могу се једноставно мењати. Недостатак је то што векторска графика није најпогоднија за приказивање фотографија и реалистичних слика са много нијанси, сенки и ситних детаља.

За рад са векторском графиком користе се програми као што су Inkscape, Adobe Illustrator, CorelDRAW, Figma и LibreOffice Draw.

У растерској графици слика је представљена као мозаик малих обојених тачака које се називају пиксели. Пиксел представља најмањи елемент од којег је састављена растерска слика. Када су пиксели веома мали и густо распоређени, људско око их не уочава појединачно, већ их повезује у једну целовиту слику.

Код растерских слика важан појам је резолуција. У свакодневној употреби она се често повезује са димензијама слике у пикселима, на пример 1920 × 1080 пиксела. То значи да слика има 1920 пиксела по ширини и 1080 пиксела по висини. При штампању и скенирању резолуција се најчешће изражава бројем тачака по инчу, на пример 300 dpi (dots per inch). Већи број пиксела омогућава да слика садржи више детаља, док је већа густина тачака при штампању важна за јаснији отисак.

Предност растерске графике је у томе што може добро да прикаже сложене слике са много боја, сенки и постепених прелаза између боја. Због тога је веома погодна за фотографије и реалистичне слике.

Недостатак растерске графике је то што при великом увећању слика може да изгуби квалитет. Пошто је састављена од пиксела, они при увећавању могу постати видљиви, па слика изгледа замућено или као да је састављена од малих квадратића. Такође, растерске слике високе резолуције могу заузимати више меморијског простора.

За рад са растерском графиком користе се програми за цртање и обраду слика, као што су Microsoft Paint, Adobe Photoshop, GIMP, Krita и Paint.NET.

Графичке датотеке могу се чувати у различитим форматима. Од формата могу зависити величина датотеке, квалитет слике, могућност коришћења провидне позадине и начин на који се слика може употребљавати или уређивати.

Основни формати слика су:

- JPG или JPEG је растерски формат који се најчешће користи за дигиталне фотографије и слике са много боја и детаља. Погодан је за фотографије које се постављају на интернет или шаљу електронском поштом, али није погодан за слике са провидном позадином.

- PNG је растерски формат који чува слику без губитка квалитета приликом компресије. Овај формат подржава провидну позадину, па се често користи за иконе, слике за веб-странице, снимке екрана, графичке елементе и слике код којих је важно да ивице и детаљи остану јасни.

- SVG је формат који се најчешће користи за векторску графику. Погодан је за логотипе, иконе, симболе и једноставне илустрације које треба приказивати у различитим величинама, а да не постану замућене или састављене од малих квадратића.

При избору формата треба водити рачуна о намени слике. За фотографије се најчешће користи JPG, за слике са провидном позадином и јасним графичким елементима PNG, а за логотипе, иконе и векторске илустрације SVG.

Други формати графичких датотека су:

- За растерску графику: GIF, BMP, TIFF, RAW, PCX и TGA.

- За векторску графику: AI, EPS, CDR, DXF, WMF и EMF.

Разумевање разлике између растерске и векторске графике помаже при правилном коришћењу, уређивању и чувању слика. Када се зна како је слика изграђена и у којем је формату сачувана, лакше је изабрати одговарајући начин њене обраде, штампања или дељења.

ЗАКЉУЧАК:

Разумевање разлике између растерске и векторске графике кључно је за правилну употребу дигиталних слика. Познавање појмова пиксел и резолуција помаже у процени квалитета слике. Сагледавање предности и недостатака оба типа графике омогућава избор најприкладнијег приступа у зависности од потребе. Познавање формата JPG, PNG и SVG доприноси ефикаснијем коришћењу и чувању графичких садржаја.

ПИТАЊА ЗА ПОНАВЉАЊЕ ЗНАЊА

1. Упореди растерску и векторску графику према начину на који је слика изграђена.
2. Објасни зашто се дигитална фотографија најчешће чува као растерска слика, а лого као векторска графика.
3. Анализирај шта ће се догодити ако се растерска слика ниске резолуције увећа за штампање великог постера.
4. Упореди формате JPG и PNG. У којој ситуацији би изабрао/ла JPG, а у којој PNG? Објасни зашто.
5. Упореди растерску и векторску слику према њиховом понашању при увећавању (zoom).
6. Анализирај како пиксели утичу на квалитет фотографије.
7. Објасни разлику између JPG и PNG формата у погледу квалитета и величине датотеке.
8. Процени који је формат најприкладнији за следеће ситуације: фотографија за веб-страницу, лого за школу и икона са провидном позадином. Образложи избор.
9. Замисли да треба да припремиш постер за школски догађај. Које елементе би израдио/ла као растерску, а које као векторску графику? Објасни избор.

4.3

ОСНОВИ ОБРАДЕ СЛИКЕ

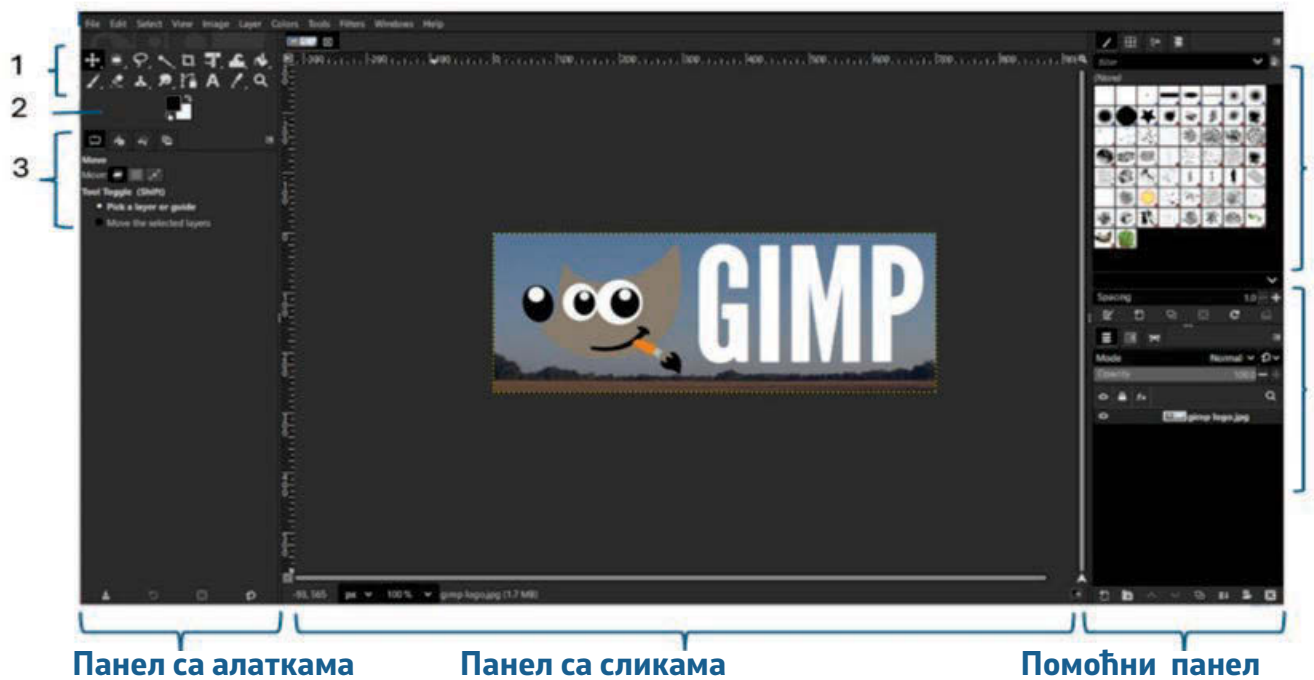
Обрада слике представља поступак мењања и прилагођавања дигиталне слике ради побољшања њеног изгледа, квалитета или употребне вредности. Она обухвата различите поступке, као што су уклањање непотребних делова, промена величине, подешавање осветљености, контраста и боја, додавање текста или других елемената, као и примена одређених ефеката. Овим поступцима слика се може припремити за штампање, презентацију, веб-страницу или другу врсту дигиталног садржаја. Због тога је познавање основних техника обраде слика важно за правилно и ефикасно коришћење визуелних информација.

Један од програма који се користе за рад са растерском графиком је ГИМП (GIMP – GNU Image Manipulation Program). GIMP је слободна софтверска апликација за обраду слика, доступна под Општом јавном лиценцом GNU GPL. Ова лиценца омогућава корисницима да слободно користе програм, проучавају и мењају његов изворни код, као и да деле измењене верзије под условима исте лиценце.

GIMP нуди радно окружење у којем корисник може да отвара, уређује и чува слике у различитим форматима. Садржи велики број алата за рад са сликама, организованих у менije, панеле и палете, што омогућава поступно и прегледно извршавање задатака. Посебно значајна могућност је рад са слојевима, помоћу којих се различити делови слике могу засебно уређивати.

4.3.1. РАДНО ОКРУЖЕЊЕ ГИМПА

У режиму једног прозора (Single-Window Mode), радно окружење програма GIMP најчешће је организовано у три главна дела: леви панел са алатима, централна радна површина на којој се слика приказује и уређује и десни панел са додатним дијалозима, као што су слојеви, канали, путање и друга подешавања. Ширина панела може се прилагодити повлачењем граничне линије између њих.



Слика 4.1: Радно окружење ГИМПа

Панел са алаткама садржи:

1. алате за обраду слике, који се могу изабрати и преко менија Tools;

2. икона за избор предње (главне, примарне) и позадинске (секундарне) боје у облику белог и црног правоугаоника;

3. својства изабраног алата (помоћни панел Tool Options).

Помоћни панел на десној страни радног окружења подељен је на горњу групу помоћних прозора, означену бројем 4, и доњу групу, означену бројем 5. Они пружају додатне могућности за обраду слике, као што су рад са слојевима, каналима, четкицама и друго. Помоћни прозори који се не користе могу се затворити командом Close Tab из менија који се отвара кликом на малу стрелицу у горњем десном углу. Поново се могу отворити избором Windows → Dockable Dialogs.

Централни део радног окружења заузима панел за слику. На врху овог панела налазе се картице са умањеним приказима учитаних слика.

Изнад слике налази се хоризонтални лењир, а са њене леве стране вертикални лењир.

Испод слике и са њене десне стране налазе се хоризонтални и вертикални клизач.

Статусна трака на дну панела за слику подељена је на четири дела: координате показивача миша, које се приказују само када се показивач налази у панелу за слику; падајућу листу за избор мерне јединице; поље за избор нивоа зумирања у процентима; и област у којој се приказују назив и величина датотеке из које је слика учитана. Подешавање радног окружења обавља се у прозору Preferences, који се налази у менију Edit.

4.3.2. ОТВАРАЊЕ, КРЕИРАЊЕ НОВЕ СЛИКЕ, СНИМАЊЕ, ЕКСПОРТИРАЊЕ СЛИКА

Команде за отварање (Open), креирање нове слике (New) и чување (Save, Save As) налазе се у менију File. Слике се у GIMP-у чувају у формату XCF. У другом формату могу се сачувати помоћу команде Export As.

4.3.3. ПРОМЕНА МОДЕЛА БОЈА СЛИКЕ

Промена модела боја може се извршити избором ставке Mode у менију Image. GIMP нуди три модела:

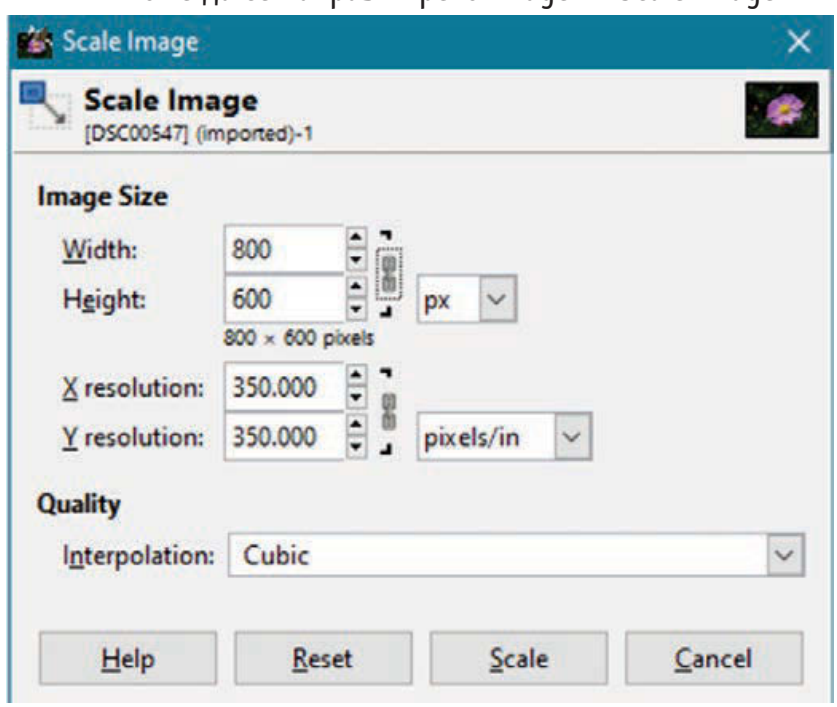
– RGB – стандардни модел боја са 24-битном дубином;

– Grayscale – сваки пиксел слике представљен је нијансом сиве боје;

– Indexed – боја сваког пиксела одређена је индексом (редним бројем) боје из палете која може да садржи до 256 боја.

4.3.4. ПРОМЕНА ВЕЛИЧИНЕ СЛИКЕ

Може да се направи преко Image -> Scale Image



Слика 4.2: Image -> Scale Image

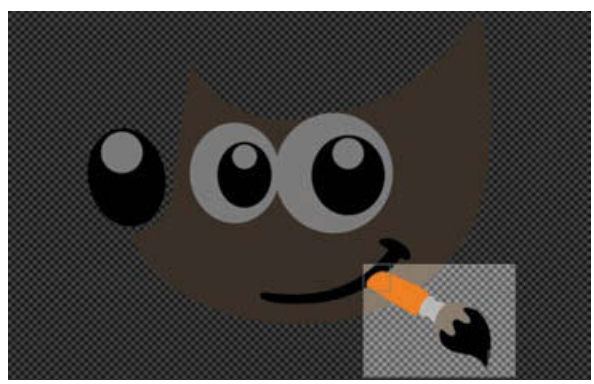
Нова ширина (Width) и висина (Height) могу се задати у различитим мерним јединицама, на пример у пикселима (px, pixels) или процентима (%, percent).

4.3.5. ПРОМЕНА ВЕЛИЧИНЕ ПЛАТНА

Платно (Canvas) одређује видљиву површину слике. Подразумевано се величина платна подудара са величином слике. Ако је платно веће од слике, око ње се ствара додатни простор. Ако је платно мање од слике, делови слике који се налазе изван платна неће бити видљиви. Међутим, ти делови се не губе, већ ће се поново појавити ако се платно увећа. Прозор за промену величине платна отвара се избором Image → Canvas Size.

4.3.6. ОДСЕЦАЊЕ ДЕЛА СЛИКЕ

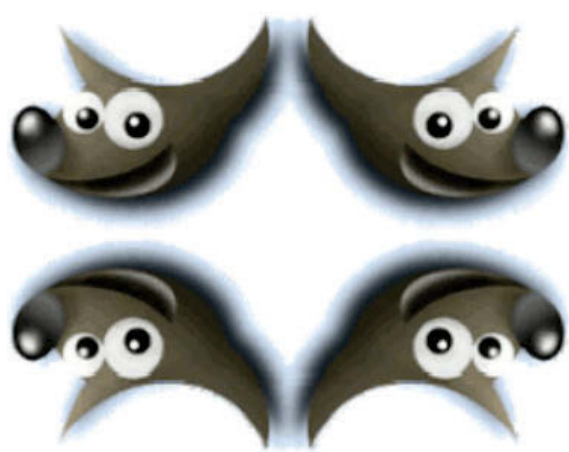
Обавља се избором алата Crop.  Означава се област за исецање и притиска се тастер Enter ради потврде. Резултат ће бити нова слика која садржи само унутрашњост означене области. Све што се налази изван те области биће одбачено. Исецање се може отказати притиском на тастер Esc.



Слика 4.3: Одсецање

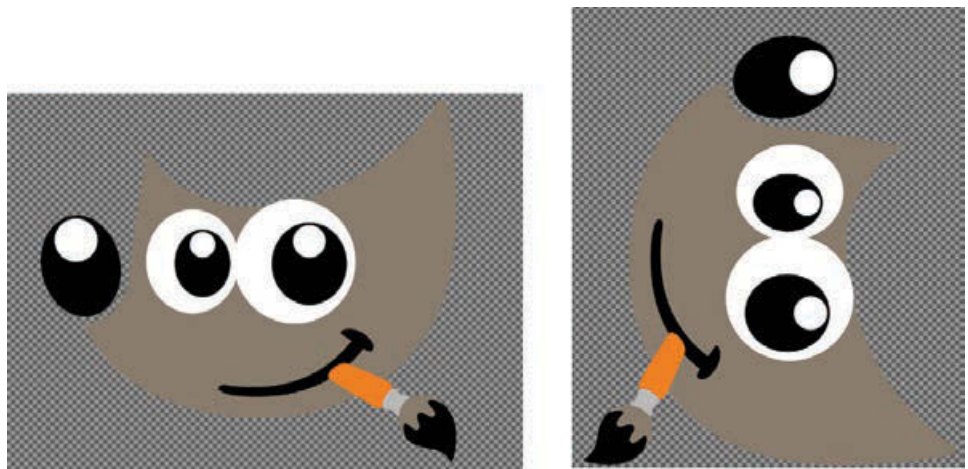
4.3.7. ПРЕСЛИКАВАЊЕ СЛИКЕ

Хоризонтално или вертикално пресликавање слике обавља се избором алата Flip из палете алата или избором Image → Transform → Flip Horizontally / Flip Vertically.



4.3.8. РОТИРАЊЕ

Роти рање слике обавља се избором алата за ротирање из палете алата или избором Image → Transform → Rotate 90° clockwise (ротирање за 90 степени у смеру казаљке на сату) / Rotate 90° counter-clockwise (ротирање за 90 степени супротно смеру казаљке на сату) / Rotate 180° (ротирање за 180 степени).



Слика 4.5: Ротација

4.3.9. ЦРТАЊЕ КРИВИХ И ПРАВИХ ЛИНИЈА

Следеће три алатке за цртање кривих и правих линија користе се на исти начин:

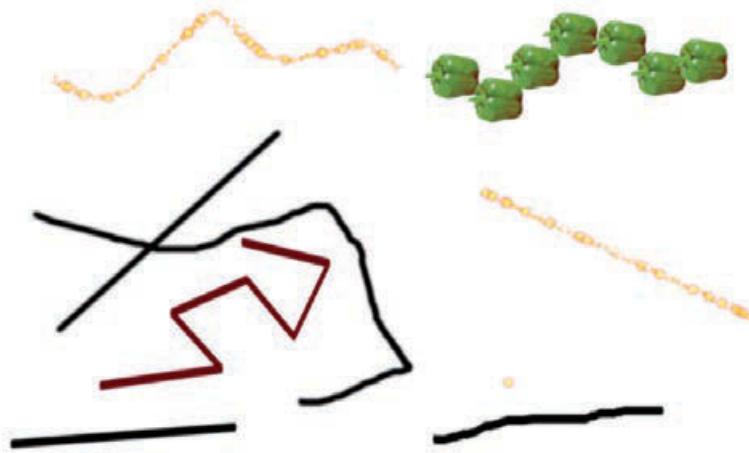
- оловка за цртање грубих линија,

- четка за цртање меких линија,

- спреј.

Најпре се у прозору са својствима алата подешавају опције као што су боја (Foreground Color), облик (Brush) и величина четкице (Size). Затим се притисне леви тастер миша на платну и показивач се помера без отпуштања тастера све док се не нацрта жељени облик.

Праве линије цртају се постављањем њихових крајњих тачака и држањем тастера Shift приликом клика на платно. Прва тачка поставља се без држања тастера Shift. Сваки пут када се кликне на овај начин, програм ће нову тачку повезати са претходном правом линијом.

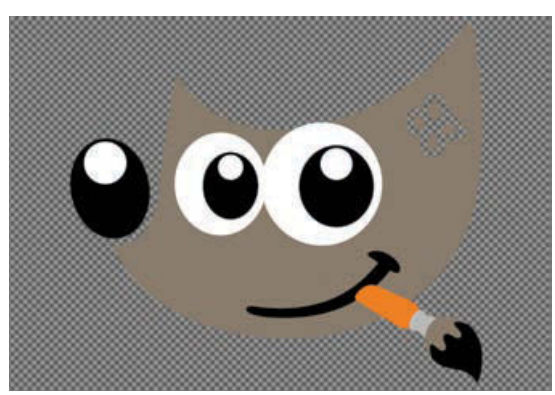


Слика 4.6: Цртање линија

4.3.10. БРИСАЊЕ ДЕЛОВА СЛИКЕ

Алат за брисање користи се на исти начин као и алати за цртање и има слична својства, као што су облик, величина четкице и непрозирност. За разлику од наведених алата за цртање, који користе примарну боју (Foreground Color), гумица за брисање користи позадинску боју.

Ако слој слике има провидну позадину, област избрисана гумицом биће провидна. Ако слој нема провидну позадину, избрисана област биће обојена позадинском бојом.

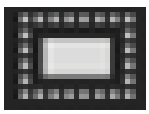


Слика 4.7: Брисање слике са транспарентном позадином

4.3.11. АЛАТКЕ ЗА СЕЛЕКТИРАЊЕ

Команде за селекцију груписане су у менију Select. Основне команде за рад са селекцијом су: All – селектовање целе слике, None – поништавање селекције, Invert – обртање селекције и Float – претварање селекције у привремени слој.

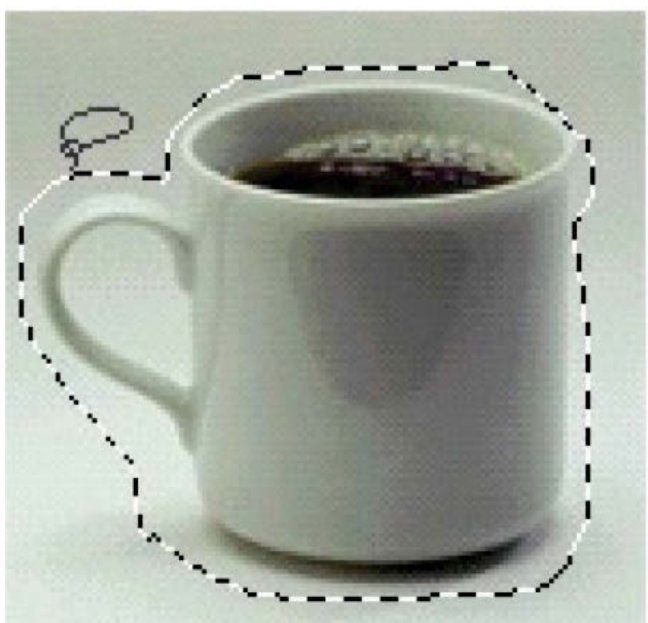
Постоји неколико алата за селектовање. Најједноставнији начин селекције јесте исцртавање дела који треба издвојити. У ту сврху могу се користити следећи алати из палете алата:



- алатка за избор правоугаоне површине

- алатка за елипсу

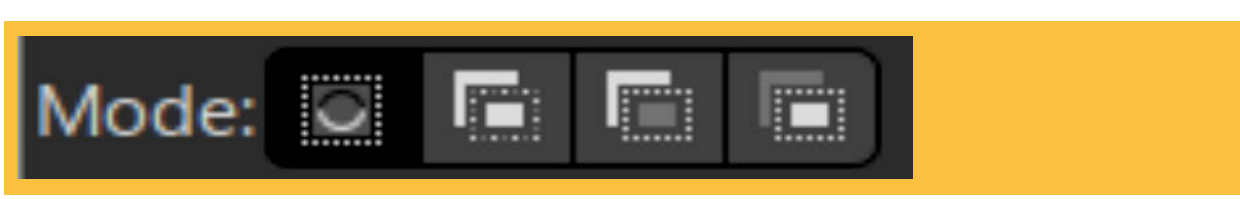
- алатка за слободан избор



Слика 4.8: Селекција са алатком за слободан избор

Селекција се потврђује притиском на тастер Enter или избором другог алата за уређивање слике. Држањем тастера Shift током селектовања правоугаоник се претвара у квадрат, а елипса у круг. Све док селекција није потврђена, њена величина може да се мења.

Избором алата за селектовање постаје доступна и опција Mode, која одређује однос између нове и постојеће селекције.



Има четири могућности:

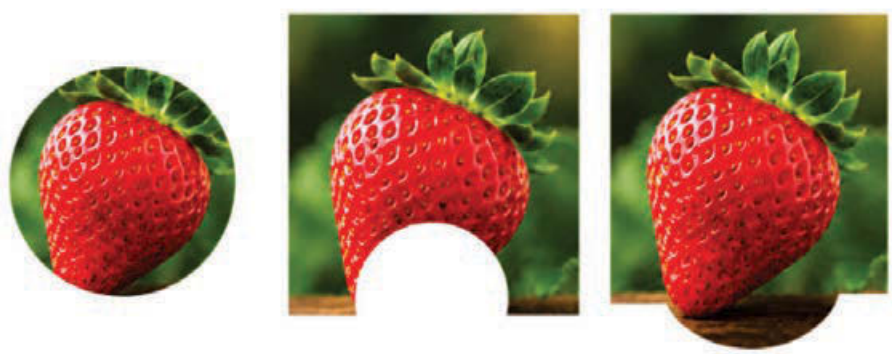
1. Replace the current selection – постојећа селекција се поништава и креира се нова.

2. Add to the current selection – постојећа селекција спаја се са новом (унија две селекције).

3. Subtract from the current selection – из постојеће селекције уклањају се делови који су заједнички са новом селекцијом (разлика – од постојеће селекције одузима се област нове селекције).

4. Intersect with the current selection – селекција обухвата заједничке делове постојеће и нове селекције (пресек постојеће и нове селекције).

Начин комбиновања нове селекције са постојећом треба изабрати пре исцртавања нове селекције.



Слика 4.10: Пресек, разлика и унија двеју селекција

У ГИМПУ је омогућена и селекција према боји. Алатке Fuzzy Select и Select by Color омогућавају да се изабере област која много личи на боју тачке на коју је кликнуто. Разлика између алатки Fuzzy Select и Select by Color је у томе што ће прва изабрати само област око почетне тачке, а друга ће изабрати све области на слици (не само око почетне тачке, већ било где на слици) које довољно личе на боју почетне тачке.

Селекција може да се изврши и алатком маказице. Алат Маказе користи се за селектовање делова слике неправилног облика. Корисник поставља тачке око објекта, а програм аутоматски прати његову ивицу на основу разлика у боји и контрасту. Након затварања селекције, изабрани део може се копирати, преместити, избрисати или додатно уредити.

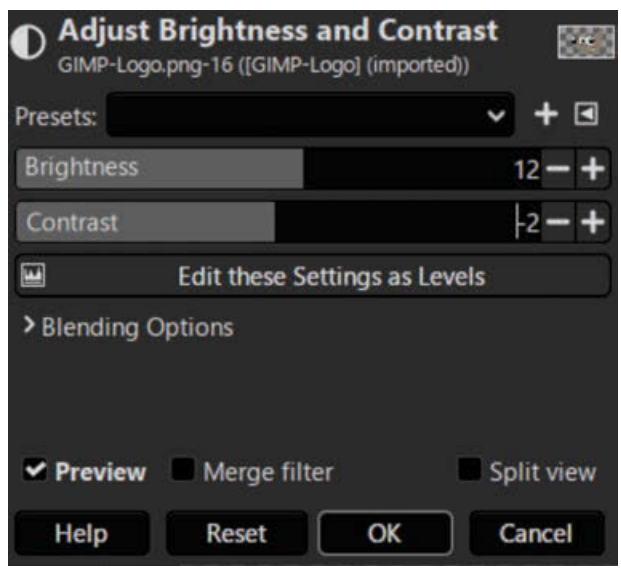
4.3.12. ОСВЕТЉАВАЊЕ СЛИКЕ

Користи се када је слика превише тамна или када одређени детаљи нису довољно видљиви. У GIMP-у се ова промена може извршити избором Colors → Brightness-Contrast. Након избора ове опције отвара се прозор у којем се вредност Brightness повећава или смањује помоћу клизача. Повећавањем вредности слика постаје светлија, а смањивањем тамнија. При уређивању треба водити рачуна да осветљеност не буде превелика јер се могу изгубити детаљи и слика може изгледати неприродно.

4.3.13. КОНТРАСТ СЛИКЕ

Користи се за наглашавање разлике између светлих и тамних делова слике. У истом прозору, помоћу клизача Contrast, вредност контраста може се повећати или смањити.

Повећањем контраста светли делови постају светлији, а тамни делови тамнији, па слика изгледа изражајније. Смањивањем контраста разлике између светлих и тамних делова постају мање, због чега слика може изгледати блеђе или мекше. Када се постигне одговарајући изглед, промена се потврђује притиском на дугме OK.



Слика 4.11: Дотеривање светла и контраста

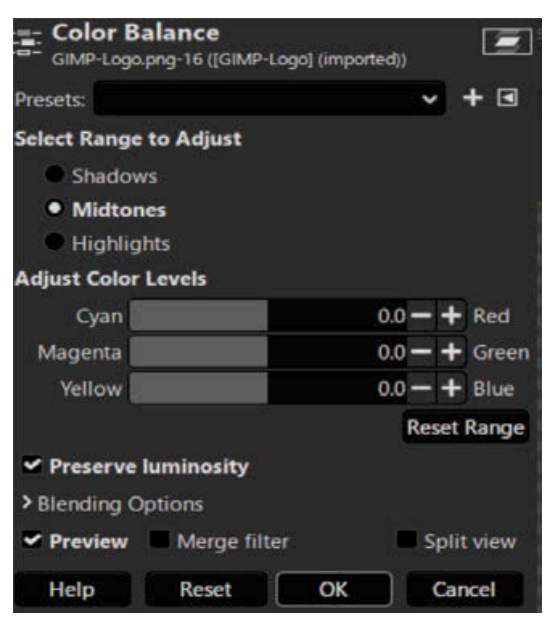
При операцијама повезаним са селекцијом користи се и алат за премештање.

Дебљина граничне линије селектоване области може се променити избором Select → Border, након чега се отвара прозор за подешавање.

4.3.14. БОЈА СЛИКЕ

За подешавање тамних боја бира се Select Range to Adjust → Shadows. Опција Midtones означава средње тонове, а Highlights светле тонове. Избором дугмета Reset сва подешавања враћају се на подразумеване вредности.

Боје на слици могу се мењати и помоћу команде Colors → Hue-Saturation.



Слика 4.12: Прозор Color Balance

Слика 4.13: Практична примена команде Hue-Saturation

4.3.15. КЛОНИРАЊЕ СЛИКЕ

Користи се за копирање делова једне области слике и њихово преношење на другу област. У GIMP ово се ради са алатком Clone Tool ( ), која се бира из панела са алаткама. Најпре се бира изворни део слике, односно област из које ће се копирати пиксели. То се ради држањем тастера Ctrl и кликом на изабрану област.

Затим се показивач миша поставља на место на које садржај треба копирати. Притисне се леви тастер миша и, без његовог отпуштања, показивач се превлачи преко дела који треба уредити. На тај начин изабрани садржај пресликава се на ново место.

При клонирању треба водити рачуна да изворни део одговара бојом, текстуром и осветљењем како би измена изгледала природно. Овај алат најчешће се користи за уклањање мањих неправилности, прекривање нежељених делова или понављање одређених елемената слике.

Слика 4.14: Пример за клонирање слике

4.3.16. АЛАТКА ЗА ПОПРАВЉАЊЕ СЛИКЕ

Алат Healing ( ) користи се за поправљање малих недостатака на слици, као што су мрље, тачке или огреботине.

- Кликне се на икону алата Healing.

- Означи се место са којег се узима узорак тако што се држи притиснут тастер Ctrl и кликне на то место. Након означавања извора тастер Ctrl треба отпустити. GIMP ће исцртати круг око изабране тачке.

- Показивач се постави на место које треба исправити. Ако је област довољно мала да стане у оквир четкице, често је довољан један клик. Ако је област већа од четкице, притисне се леви тастер миша и показивач се помера без његовог отпуштања све док се цела област не прекрије.

Ако је потребно, ради постизања најбољег резултата иста област може се поправити више пута, све док се исправљено место довољно добро не уклопи у околину.

Слика 4.15: Пример поправљања слике

4.3.17. РАД СА ТЕКСТОМ

У GIMP, поред обраде слике, може да се дода и уређује текст. У том циљу користи се алатка Text Tool ( ), која се бира из панела са алаткама. Након избора алата кликне се на место на слици на којем треба унети текст, а затим се упише жељени садржај. Текст се може уређивати помоћу опција за избор фонта, величине, боје, поравнања и размака између слова и редова.

При додавању текста GIMP аутоматски креира нови текстуални слој, што омогућава да се текст уређује одвојено од осталих делова слике

Слика 4.16: Трака за форматирање текста

ЗАКЉУЧАК:

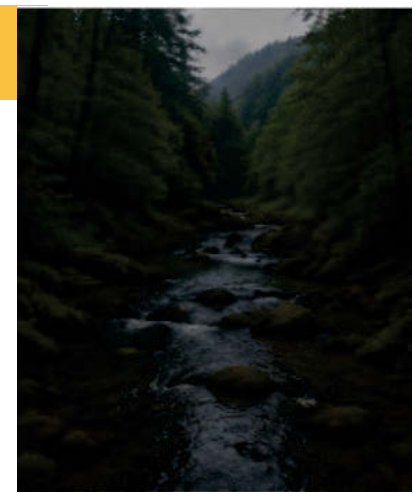
Радно окружење програма GIMP састоји се од три панела: панела са алатима, панела за слику и помоћног панела. Слике израђене у GIMP-у чувају се у формату XCF, са наставком .xcf. За чување слика у другом формату користи се команда File → Export As. Image → Scale Image користи се за промену величине слике, Image → Canvas Size за прилагођавање платна, а Crop Tool за исецање дела слике. Ротирање и пресликавање обављају се избором Image → Transform → Rotate / Flip. За селектовање објеката користе се алати Rectangle Select, Ellipse Select и Free Select, након чега се изабрани делови могу избрисати или преместити. Алати Eraser Tool, Clone Tool и Healing Tool користе се за уклањање или исправљање делова слике. Изглед слике подешава се командама Colors → Brightness-Contrast, Color Balance и Hue-Saturation. Текст се додаје помоћу алата Text Tool, чиме се омогућава креирање уређених и креативних графичких решења.

ПИТАЊА ЗА ПОНАВЉАЊЕ ЗНАЊА

1. Упореди промену величине слике са променом величине платна.
2. У којим ситуацијама би користио/ла један, а у којим други поступак?
3. Анализирај разлику између наредби Save As и Export As у програму GIMP. Зашто је важно да се пројекат сачува у формату XCF, а коначна слика извезе у PNG или JPG формат?
4. Објасни шта ће се догодити ако се слика претвори из RGB у Grayscale модел боја. Које информације са слике се губе?
5. Упореди алатке Fuzzy Select и Select by Color. У ком случају је боље користити једну, а у ком другу алатку?
6. Анализирај разлику између алатке Clone Tool и алатке Healing. Зашто Healing може да да природнији резултат при уклањању малих недостатака?
7. Објасни како режими селекције Add, Subtract и Intersect утичу на постојећу селекцију. Наведи пример када би сваки од ових режима био користан
8. Упореди исецање дела слике са селектовањем и брисањем делова слике. У чему је разлика у резултату?
9. Анализирај како промена осветљења и контраста може да побољша или погорша квалитет слике.
10. Процени која би алатка била најпогоднија за уклањање мале мрље са фотографије: Clone Tool, Healing или Eraser. Објасни избор.
11. Процени када је оправдано смањити контраст слике. Да ли повећавање контраста увек побољшава слику?
12. Процени да ли алатка Eraser увек брише део слике на исти начин. Како на то утиче да ли слој има транспарентну позадину или не?
13. Изабери најпогоднију алатку за селектовање објекта неправилног облика и објасни зашто је управо та алатка најприкладнија.
14. Вреднуј значај радног окружења у програму GIMP. Како панели, менији, лењири и статусна трака помажу кориснику при обради слике?
15. Осмисли поступак за уређивање фотографије која је тамна, благо искривљена и има непотребан простор око главног објекта. Које алатке и наредбе би користио/ла и којим редоследом?
16. Уради и следеће практичне вежбе.

Исеци слику тако да главни објекат остане у фокусу, а затим промени њену величину.

Изврши корекцију тамне и бледе слике.



Клонирај делове околне траве и попуни празну површину, тако да слика изгледа уредно и природно.

Користећи алатку за селекцију, промени боју цвећа.

Креирај задату слику.

4.4

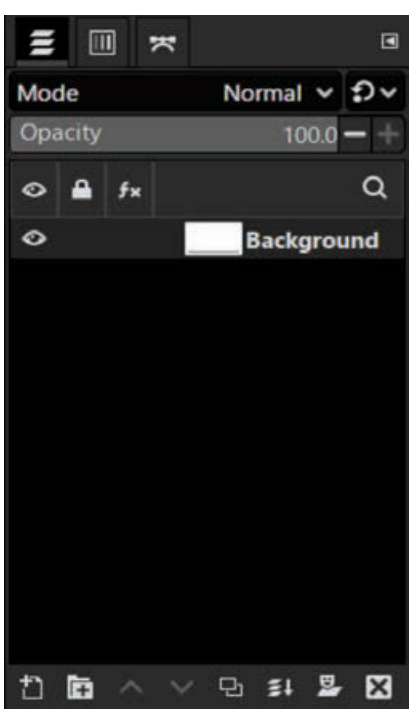
СЛОЈЕВИ (LAYERS) У ГРАФИЦИ

При обради слике у програму GIMP често се користи више елемената, као што су фотографија, текст, облик, цртеж или друга слика. Ради једноставнијег рада, ови елементи могу се поставити на посебне слојеве. Слојеви омогућавају да се сваки део слике уређује засебно, без непосредног мењања осталих делова.

Слој се може замислити као провидна површина постављена преко друге површине. Ако се на једном слоју налази фотографија, на другом текст, а на трећем облик, сви они заједно чине коначну слику. Ипак, сваки слој може засебно да се премести, избрише, сакрије, преуреди или измени. Управо зато су слојеви важни за организовану и прецизну обраду слике.

Креирај задату слику.

У GIMP-у слојеви се приказују у панелу Layers. Ако панел није видљив, може се отворити избором Windows → Dockable Dialogs → Layers. У овом панелу приказују се сви слојеви који се користе на слици, њихов редослед, називи и видљивост. Активни слој је онај који је изабран у панелу и на њега се примењују алати и команде.

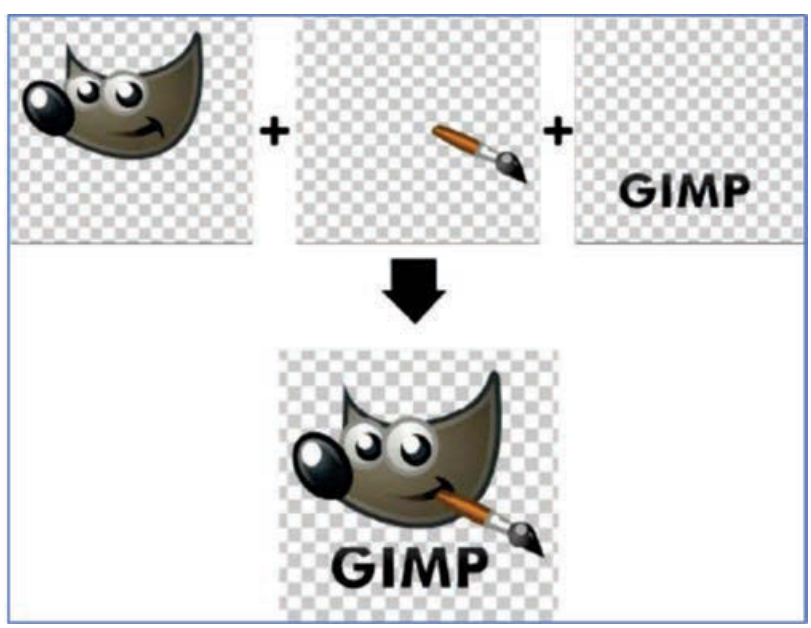


1. Креирање новог слоја
2. Креирање групе слојева
3. Померање слоја за једну позицију нагоре
4. Померање слоја за једну позицију надоле
5. Дуплирање активног слоја
6. Спајање активног слоја са слојем испод њега
7. Креирање маске слоја
8. Брисање слоја

Слика 4.17: Панел за слојеве

Нови слој се додаје наредбом Layer → New Layer или помоћу дугмета за нови слој у панелу Layers. При креирању новог слоја могу се задати његово име, величина и тип попуњавања, на пример, провидна позадина, бела позадина или одређена боја позадине. Именовање слојева је корисно, посебно када слика садржи више елемената.

Редослед слојева утиче на то који ће се елементи приказивати испред других. Слој који се налази више у панелу Layers приказује се испред слојева који су испод њега. Редослед се може променити превлачењем слоја нагоре или надоле у панелу. Такође се могу користити команде Layer → Stack → Raise Layer за померање слоја нагоре и Layer → Stack → Lower Layer за његово померање надоле. Након завршетка обраде слике слојеви се могу спојити. Командом Layer → Merge Down спајају се суседни слојеви. На пример, ако се слој са текстом налази изнад слоја са сликом, овом командом текст и слика биће спојени у један слој. Могу се спојити и сви слојеви који су тренутно видљиви, односно имају укључену икону „ока". То се изводи командом Image → Merge Visible Layers. Ова команда је корисна када пројекат садржи више слојева, али је потребно да се сви видљиви елементи налазе на једном слоју. Командом Image → Flatten Image сви слојеви спајају се у један слој и уклања се провидност. Она се најчешће користи пре чувања слике у форматима који не подржавају слојеве, као што је JPG.



Слика 4.18. Слика креирана применом слојева

Видљивошћу слоја управља се помоћу иконе ока која се налази поред назива слоја у панелу Layers. Када је око приказано, слој је видљив. Када се икона искључи, садржај тог слоја привремено се не види, али није избрисан. Ово је корисно када треба упоредити различите изгледе слике или када одређени елемент привремено не треба да буде приказан.

Слој се може избрисати тако што се изабере у панелу Layers, а затим употреби команда Layer → Delete Layer или дугме за брисање у панелу. Ако је потребно направити копију постојећег слоја, користи се команда Layer → Duplicate Layer. Ово је корисно када желимо да испробамо неку измену, а да притом сачувамо оригиналну верзију слоја.

Слика 4.19: Слика креирана од седам слојева

При раду са више слојева сваки елемент може се засебно уређивати. На пример, фотографија се може поставити као позадина, преко ње додати текст помоћу алата Text Tool, а затим на нови слој поставити облик или друга слика. Пошто се сваки елемент налази на посебном слоју, једноставно му се могу променити положај, величина, боја, видљивост или редослед.

У GIMP-у се може мењати и непрозирност слоја помоћу опције Opacity у панелу Layers. Смањивањем вредности непрозирности слој постаје провиднији и кроз њега се делимично могу видети слојеви испод њега. Ова могућност користи се за ублажавање оштрих граница између елемената, постизање постепених прелаза боја, складније спајање елемената или комбиновање више елемената у једну композицију.

Након завршетка уређивања, слика се може сачувати као радни пројекат у формату XCF избором File → Save As, чиме се чувају сви слојеви. Ако слику треба користити као обичну слику, на пример у документу, презентацији или на веб-страници, она се извози избором File → Export As у формату PNG или JPG. При извозу се слојеви спајају у једну коначну слику.

ЗАКЉУЧАК:

За једноставнију обраду слика слојеви се користе као начин организовања појединачних елемената унутар једне слике. На сваком слоју може се радити независно, тако да се садржај осталих слојева не мења. Рад са слојевима обавља се преко панела Layers, у којем је активни слој означен плавом бојом. Након обраде слике сви слојеви спајају се командом Image → Flatten Image. Слојевима се може управљати помоћу следећих команди: додавање (Layer → New Layer), брисање (Layer → Delete Layer) и дуплирање (Layer → Duplicate Layer), промена редоследа (Layer → Stack → Raise/Lower Layer) и мењање провидности (Opacity у панелу Layers).

Ученици ће умети да мењају видљивост и редослед слојева и тако контролишу који се елемент приказује изнад или испод другог.

Комбиновањем више слојева креираће сложенија и креативнија графичка решења.

ПИТАЊА ЗА ПОНАВЉАЊЕ ЗНАЊА

1. Шта су слојеви у програму GIMP?
2. Зашто се слојеви користе при обради слике?
3. Где се слојеви приказују у програму GIMP?
4. Како се отвара панел Layers ако није видљив?
5. Шта представља активни слој?
6. Како се додаје нови слој у програму GIMP?
7. Зашто је корисно именовати слојеве?
8. Како редослед слојева утиче на изглед слике?
9. Како се може променити редослед слојева?
10. Чему служи икона ока поред слоја?
11. Која је разлика између скривања и брисања слоја?
12. Када дуплирање слоја може бити корисно?
13. Зашто је добро да текст буде постављен на посебан слој?
14. У ком формату се чува радна верзија слике у програму GIMP?
15. Која је разлика између Save As и Export As?
16. Отвори нову слику у програму GIMP. Креирај најмање три слоја: позадину, облик и текст. Позадину попуни бојом, на другом слоју нацртај правоугаоник или други облик, а на трећем слоју унеси кратак текст.
17. Коришћењем слојева креирај постер о безбедном понашању на интернету.

4.5

ГЕНЕРАТИВНА ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА ЗА ГРАФИКУ

Генеративна вештачка интелигенција представља технологију која може да ствара нове садржаје на основу упутстава корисника. У области графике користи се за креирање слика, илустрација, постера, банера, логотипа, позадина и других визуелних елемената. Уместо да корисник слику у потпуности ручно црта или уређује, може да унесе опис на основу којег ВИ алат предлаже визуелно решење. Једна од најчешћих примена генеративне вештачке интелигенције у графици јесте text-to-image. Овај појам означава креирање слике на основу текстуалног описа. Корисник уноси једну или више реченица којима објашњава шта жели да буде приказано, а алат генерише слику према тим смерницама. На пример, ако се унесе опис „постер за школски сајам са књигама, светлим бојама и великим насловом“, ВИ алат може да креира слику која одговара том опису.

Текстуално упутство које се задаје ВИ систему назива се prompt. Оно треба да буде јасно, конкретно и довољно описно. У њему се могу навести објекти који треба да се појаве, стил слике, боје, распоред елемената, величина, атмосфера и намена графике. Што је prompt прецизнији, већа је вероватноћа да ће добијени резултат бити ближи замисли корисника.

На пример, prompt „направи слику природе“ превише је уопштен. Бољи prompt би био: „Креирај хоризонтални банер са планинским пејзажом, зеленом шумом, плавим небом и простором за наслов у горњем левом углу.“ У другом примеру алат добија више информација о садржају, бојама, распореду и намени слике.

Слика 4.20. Примена општих инструкција, извор Leonardo.ai

Слика 4.21. Примена општих инструкција, извор Leonardo.ai

Постоји више апликација и онлајн алата помоћу којих се могу генерисати графички садржаји коришћењем вештачке интелигенције. Примери су Adobe Firefly, Canva Magic Media, Microsoft Designer, Midjourney, ChatGPT (са могућношћу генерисања слика), Leonardo AI, Ideogram и Freepik AI. Ови алати омогућавају креирање слика, илустрација, постера, банера и других визуелних садржаја на основу текстуалног prompt-а. Неки су више усмерени на уметничке слике, други на графички дизајн и материјале за друштвене мреже, а трећи на уређивање и дораду постојећих слика.

ВИ алати за генерисање слика могу се користити као помоћ при дизајнирању. Они омогућавају брзо стварање почетних идеја, визуелних предлога и различитих варијанти истог дизајна. Корисник може да упореди више резултата, изабере најпогоднији и затим га додатно уреди у програму за графичку обраду. На тај начин ВИ не замењује у потпуности креативни рад, већ може да помогне при планирању, проналажењу инспирације и бржој изради графичких решења.

Генеративна вештачка интелигенција може да помогне и при претварању или прилагођавању графике из једног облика у други. Растерска слика састављена је од пиксела и најчешће се користи за фотографије и сложене слике са много детаља. Векторска слика састављена је од линија, облика и математички дефинисаних елемената, па се може увећавати без губитка квалитета. Помоћу одређених ВИ алата или алата са подршком вештачке интелигенције растерска слика може се поједноставити и претворити у векторски приказ, на пример у логотип, икону или илустрацију. Са друге стране, векторска графика може се извести као растерска слика када треба да се користи у документу, презентацији или на веб-страници.

При коришћењу ВИ алата важно је анализирати резултате. Генерисана слика не мора увек бити тачна, одговарајућа или квалитетна. Понекад може да садржи неправилности у облицима, тексту, пропорцијама, бојама или распореду елемената. Због тога корисник треба пажљиво да провери да ли слика одговара задатом циљу, да ли је јасна и визуелно уређена и да ли се може употребити за предвиђену намену.

Генерисана графичка решења могу се упоредити са ручно креираним графичким решењима. Ручно креирана графика омогућава већу контролу над сваким елементом јер корисник сам поставља облике, боје, текст и распоред. Графика генерисана помоћу ВИ, са друге стране, омогућава

брже добијање идеја и визуелних предлога, али понекад захтева додатну проверу и уређивање. Најбољи резултати често се постижу када се ВИ користи као помоћни алат, а корисник додатно уређује и обликује коначни дизајн.

У дизајну се ВИ алати могу применити при креирању постера, банера, насловних слика, реклама, илустрација, позадина и визуелних материјала за друштвене мреже. На пример, ученик може да генерише позадину помоћу ВИ алата, затим да је отвори у програму за обраду слика, дода текст, промени боје, уклони делове слике и припреми коначни постер. Тако се изучене технике графичке обраде комбинују са могућностима вештачке интелигенције.

При раду са ВИ алатима треба поштовати и правила одговорног коришћења. Не треба генерисати садржаје који вређају, обмањују или погрешно представљају особе и догађаје. Такође треба водити рачуна о томе да ли се слика сме користити за јавно објављивање, школски пројекат или у комерцијалне сврхе. Корисник увек треба критички да размотри добијени резултат и, ако је потребно, наведе да је при изради коришћен ВИ алат.

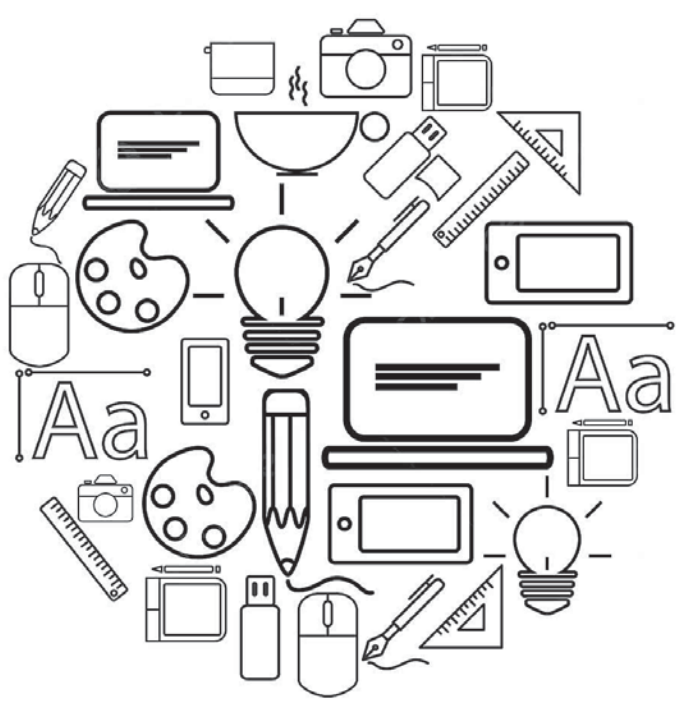
Генеративна вештачка интелигенција представља корисну подршку у савременој графичкој обради. Она омогућава брзо стварање идеја и визуелних решења, али успешан дизајн и даље зависи од знања, процене и креативности корисника. Због тога је важно да корисник уме да формулише јасан prompt, анализира добијене резултате и комбинује ВИ алате са ручним техникама обраде слика.

ЗАКЉУЧАК:

Коришћење ВИ алата за креирање дигиталних слика ефикасније је када се формулишу једноставна и јасна упутства (prompt-и) која непосредно утичу на квалитет добијених резултата. Анализом генерисаних слика развијају се критичко мишљење и способност процењивања тачности, креативности и релевантности садржаја које је створила ВИ. Поређење слика креираних помоћу вештачке интелигенције са ручно израђеним графичким решењима указује на предности, али и на ограничења оба приступа. ВИ може бити користан алат који допуњује, али не замењује људско креативно изражавање.

ПИТАЊА ЗА ПОНАВЉАЊЕ ЗНАЊА

1. Шта представља генеративна вештачка интелигенција?
2. Како се генеративна вештачка интелигенција користи у графици?
3. Шта значи појам text-to-image?
4. Шта је prompt?
5. Који подаци могу да се наведу у prompt-у за генерисање слике?
6. Наведи пример ВИ алатке помоћу које може да се генерише слика.
7. Зашто треба анализирати добијену слику из ВИ алатке пре него што се употреби?
8. Упореди слику генерисану помоћу ВИ и ручно креирану графику. Које су предности и недостаци оба начина?
9. Побољшај следећи prompt: „Направи постер за школу.” Додај детаље о теми, бојама, распореду и тексту.
10. Осмисли prompt за банер на тему „Дигитална безбедност”.
11. Помоћу ВИ алатке генериши почетну слику за постер на тему „Дигитална безбедност”. Најпре формулиши prompt у којем ћеш навести шта слика треба да садржи, које боје треба да се користе и где треба да буде простор за наслов. Затим анализирај добијени резултат и одлучи да ли је слика одговарајућа за постер. Отвори слику у програму за обраду слика, додај наслов, кратак текст и по потреби изврши корекције величине, боје или распореда. Коначни постер сачувај као радни пројекат и извези га као слику.



ПИТАЊА ЗА ПОНАВЉАЊЕ НА ТЕМАТА

Заокружи тачан одговор.

1. Шта представља мултимедија?

А. Уређивање фотографија у графичком програму
Б. Комбиновање више елеменат за преношење садржаја
В. Чување слика у разним форматима
Г. Претварање текста у табелу

2. Шта је растерска слика?

А. Слика састављена од линија и геометријских облика
Б. Слика састављена од пиксела
В. Слика која не може да се штампа

3. Шта је векторска слика?

А. Слика састављена од пиксела
Б. Слика састављена од звука и видеа
В. Слика састављена од линија, кривих и облика
Г. Слика која може да се отвори само у GIMP

4. Који формат се најчешће користи за фотографије?

А. JPG Б. SVG В. AI Г. DXF

5. Који формат је најбољи за лого које треба да се користи у разним величинама?

А. JPG Б. BMP В. SVG Г. RAW

6.У ком формату GIMP је сачувана радна верзија слике са слојевима?

А. JPG Б. PNG В. XCF Г. GIF

7. Шта значи text-to-image?

А. Претварање слике у текст
Б. Генерисање слике на основу текстуалног описа
В. Брисање текста са слике
Г. Чување текста као слике

8. Једна слика треба да се користи као мала икона на веб-страни, али и као велики знак на постеру. Који тип графике је најповољнији?

А. Растерска графика, зато што је састављена од пиксела

Б. Векторска графика, зато што може да се повећава без губљења јасности

В. JPG фотографија, зато што увек има малу величину

Г. GIF датотека, зато што је увек најквалитетнија

9. Ако растерска слика мале резолуције повећа много, шта ће се највероватније догодити?

А. Постаће оштрија

Б. Претвориће се у векторску слику

В. Изгледаће замућено или као да је састављена од малих квадрата

Г. Добиће провидну позадину

10. У GIMP ученик уређује текст, али промене се примењују на погрешном елементу. Који је највероватнији разлог?

А. Избран Изабран јеан активан слој

Б. Слика је премала

В. Фонт је превелики

Г. Документ је сачуван као XCF

11. Ако се слој са текстом постави испод слоја са тамном позадином, шта може да се деси?

А. Текст може да буде невидљив

Б. Текст аутоматски постаје већи

В. Позадина ће се избрисати

Г. Слика ће се извести као PNG

12. Која је разлика између сакривања слоја и брисања слоја?

А. Нема разлике

Б. Сакривање је привремено, а брисање трајно уклања слој

В. Брисање само смањује непровидност

Г. Сакривање мења величину слике

13. Ако је слика превише тамна, која команда је најбоља за почетну корекцију?

А. Colors → Brightness-Contrast

Б. File → Export As

В. Layer → Delete Layer

Г. Select → None

14. Зашто алатка Healing често даје природнији резултат од Clone Tool при уклањању мале мрље?

А. Зато што аутоматски брише целу слику

Б. Зато што прилагођава поправку према окружењу

В. Зато што увек претвара слику у векторску

Г. Зато што додаје текст на слику

15. Када је боље да се користи Crop Tool уместо Canvas Size?

А. Када треба да се уклони непотребни део око главног објекта

Б. Када треба да се дода нови простор око слике

В. Када треба да се промени боја текста

Г. Када треба да се креира нови слој

16. Који prompt је најбоље формулисан?

А. Направи нешто лепо.

Б. Слика за школу.

В. Креирај хоризонтални банер за школски сајам књига, са светлом позадином, књигама, ученицима и простором за наслов.

Г. Направи боју.

17. Када је најоправданије да се сачува пројекат као што је XCF?

А. Када слика треба касније да се поново уређује по слојевима

Б. Када слика треба одмах да се одштампа као фотографија

В. Када слика не садржи никакве елементе

Г. Када треба да се избрише текст

18. Треба да се изради постер са ВИ генерисаном позадином, насловом и додатном сликом. Који редослед рада је најбољи?

А. Прво да се извезе постер, затим да се дода текст

Б. Да се генерише позадина, да се провери резултат, да се уреди у графичком програму и да се дода текст

В. Да се избрише позадина и да се сачува празна слика

Г. Да се користи само један слој за све елементе

19. Ако треба да се направе две верзије истог постера, једна са логом и друга без лога, која могућност у GIMP највише помаже?

А. Искључивање и укључивање видљивости слоја са логом

Б. Промена резолуције екрана

В. Брисање свих слојева

Г. Отварање калкулатора

20. Ученик има фотографију са малом мрљом на небу. Који поступак је најбољи?

А. Да претвори целу слику у SVG

Б. Да користи алатку Healing или Clone Tool да поправи област

В. Да смањи слику на 10 пиксела

Г. Да дода звук

21. Ако је потребно направити једноставан постер у програму GIMP са позадином, обликом, сликом и текстом, како је најбоље организовати елементе?

А. Сви да се поставе на један слој

Б. Сваки главни елемент да буде на посебном слоју

В. Текст да се избрише након уношења

Г. Слика да се сачува само као звук

22. Који план најбоље комбинује ВИ алатку са ручном обрадом слике?

А. ВИ алатка да направи почетну слику, а корисник да је анализира, да је уреди и да је при-способи задатку

Б. Корисник не трба да проверава ништа и одмах да објави слику

В. Да се користи само ручна обрада, без никакве анализе

Г. Да се користи само један формат за слике, без обзира на њихову намену.

TEMA 5

ТЕМА 5: ВЕБ--РАЗВОЈ

Основе веб-развоја су креативан начин да упознаш како функционишу веб-странице које свакодневно користимо. Креирање веб-страница комбинује креативност и логичко размишљање. Свака веб-страница гради се помоћу три основне технологије: HTML поставља основну структуру, CSS уређује изглед и боје, а JavaScript додаје интеракцију и динамику страници.

Када научиш ове основе, моћи ћеш да правиш сопствене веб-странице и претвараш своје идеје у стварне дигиталне пројекте. Такође је важно да разумеш како да страница добро изгледа на телефону, таблету и компјутеру, као и да буде једноставна за коришћење.

Ово је први корак за свакога ко жели да започне са веб-развојем и можда се у будућности бави овом професијом.

НОВИ ПОЈМОВИ

- структура веб-странице, HTML документ, основне HTML ознаке – heading, paragraph, image, link, CSS, селектори, боје, фонтови, позадина, основни распоред елемената веб-странице (layout), основе адаптивног (responsive) дизајна, прилагођавање различитим уређајима, ВИ инструкције (prompt).

Већ знаш:

- Које могућности нуде различити интернет-сервиси.
- Која је улога WWW сервиса на интернету.
- Да наведеш примере за Web 1.0, Web 2.0, Web 3.0, Web 4.0, Web 5.0.

У САДРЖАЈИМА КОЈИ СЛЕДЕ НАУЧИЋЕШ ДА:

- објашњаваш основну структуру веб-странице и примењујеш HTML и CSS за креирање и уређивање веб-садржаја;
- примењујеш основне принципе распореда и адаптивног (responsive) дизајна при прилагођавању веб-страница различитим уређајима;
- користиш алатке са вештачком интелигенцијом за генерисање веб-садржаја;
- формулишеш инструкције (prompt), анализираш добијене резултате и уређујеш ВИ-генерисан код;
- креираш једноставан веб-пројекат комбинованом применом ручно израђеног и аутоматски генерисаног кода.

ЗАДАЦИ ЗА ПОНАВЉАЊЕ:

- Наведи најмање три различита интернет сервиса и опиши чему служе.
- Како WWW сервис утиче на начин приступа информацијама у поређењу са другим интернет сервисима (на пример е-поштом или FTP-ом).
- Објасни главне разлике између Web 1.0 и Web 2.0 користећи сопствене речи.
- Наведи конкретан пример за Web 3.0 и објасни како тај сервис побољшава могућности интернета.
- Упореди Web 2.0 и Web 4.0 у погледу интеракције са корисницима и објасни њихове главне карактеристике.
- Осмисли сопствени пример Web 5.0 апликације и објасни како би она помогла корисницима у свакодневном животу.

5.1

ПОЈАМ ВЕБ-СТРАНЕ

- Веб-страница је дигитални документ који се приказује у веб-прегледачу (browser), као што су Google Chrome, Mozilla Firefox или Microsoft Edge. Она је део ширег система који се назива веб-сајт и представља скуп више повезаних веб-страница. Сваки веб-сајт састоји се од једне или више веб-страница које су међусобно повезане хипервезама. Постављен је на веб-серверу на којем је инсталиран софтвер за испоруку веб-страница.
- Свака веб-страница има јединствену адресу, познату као URL (Uniform Resource Locator), преко које корисници могу да јој приступе путем интернета. Веб-странице се најчешће креирају помоћу језика HTML (HyperText Markup Language), који није програмски језик, већ служи за дефинисање структуре и садржаја странице. Он прегледачу даје податке о садржају и структури веб-странице и описује елементе и начин њиховог приказивања.
- Поред HTML-а, често се користе и друге технологије, као што су:
- • CSS (Cascading Style Sheets) – за стил и дизајн;
- • JavaScript – за интерактивност и динамичност.
- Веб-странице могу бити:
- • статичке – садржај је фиксан и не мења се често;
- • динамичке – садржај се мења у зависности од корисника или података (на пример, друштвене мреже или веб-продавнице).

5.1.1. ОСНОВНА СТРУКТУРА ВЕБ-СТРАНЕ

Свака веб-страница има логичку и техничку структуру која омогућава њено правилно приказивање и функционисање. Основна техничка структура дефинише се помоћу HTML-а, састоји се од неколико главних елемената и конструише се на следећи начин:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<titleНаслов страницеа</title>
</head>
<body>
<h1>Главни наслов</h1>
<p>Ово је пример текста.</p>
</body>
</html>
```

Овај код представља скелет веб-странице. Сваки елемент има следећу функцију:

1. <!DOCTYPE html>

Овај израз је декларација која прегледачу говори да је документ написан у најновијој верзији HTML5, односно да је веб-страница. Важна је за правилан приказ странице.

2. <html>

Ова ознака означава почетак HTML документа. Све што се налази између две <html> ознаке део је веб-странице.

3. <head>

У овај део уносе се метаподаци о страници, као што су:

- наслов (<title>),
- информације о скупу знакова,
- повезивање са CSS датотекама.

Овај део се не приказује директно кориснику.

4. <meta charset="UTF-8">

Користи се за правилан приказ текста на ћирилици. Пошто прегледач не мора сам правилно да препозна кодирање текста, потребно је назначити употребу Unicode стандарда, односно кодирања UTF-8.

4. <title>

Дефинише наслов странице који се приказује у табу прегледача.

5. <body>

Ово је најважнији део странице. Овде се налази све што корисник види:

- текст,
- слике,
- видео и
- линкови.

Поред техничке структуре, веб-страница има и логичку структуру која омогућава бољу организацију садржаја. Она често обухвата:

1. заглавље (header) односно горњи део веб-странице који најчешће садржи логотип, наслов и навигациони мени;

2. навигацију (navigation), мени који омогућава корисницима да се крећу кроз странице. Састоји се од линкова који воде до различитих секција или страница;

3. главни садржај (main content), централни део у ком се приказују основне информације. Може да садржи текст, слике, видео и друге елементе;

4. бочну траку (sidebar), додатни део који се налази са стране и садржи рекламе, додатне информације и линкове;.

5. подножје (footer), доњи део странице који најчешће садржи контакт податке, ауторска права и додатне линкове.

Добра структура веб-странице има велики значај из више разлога:

- Боља читљивост – корисници лакше разумеју садржај,
- Боље корисничко искуство – једноставна навигација и организација,
- Оптимизација за претраживаче (Search Engine Optimization) – претраживачи лакше анализирају страницу,
- Лакше одржавање – програмери могу једноставније да уређују и побољшавају страницу.

Правилно угнежђивање HTML ознака једно је од основних правила веб-развоја. Најједноставније речено, ознаке функционишу по принципу „кутија у кутијама“ – ознака која се последња отвори треба прва да се затвори (систем огледала или LIFO - Last In, First Out). Веб-прегледачи (као Chrome, Edge, Firefox) не приказују HTML код директно. Прво га претварају у структуру названу DOM (Document Object Model), која изгледа као породично стабло са родитељским и дечјим елементима (хијерархија).

- У правилном примеру <html> је родитељ, а <body> његово дете. Кутија <body> у потпуности је смештена унутар кутије <html>.

```
<!DOCTYPE html>
        <html>
                <head>
                        <title>Тачна структура</title>
                </head>
                <body>
                        <h1>Здраво!</h1>
                        <p>Ово је <strong>правилно</strong> угнежђен текст.</p>
                </body>
        </html>
```

У неправилном примеру (<html><body>...</html></body>): Покушавамо да затворимо спољашњу кутију (<html>) док је унутрашња кутија (<body>) још отворена.

```
<!DOCTYPE html>
         <html>
        <body>
        <head>
        <title>Нетачна структура</title> </head>
        <h1>Грешка!</h1>
        <pОво јее <strong>неправилно</p> угнежђен текст.</strong> </body>
</html>
```

Ако је код лоше угнежђен, веб-страница може изгледати потпуно различито у различитим прегледачима или на различитим уређајима. Елементи могу да се преклапају, „побегну“ са својих позиција или текст постане нечитљив.

ЗАПАМТИ ШТА СИ НАУЧИО

Веб-страница је основни елемент интернета и има кључну улогу у савременој комуникацији и информисању. Разумевање појма веб-странице и њене структуре основа је за даље учење у области веб-технологија. Основна техничка структура веб-странице састоји се од HTML елемената као што су <html>, <head> и <body>, док логичка структура обухвата делове као што су заглавље, навигација, главни садржај и подножје. Ове компоненте заједно омогућавају да веб-страница буде функционална, прегледна и корисна.

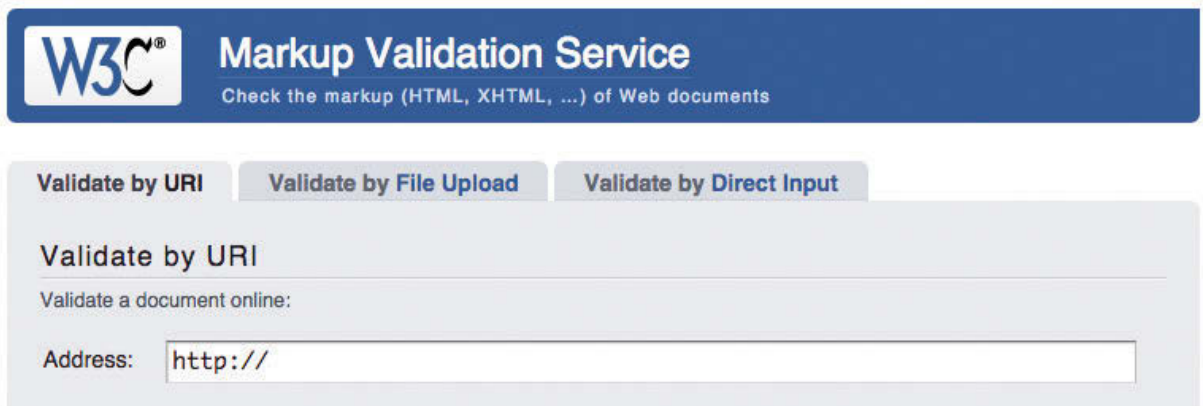
ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА

1. Која је почетна ознака која прегледачу говори да је документ веб-страница написана у HTML5?
2. Наведи три главне ознаке од којих треба да се састоји сваки правилан HTML документ.
3. Где се смештају метаподаци (као наслов странице или кодирање знакова) – у <head> или у <body> секцију?
4. Објасни разлику између садржаја у <head> ознаци и садржаја у <body> ознаци. Шта од тога види посетилац веб-странице?
5. Зашто је важно да HTML ознаке буду правилно угнежђене (на пример, зашто је <html><body>...</body></html> правилно, а <html><body>...</html></body> није)?
6. Својим речима опиши улогу ознаке <title>. Где се њен садржај приказује у веб-прегледачу?
7. Напиши комплетан код за празан HTML документ који садржи све обавезне структурне елементе, а наслов странице у табу прегледача треба да буде „Моја прва страница".
8. У следећем коду постоји структурна грешка. Каква ће се порука приказати? Пронађи грешку и напиши тачан код.

```
<!DOCTYPE html>
<html>
 <body>
  <head>
   <title>Добро дошли</title>
  </head>
  <h1>Здраво свете!</h1>
 </body>
 </body>
</html>
```

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Истражи алатку W3C Markup Validation Service. Проучи њену функционалност.



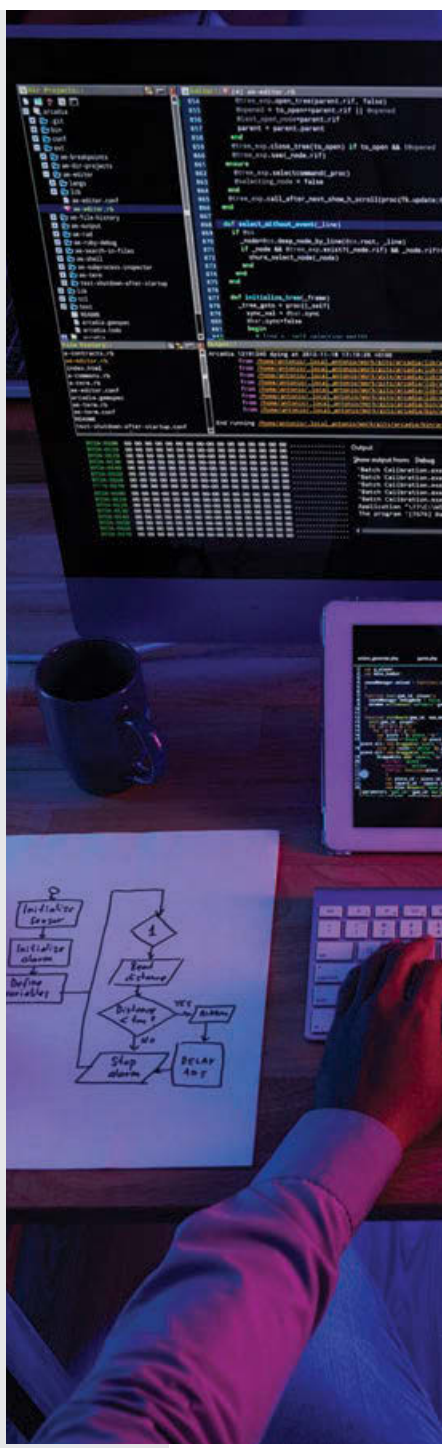
5.2

ОСНОВИ НА HTML

5.2.1. ОСНОВНЕ ОЗНАКЕ

Да би боље разумео како настаје веб-страница, замисли HTML код као архитектонски план куће. Сам код састоји се од ознака које прегледачу говоре где да постави прозор, где врата, а где кров.

У наставку ћемо размотрити основне HTML ознаке објашњене практичним примерима кроз које ћеш научити како да их примењујеш.



1. Ознака за наслове (<h1> до <h6>)

Наслови се користе за организовање садржаја и одређивање његове важности. <h1> је најважнији и највећи наслов (као наслов књиге), док је <h6> најмањи (за под-поднаслове).

- **Практичан пример:** Замисли да правиш веб-страницу са рецептима. Назив самог рецепта биће у <h1>, а поднаслови као „Састојци“ или „Начин припреме“ у <h2>.

  <h1>Рецепт за палачинке</h1>

  <h2>Потребни састојци</h2>

2. Ознаке за текст и наслове

- **пасус ( )** - користи се за писање обичног текста или одломака који се приказују као пасус. Сваки пут када отвориш нову ознаку, прегледач аутоматски оставља мало празног простора пре и после ње како би текст био читљив.

На пример, текст испод биће приказан као посебан пасус.

Палачинке су омиљени десерт свих генерација. Припремају се брзо и лако.

* **прелом
** – (break) користи се за принудно пребацивање текста у нови ред. Ова ознака је једна од ретких која **нема** завршну ознаку (пише се самостално).
* ознака **** ... , чини текст **подебљаним (Bold)** и корисна је за истицање важних речи.
* ознака **** ... , чини текст искошеним *(Italic)*.

3. Ознака за линкове (<a>)

Ознака <a> (anchor) служи за повезивање веб-странице са другом веб-страницом на интернету или другим документом. Да би прегледач знао куда води линк, користи се атрибут href. На пример, када желиш да упутиш посетиоце на Google да потраже више идеја за фил за палачинке, пишеш следећи код:

```
<a href="https://www.google.com">Потражите идеје на Google-у</a>
```

4. Ознака за слике (<img>)

Веб-странице би биле незанимљиве без визуелних елемената. Ознака <img> умеће слику на страницу. Важно је знати да ова ознака нема завршну ознаку (сама се затвара) и користи атрибуте src (путања до слике) и alt (текст који се појављује ако се слика не учита). За приказ укусне слике готових палачинки користи се следећи код:

```
<img src="palacinki.jpg" alt="Тањир са палачинкама и чоколадом">
```

Када све ове ознаке примениш у једној логичној целини унутар главног тела (<body>), добијаш следећи једноставан, али комплетан HTML код:

```
<!DOCTYPE html>
<html>
<head>
        <title>Мој први рецепт</title>
</head>
<body>
        <h1>Рецепт за брзе палачинке</h1>
        <p>Ово је најукуснији и најједноставнији рецепт за палачинке који можете да направите за само 15 минута.</p>
        <img src="palacinki.jpg" alt="Укусне палачинке">
        <p>Ако вам се допао овај рецепт, посетите наш партнерски сајт за више идеја:</p>
        <a href="https://www.google.com">Више рецепата овде</a>
</body>
</html>
```

5. Ознаке за креирање листа са тачкама или бројевима:

- – неозначена листа са тачкама,
- – нумерисана листа са бројевима 1, 2, 3...,
- – елемент листе. Увек се поставља унутар <ul> или <ol>.

Могу да се употребе за креирање листе омиљених предмета који ће се приказати на твојој веб-страници.

```
<h1>Добро дошли у моју школу</h1>
<p>Ово су моји <strong>омиљени</strong> предмети:</p>
        <ul>
        <li>Математика</li>
        <li>Веб-дизајн</li>
         </ul>
```

6. Ознака за дефинисање блока - (скраћено од division – одељење, сектор).

Ознака је несемантички блоковски елемент у HTML-у и служи као универзални празан контејнер („кутија“) за груписање других елемената ради њиховог лакшег организовања, структурирања и стилизовања помоћу CSS-а или мењања помоћу JavaScript-а. На пример, може се користити за креирање картице корисничког профила:

```
<div class="profil-karticka">
        <img src="avatar.jpg" alt="Профилна слика">
        <h2>Марко Марковски</h2>
        <p>Професија: Веб-дизајнер</p>
</div>
```

ДА ЛИ ЗНАШ?

Да ли знаш за потребу постављања „празног простора“ у тексту? Прегледач више узастопних празних места препознаје као само једно. Ако технички желиш да направиш више размака један за другим, треба да користиш специјалну ознаку за знак,   (non-breaking space).

ИСТРАЖИ

Истражи у којим случајевима се јавља потреба за дефинисањем празног простора и како се употребљава ознака   за више места.

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА:

1. Објасни разлику између логичке структуре веб-странице (онога што види корисник) и њене техничке структуре (онога што прегледач види у коду).
2. Која је улога ознаке <head>, а која ознаке <body> у HTML документу? Шта се дешава ако се текст намењен приказу стави у <head> уместо у <body>?
3. Имаш задатак да креираш онлајн мени за ресторан. Напиши HTML код који ће направити нумерисану листу (<ol>) са три јела по твом избору.
4. Размотри следећи део кода:

Добро дошли на мој блог о програмирању.

Пронађи грешку у овом коду, објасни шта није у реду према правилима угнежђивања (nesting) и напиши правилан синтаксички облик.

5. Анализирај како ће прелистач приказати текст написан у коду као:

```
<h1>Учимо HTML  заједно!</h1>
```

Која карактеристика HTML ознака за текст долази до изражаја у овом примеру?

Прегледаш једноставни HTML документ у коме је програмер поставио садржај на следећи начин:

На самом врху странице главни и најважнији наслов чланка означио је ознаком <h3>. Нешто ниже, мањи поднаслов означио је ознаком <h1>.

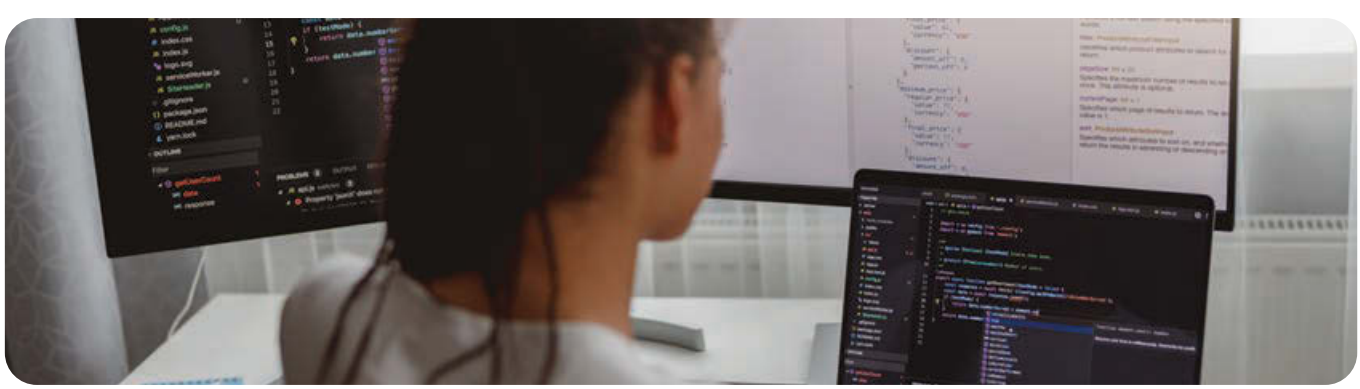
На крају је за један обичан, дугачак пасус употребио низ од пет засебних ознака <h1>, по једну за сваку реченицу, уз образложење да је желео да тај текст буде веома важан и упадљив.

Процени овакав начин коришћења основних HTML ознака. Објасни које проблеме у логичкој хијерархији ствара оваква структура и како ће утицати на претраживаче, као што је Google, који покушавају да утврде главну тему странице. Предложи шта треба изменити у коду.

Процени код и одреди какве техничке и логичке грешке су направљене:

```
<div>
<p>Погледајте наше најпознатије експонате</p>
<ul> <a href="https://muzej.mk/monaliza">
<li>Мона Лиза <img src="monaliza.jpg" alt="Слика на Мона Лиза"></li>
</a>
<li>Давидова статуа</li>
 <img src="david.jpg" alt="Слика Давидове статуе">
 <li>Звездана ноћ</li>
 </ul>
</div>
```

Предложи решење за правилно приказивање листе веб-стране музеја.



ИСТРАЖИ

Како Google роботи (Web Crawlers) скенирају хијерархију наслова (<h1> до <h6>)?

ЗАПАМТИ ШТА СИ НАУЧИО

Сваки HTML документ састоји се од два дела: заглавља (head) и тела (body). У заглављу документа наводе се информације о страници које се не приказују као њен садржај, док се садржај веб-странице пише у телу документа. Почетна ознака <html> саопштава прелистачу да је то почетак HTML документа, а завршна ознака </html> означава његов крај. Заглавље је ограничено ознакама <head> и </head>, а тело ознакама <body> и </body>. Декларација <!DOCTYPE html> користи се како би прелистач правилно приказао веб-страницу. У основни код потребно је додати и ознаку којом се дефинише начин кодирања текста. Она се поставља у заглавље: <meta charset="UTF-8"> Основне ознаке за уређивање текста приказане су у следећој табели:

Ознака (Таг)	Име / Значење	Главна функција (Улога)
<!DOCTYPE html>	Декларација документа	Саопштава прегледачу прелистачу да се користи најновија HTML5 верзија.
<html>	Коренски елемент	Обавија цео код и означава почетак и крај HTML документа.
<head>	Заглавље документа	Садржи скривене информације и мета-податке (као што је
<meta charset="UTF-8"> и <title>).	Тело документа	Садржи цео видљив део стране (текст, слике, линкови) који види корисник.
<body>	Тело документа	Садржи цео видљив део стране (текст, слике, линкови) који види корисник.
<h1> до <h6>	Наслови	Дефинишу наслове са строгом хијерархијом (од најважнијег <h1> до најмањег <h6>).
** **	Пасус (Paragraph)	Служи за организовање и приказивање обичног блок-текста.
<a>	Хипервеза (Линк)	Повезује разне веб-стране преко атрибута href (дестинација линка).
<img>	Слика	Поставља слику на страни преко извора src и садржи обавезни опис alt.
** / **	Неозначена / Означена листа	Креирају листе са тачкама ( ) или листе са бројкама ( ), користећи за сваку ставку.
** **	Одељење / Кутија (Division)	Универзални празан контејнер који служи за груписање и будуће стилизирање елемената.

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА

1. Која од следећих линија кода представља ПРАВИЛНО угнеждење (nesting) HTML ознака?

А) Учимо модеран HTML код.

Б) Учимо модеран HTML код.

В) Учимо модеран HTML код.

Г) Учимо модеран HTML код.

2. Шта ће приказати прелистач на екрану ако у VS Code напишеш следећи текст: Здра-во студенти! (са 5 празних места)?

А) „Здраво студенти!“ (тачно 5 празних места, зато што HTML чува све)

Б) „Здравостуденти!“ (спојиће речи комплетно, јер се празан простор брише)

В) „Здраво студенти!“ (само једно празно место, због карактеристика White-space Collapse)

Г) Прелистач ће пријавити синтаксичку грешку и уопште неће приказати текст.

3. Која је главна улога тага у HTML?

А) Служи за аутоматско задебљање и повећање текста на екрану.

Б) Ради као универзални празан контејнер (кутија) за груписање других елемената.

В) Користи се искључиво за уношење слика и линкова на веб-страни.

Г) Аутоматски преводи садржаје на страни на енглески језик.

4. Који од следећих тагова спада у групу „Празних елемената“ (Void Elements) који немају завршен таг и затварају се сами?

А) <p>

Б) <h1>

В) <img>

Г) <a>

5. Зашто је обавезно да напишеш линију <meta charset="UTF-8"> у <head> делу твог кода, иако VS Code сам по себи чува датотеке у том формату?

А) Да би казао прелистачу (Chrome, Safari...) тачно који стандард карактера да користи за ћирилицу како би се приказао без хијероглифа.

Б) Да би казао програму VS Code да имаш дозволу да пишеш код на српском језику.

В) Да се активирају аутоматске скраћенице (Emmet) за брже писање кода.

Г) Тиме се дефинише главни наслов који се појављује на највишем табу прелистача.

6. Замисли да правиш веб-страницу и желиш да поставиш слику логотипа своје школе. Напиши комплетну HTML линију за уметање слике и укратко објасни зашто је важно навести атрибут alt у тој ознаци. Наведи најмање један разлог.

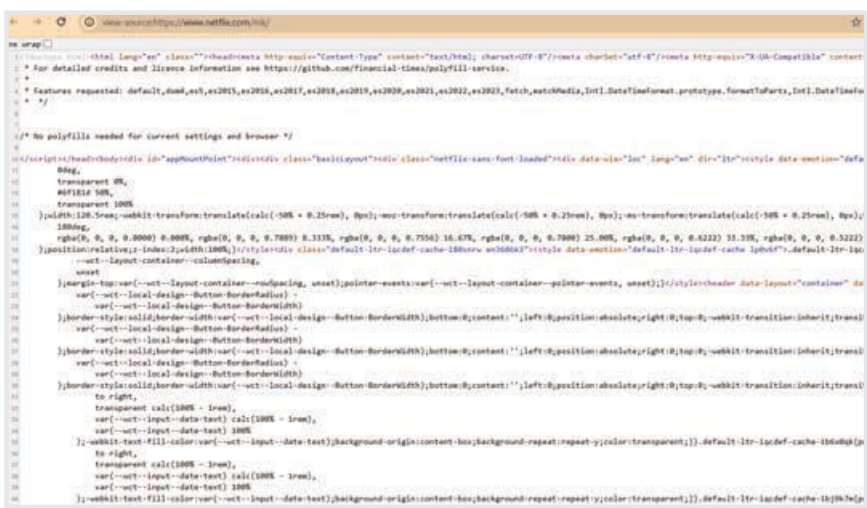
ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Савремени HTML не служи само за приказивање текста и слика. Он подржава уграђене напредне технологије (API-је) које су некада захтевале сложене програме, а данас се активирају помоћу неколико линија кода:

1. Геолокација (Geolocation): Помоћу HTML-а и мало JavaScript-а можеш да затражиш дозволу корисника како би сајт утврдио његову тачну локацију на мапи.
2. Локално складиштење (Local Storage): HTML5 омогућава сајту да податке чува непосредно у прелистачу корисника, као што су избор тамног режима (Dark Mode) или производи у корпи. Чак и ако корисник искључи рачунар и врати се следећег дана, сајт ће запамтити његов избор.
3. Аудио и видео стримовање: Ознаке <audio> и <video> имају напредне контроле које омогућавају да направиш сопствени плејер попут YouTube-а или Spotify-а непосредно у прелистачу, без спољних додатака.

5.2.2 УРЕЂИВАЧИ ЗА ПИСАЊЕ HTML КОДА

Веб-страница је документ који садржи HTML код и има наставак .html. Отвори своју омиљену веб-страницу, кликни десним тастером било где и из менија изабери View page source. Приказаће се HTML код странице који изгледа као текстуални документ и може да се пише у текстуалном едитору као што су Notepad или Notepad++. Не препоручује се писање кода у програму за обраду текста (Microsoft Word, Apple Mac-Pages и др.) јер они праве датотеке које прегледач не може правилно да прочита. HTML код може да се пише и у другим едиторима, као што је бесплатни Visual Studio Code који се инсталира, или у онлајн едиторима као што су CodePen.io, OneCompiler и други. У овом уџбенику за изучавање HTML-а користиће се Visual Studio Code (VS Code), едитор отвореног кода и индустријски стандард за веб-развој. При писању кода нуди паметне предлоге, аутоматски приказује доступне датотеке и омогућава приказ веб-странице у прегледачу унутар самог VS Code-а.



Слика 1: Изглед HTML кода

5.2.3. ИНСТАЛАЦИЈА НА HTML ЕДИТОР VISUAL STUDIO CODE

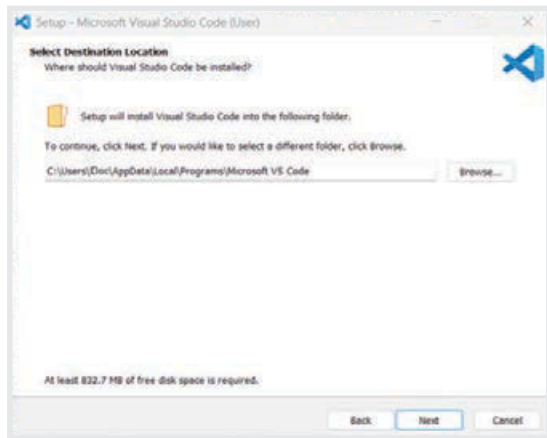
Инсталирање програма Visual Studio Code (VS Code) веома је брз и једноставан поступак. Отвори званичну веб-страницу програма Visual Studio Code и изабери верзију која одговара твом оперативном систему. Кликни на потребну верзију и сачекај неколико секунди да се инсталациона датотека, са наставком .exe, преузме на твој компјутер.

1. Пронађи преузету датотеку, на пример VSCodeUserSetup-x64.exe, на свом компјутеру и кликни на њу двапут (double-click) да би започео поступак инсталације.
2. Покренуће се чаробњак који ће те водити кроз поступак. Најпре се појављује прозор са уговором о лиценци (License Agreement

Слика 2: Прихватање услова уговора

3. Прочитај уговор, изабери опцију „I accept the agreement“, а затим кликни на дугме Next
4. Затим одабери локацију у компјутеру где ћеш сачувати датотеку.

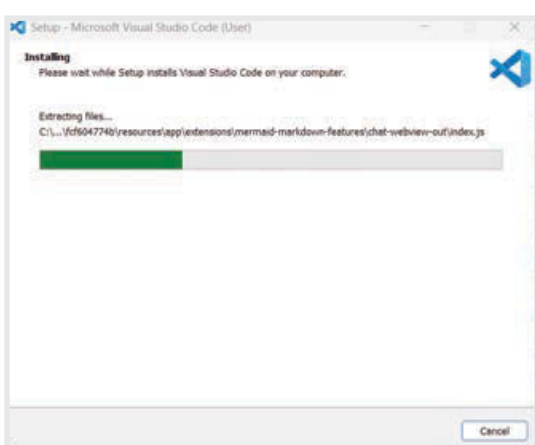
Слика 3: Одређивање локације за инсталацију



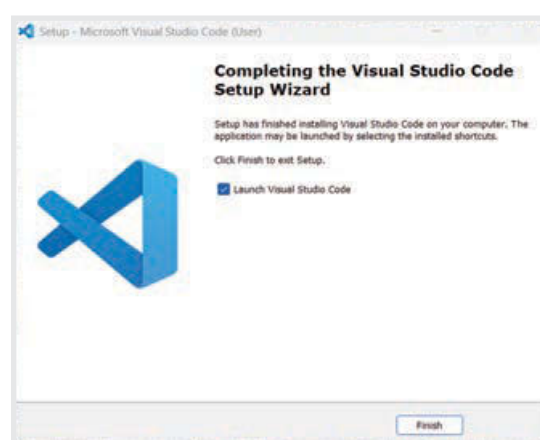
Када се отвори прозор са насловом "Select Additional Tasks“ можеш да изабереш следеће опције:

- Create a desktop icon: Да добијеш пречицу (икону) за VS Code директно на десктопу ради брзог покретања.
- Add “Open with Code” to Windows Explorer... (Постоје две овакве опције, изабери обе). Ово је корисно јер ћеш касније, када будеш имао фасциклу са HTML пројектима, моћи десним кликом на фасциклу да изабереш „Open with Code“, чиме се цео пројекат одмах отвара у едитору.
- Register Code as an editor for supported file types: Ово омогућава да се двоструким кликом на било коју .html или .css датотеку она аутоматски отвори у VS Code-у.
- Add to PATH: Ова опција је подразумевано означена и омогућава да програм правилно ради са системом.

Када изабереш ова поља, кликни **Next** да започне процес инсталације.

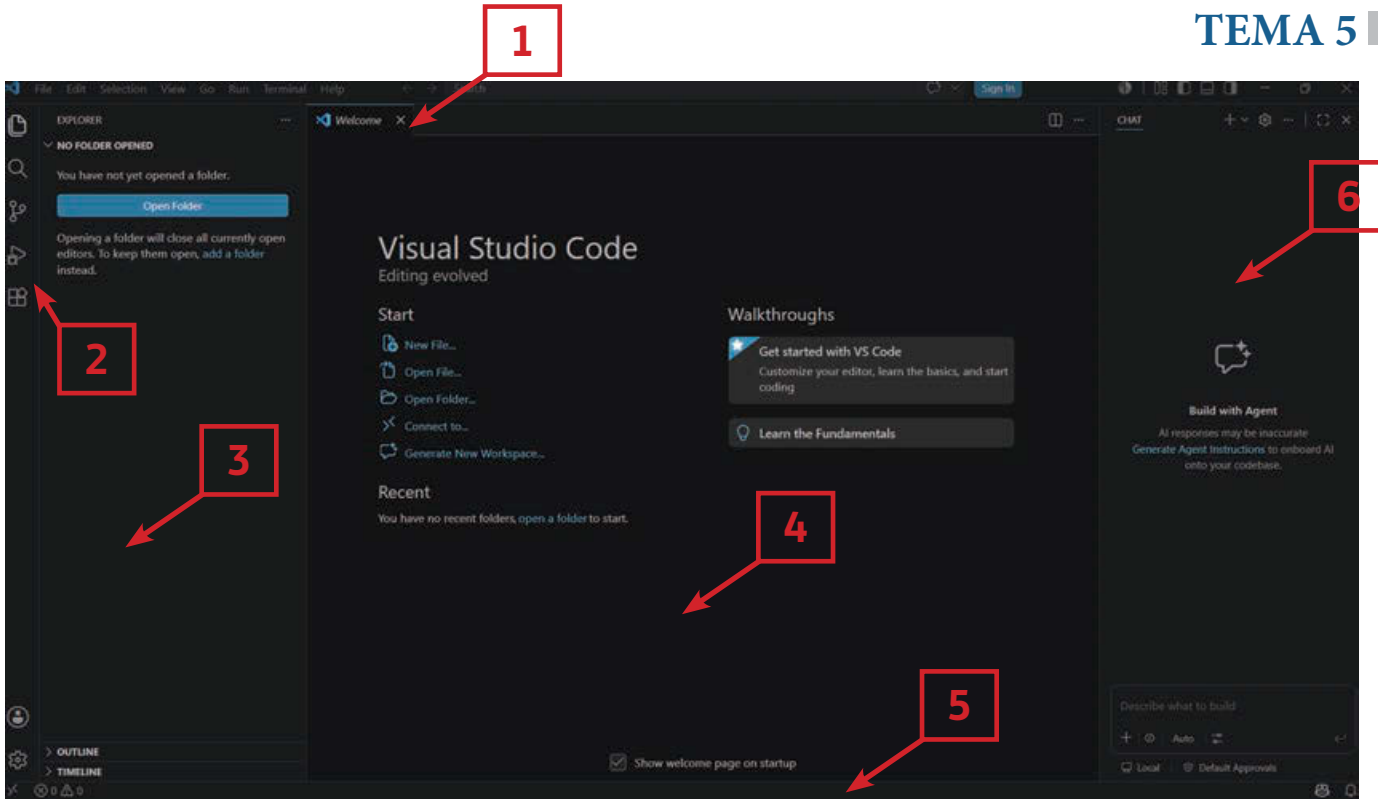


Слика 4:Процес инсталације



Слика 5:Завршена инсталација VS Code-а

5. Инсталација траје око 10 до 20 секунди, док се зелена трака не попуни до краја. Кликни на дугме Finish и покренуће се почетни радни прозор едитора.



Слика 6: Почетни радни екран у VS Code-у прозор

1. Почетни таб: „Welcome“ (Добро дошли)

На самој средини екрана отвориће се велики таб Welcome. То је твоја почетна станица у којој се налазе пречице за:

- **New File** – креирање новог, празног документа.
- **Open Folder** – отварање фасцикле са компјутера (најважнија опција за програмере).
- **Clone Git Repository** – за напредне кориснике који преузимају код са интернета.

2. Трака са алаткама на крајњој левој ивици (Activity Bar)

То је уска вертикална трака са иконама на самој левој ивици прозора. Ове иконе су твоји главни „режими рада“:

- **Explorer (икона са две фасцикле):** Ово ћеш користити 95% времена. Када кликнеш овде, отвара се бочни прозор у ком видиш своје фасцикле и датотеке.
- **Search (лупа):**За претраживање и замену речи у свим датотекама одједном.
- **Source Control (грана/чвор):**Намењено за Git и верзионисање кода.
- **Run and Debug (бубица са троуглом):** За тестирање и проналажење грешака у коду.
- **Extensions (четири коцке, једна издвојена):** Твоја „продавница“ бесплатних додатака. Одавде ћемо касније инсталирати корисне алатке за лакше писање кода.

3. Бочни прозор (Sidebar)

Овај простор се налази одмах десно од траке са алаткама. Ако ти је укључена опција *Explorer*, овде ћеш видети структуру свог пројекта – које HTML датотеке имаш, да ли постоје фасцикле за слике и слично. У почетку, док није отворена фасцикла, овде стоји велико плаво дугме **"Open Folder"**.

4. Главни простор за уређивање (Editor)

Ово је највећи, централни део екрана. Када кликнеш на неку датотеку (на пример index.html), њен садржај отвориће се управо овде. То је твоје „платно“ на ком ћеш писати и мењати HTML код. и менуваш HTML кодот. Можеш да отвориш више датотека истовремено и оне ће бити поређане у табовима на врху, као у Chrome-у.

5. ДОЊА СТАТУСНА ТРАКА (STATUS BAR)

На доњој ивици прозора налази се танка трака (најчешће плава или љубичаста). Она даје тренутне информације о датотеци у којој радиш: у ком реду се налазиш, који програмски језик је активан (доле десно треба да пише HTML) и које је кодирање (подразумевано UTF-8). Прозор је направљен тако да не одвлачи пажњу: лево је организација датотека, а у центру сам код.

6. Радни простор за рад са ВИ асистентом.

5.2.4. РАД СА ЕДИТОРОМ

Поступак за почетак рада у VS Code-у прилично је једноставан и брз. Када први пут отвориш програм, најважније је да запамтиш да **HTML датотеке не могу да „висе“ саме у ваздуху** – увек треба уредно да буду сачуване у сопственој фасцикли (folder) на компјутеру. Испод су кораци објашњени један по један:

Корак 1: Направи фасциклу за пројекат

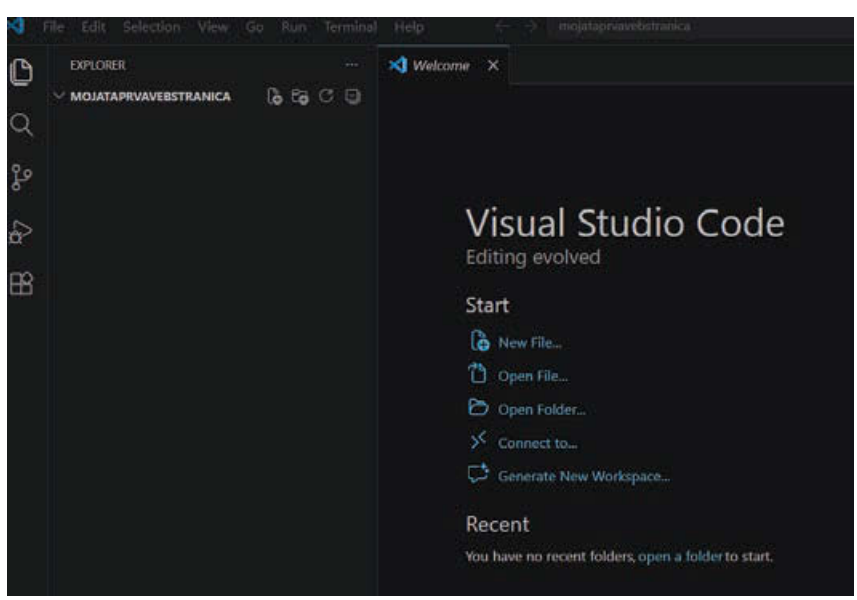
Пре него што уопште отвориш **Visual Studio Code (VS C)**, направи нову, празну фасциклу на компјутеру. Назови је једноставно, на пример: moj-prvi-projekat (најбоље је користити мала латинична слова без размака и назив који асоцира на садржај веб-странице).

Корак 2: Отвори фасциклу у VS Code-у

Сада отвори VS Code. Постоје два начина да унесеш фасциклу:

1. Преко менија **File -> Open Folder**... и изабери фасциклу коју си направио у Кораку 1.
2. Или једноставно **превуци и испусти (Drag, Drop)** фасциклу мишем директно у празан простор VS Code-а.

Зашто је ово важно? На овај начин VS Code зна да све што од сада радиш припада том конкретном пројекту. Лева страна екрана постаје твој радни простор (Explorer).



Слика 7: Креирана фасцикла у Visual Studio Code-у

Корак 3: Креирај нову HTML датотеку

На левој страни, у делу где пише назив твоје фасцикле, кликни на малу икону за **New File** (изгледа као лист папира са знаком плус).

Напиши назив датотеке: **index.html** и притисни Enter.

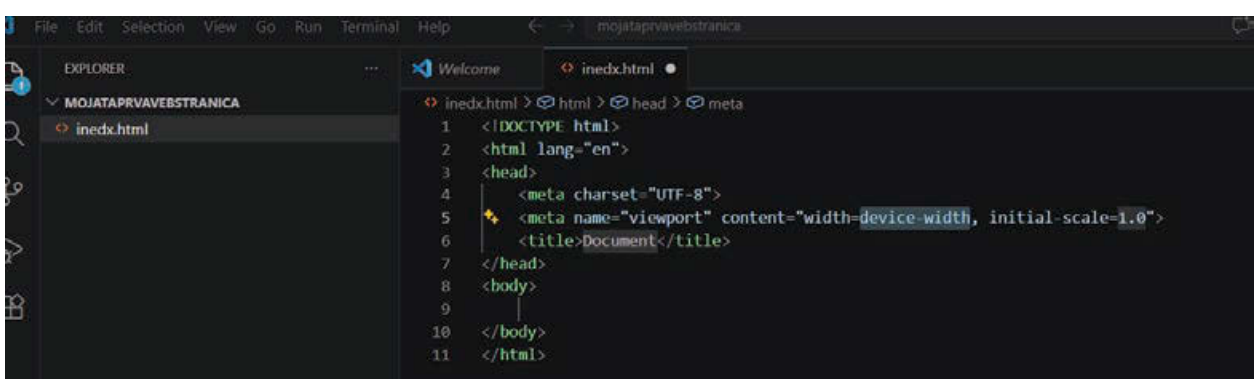
Слика 8: Именована датотека у Visual Studio Code-у

Важно: Назив треба да се завршава са .html да би компјутер препознао да је то веб-страница, а не обичан текстуални документ. Назив index је светски стандард за главну (почетну) страницу сваког сајта.

Корак 4: Активирај почетни скелет

Сада када је датотека отворена и празна:

1. У првој линији напиши само један узвичник: **!**
2. Појавиће се мали мени (Emmet пречица). Притисни **Enter** или **Tab**.
3. VS Code ће сам генерисати цео обавезни почетни код (скелет са <html>, <head>, <body> и ознаком за UTF-8).



Слика 9: Креиран основни скелет веб-странице

Корак 5: Напиши свој први код

Прегледач на екрану приказује само оно што се налази између почетне <body> и завршне </body> ознаке.

Постави курсор испод прве ознаке <body> и напиши:

```
<body>
  <h1>Здраво свете!</h1>
  <p>Ово је моја прва веб-страница написана у VS Code.</p>
</body>
```

Корак 6: Сачувај код (Save)

Горе на табу где пише index.html приметићеш мали бели кружић. То значи: „Написао си нов код који још није сачуван!“

- Изабери пречицу на тастатури:**Ctrl + S** (за Windows) или **Cmd + S** (за Mac).
- Бели кружић ће нестати и претворити се у знак икс (x). Сада је твој код безбедно сачуван на хард диску.

Тиме је твој први код завршен и спреман за преглед.

Постоји више начина за преглед веб-странице, а у наставку су објашњена три.

Начин 1: Отварање директно из фасцикле. То је најједноставнији начин који ради на сваком компјутеру и не захтева интернет нити инсталирање додатних додатака.

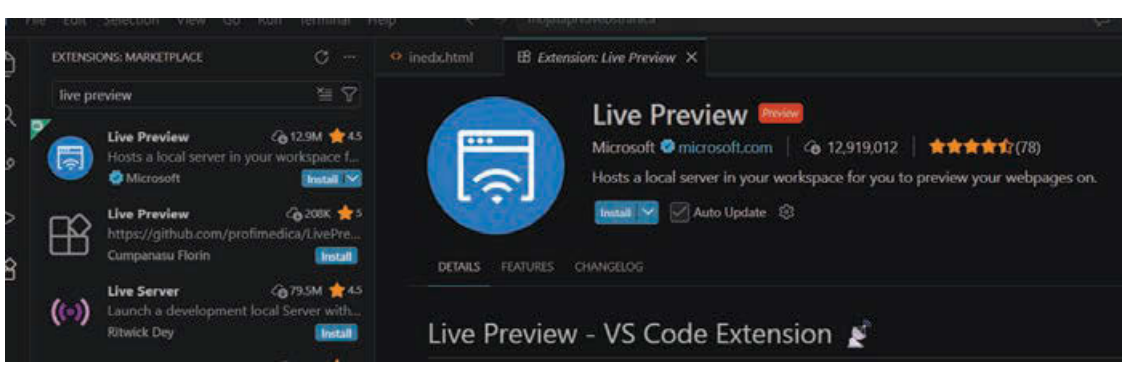
- Отвори фасциклу на компјутеру у којој си сачувао фајл index.html.
- Два пута кликни на датотеку index.html.
- Компјутер ће аутоматски отворити датотеку у подразумеваном прегледачу (Chrome, Edge, Firefox).

Важна напомена за овај начин:Сваки пут када нешто измениш у коду у VS Code-у, прво треба да сачуваш промену помоћу **Ctrl + S** и затим ручно отвориш прегледач и кликнеш на дугме **Refresh** или изабереш **F5** на тастатури да би видео промене.

Начин 2: Помоћу додатка „Live Preview“ (преглед директно у VS Code)

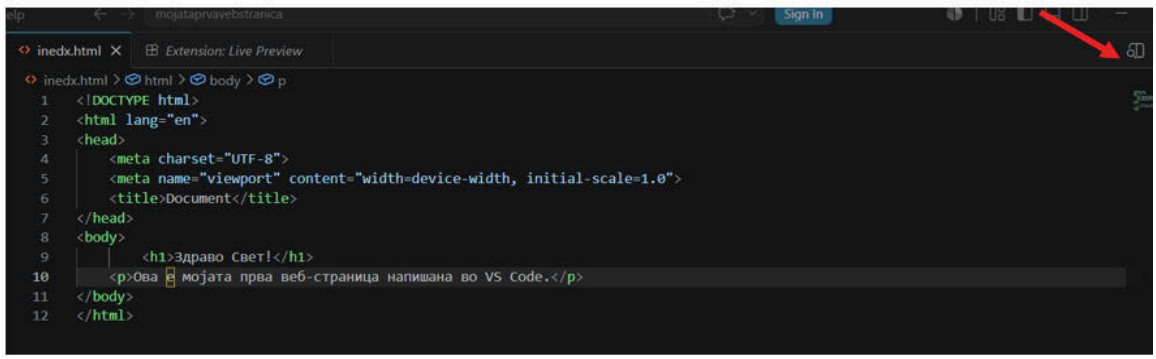
Ако немаш два монитора или не желиш да делиш екран на две половине са Chrome-ом, Microsoft је направио додатак који отвара мини-прегледач директно у самом програму.

1. Изабери **Extensions** (четири коцке лево) и потражи **Live Preview** (званични додатак компаније Microsoft).
2. Кликни **Install**.



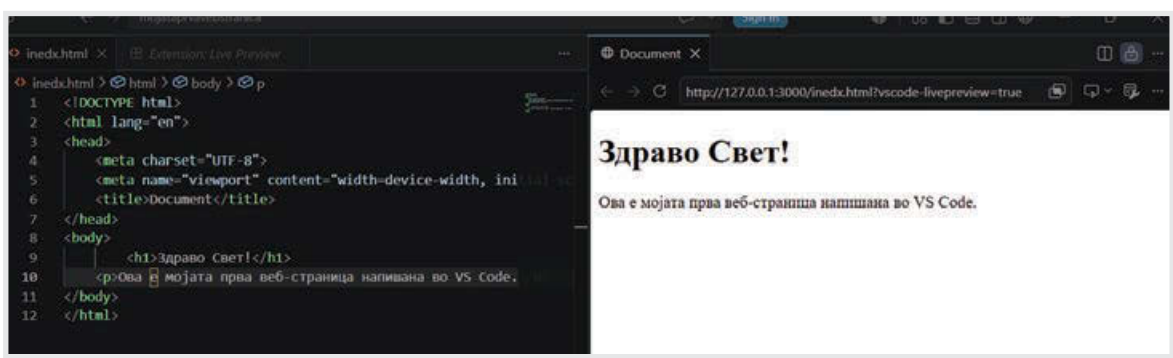
Слика 10: Инсталација додатка Live Preview

3. Када се инсталација заврши, дугме ће се променити у зупчаник (што значи да је инсталиран). Од тог тренутка додатак је трајно уграђен у твој VS Code и спреман за коришћење једним кликом на икону малог екрана у горњем десном углу.
4. Када се вратиш у свој index.html код, у горњем десном углу екрана приметићеш нову икону која личи на **мали компјутерски екран са лупом**.



Слика 11: Инсталиран Live Preview

5. Кликни на ту икону и екран VS Code-а поделиће се на два дела: лево ће бити код, а десно ће се одмах појавити твоја страница уживо. Свако чување одмах се приказује тамо.



Слика 12: Радни простор и приказ веб-странице

Начин 3: Преко ВИ асистента на десној страни

Пошто већ имаш ВИ прозор на десној страни, можеш да искористиш и његове могућности за брз преглед.

1. Селектуј цео свој HTML код.
2. Питај ВИ асистента у разговору: „Како ће изгледати овај код када се рендерује?“ или „Show me a preview of this HTML“.
3. Многи савремени ВИ агенти (као Copilot или Cursor) имају уграђено дугме **„Preview“** директно у свом прозору, чијим кликом приказују визуелни преглед тренутног дизајна странице.

За почетно учење најбољи је Начин 2 (Live Preview), јер је све на једном месту и нема потребе да помераш миш изван VS Code.

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА:

1. Која комбинација тастера је светски стандард у VS Code-у за чување промена у коду?

- А) Ctrl + C
- Б) Ctrl + S
- В) Ctrl + V
- Г) Ctrl + Z

2. Шта означава икона са четири коцке (где је једна одвојена) на крајњој левој вертикалној траци у VS Code-у?

- А) Мени за претраживање речи кроз код
- Б) Мени за рад са фасциклама и фајловима (Explorer)
- В) Продавница за инсталирање додатака (Extensions), као *Live Preview*
- Г) Део у ком се отвара вештачка интелигенција за разговор

3. Уместо додатка за приказ, веб-страницу отвараш двоструким кликом директно из фасцикле на компјутеру у прегледачу. Када допишеш нови HTML код у VS Code-у и сачуваш га са Ctrl + S, који је следећи обавезан корак да би видео промену у Chrome-у?

- А) Треба потпуно затворити прегледач и поново га отворити двоструким кликом из фасцикле.
- Б) Треба ручно кликнути на дугме Refresh (Освежи / Учитај поново) у прегледачу или притиснути F5 на тастатури.
- В) Не треба ништа, страница ће се сама променити истог тренутка када притиснеш Ctrl + S.
- Г) Треба избрисати стару HTML датотеку из фасцикле и направити нову.

4. Уместо да ручно пишеш цео почетни скелет HTML документа, који знак (пречицу) треба да напишеш у празном фајлу да би VS Code (преко Emmet-а) сам генерисао тај код једним притиском на Enter?

- А) Упитник (?)
- Б) Тараба (#)
- В) Узвичник (!)
- Г) Коса црта (/)

5. Када радиш са уграђеним ВИ асистентом (на десној страни екрана), који је најефикаснији начин да агенту кажеш да објасни или исправи само једну одређену линију твог кода?

- А) Треба ручно преписати цео код у прозор за разговор.
- Б) Довољно је мишем селектовати (означити) ту линију кода на левој страни, а затим поставити питање у десном ВИ прозору.
- В) Треба сачувати фајл у посебном формату .txt и послати га асистенту.
- Г) ВИ асистенти не могу да виде шта пишеш, па мораш да наведеш тачну линију (нпр. „Линија 14“).

6. Који је тачан редослед корака за започињање новог пројекта у празном VS Code-у, тако да програм одмах почне да помаже (боји тагове и даје предлоге)?

- А) Прво пишеш код, затим правиш нову датотеку, па је на крају стављаш у фасциклу.
- Б) Прво сачуваш празан текст на десктопу, затим га преименујеш у фасциклу, па отвориш Chrome.
- В) Прво направиш нову датотеку, затим напишеш узвичник, а тек на крају размишљаш где ћеш је сачувати на компјутеру.
- Г) Прво отвориш фасциклу пројекта (**Open Folder**), у њој креираш нову датотеку (**New File**) и одмах је сачуваш са наставком **.html**.
- Д) Редослед уопште није важан, VS Code сам зна шта желиш да пишеш још пре него што укључиш рачунар.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Ако често пишеш исту структуру (на пример, скелет картице, табелу са специфичним класама или мени), нема потребе да је стално преписујеш или радиш Copy/Paste из старих пројеката. Можеш да направиш **сопствену Emmet пречицу**. Кликни на зупчаник доле лево (Settings) и изабери **User Snippets**. Затим изабери **html.json** и тамо можеш да дефинишеш своју пречицу. На пример, ако дефинишеш пречицу mojakarticka, чим је напишеш у празном HTML фајлу и притиснеш Tab, VS Code ће аутоматски учитати цео твој унапред дефинисани код са насловима, класама и сликама. Такође, када HTML датотека има 500 или 1000 линија, скроловање горе-доле постаје заморно. Напредни програмери користе паметну навигацију.

1. **Breadcrumbs (мрвице хлеба):**На врху изнад кода увек видиш путању (нпр. src > index.html > body > div.container > table). Можеш да кликнеш на било коју ознаку у тој путањи и одмах „скочиш“ директно на њу, без скроловања.
2. **Outline мени:**У доњем левом углу (испод фасцикли) налази се мени Outline. Он приказује HTML документ као породично стабло. Ако кликнеш на неки таг у том стаблу, едитор те одмах води до њега.

5.2.5. ПРИМЕНА НА HTML ОЗНАКИ

У HTML језику атрибути се користе за додавање додатних карактеристика ознакама. Ако су саме ознаке скелет веб-странице (показују где је наслов, где пасус), атрибути им дају додатне информације. Увек се пишу унутар почетне ознаке (никада у завршној). Пишу се у облику: ime_atributa="vrednost", односно имају име и вредност. Име атрибута пише се малим словима, а вредност увек треба да буде између наводника („“) или полунаводника (‘). Чешће се користе наводници. Полунаводници се користе када вредност садржи наводнике, на пример ime='булевар „8. септембра“ '.

Такође, да би ознака за линк (<a>) знала куда да нас „одведе“, користимо атрибут href.

```
<a href="https://www.google.com">Иди на Google</a>
```

Користе се и други атрибути за хипервезу,**target="_blank"** – отвара линк у **новом табу**, тако да корисник не напушта твоју страницу, а **title** – приказује мали текстуални савет (tooltip) када задржиш миш изнад линка.

```
<a href="https://sr.wikipedia.org" title="Иди на Википедију">Посети Википедију</a>
```

(једноставна хипервеза)

```
<a href="https://www.google.com" target="_blank">Отвори Google у новом табу</a>
```

(хипервеза која се отвара у новом табу)

Ознака за слику (<img>) специфична је јер је самостална (нема завршну ознаку). За приказ слике користи атрибуте src (извор слике) и alt (алтернативни текст):

```
<img src="slika.jpg" alt="Опис слике">
```

Постоје и атрибути за ширину и величину слике,**width и height** помоћу којих се величина слике задаје у пикселима.

```
<img src="avatar-korisnik.png" alt="Профилна слика корисника" width="150" height="150">
```

Један од најчешћих елемената на интернету је када желиш да **клик на саму слику одведе на неки линк**(као када кликнеш на логотип да се вратиш на почетну страницу).

Да би то урадио, једноставно угнежђујеш ознаку <img> унутар, између почетне и завршне ознаке <a>.

```
<a href="https://www.youtube.com" target="_blank">
  <img src="youtube-logo.png" alt="Логотип на YouTube" width="200">
</a>
```

Све ове ознаке могу да се комбинују у један код и да се креира занимљива страница. На пример, следећи код креира страницу са сликом и хипервезом која води до више информација о слици:

```
<h1>Слика малог пса</h1>

<img src="https://images.unsplash.com/photo-1543466835-00a7907e9de1" alt="Пас који гледа нагоре" width="300">

<br> <a href="https://sr.wikipedia.org/wiki/Пас" target="_blank">Прочитај више о псима на Википедији</a>
```

За креирање табела у HTML-у користе се посебни атрибути за спајање ћелија. Када правимо табелу, користимо једну главну ознаку у коју се угнежђује неколико помоћних ознака за редове и колоне:

Главне ознаке за табелу:

- **<table>** – започиње и завршава целу табелу.
- **<tr>** (Table Row) – прави нов **хоризонтални ред**.
- **<td>** (Table Data) – прави обичну **ћелију (колону)** у том реду у којој се налазе подаци.
- **<th>** (Table Header) – прави **насловну ћелију** (текст аутоматски постаје подебљан и центриран).

Ако правимо распоред часова и један предмет се одржава два часа, треба да спојимо две колоне или два реда. За то користимо следећа два атрибута који се пишу искључиво унутар <td> или <th> ознака:

Атрибут colspan (спајање колона)

Овај атрибут говори пољу да се хоризонтално „растегне“ преко више колона одједном.

- **Пример:** Ако имаш блок-часове информатике, уместо два посебна поља правиш једно овакво:

```
<td colspan="2">Информатика (блок-час)</td>
```

Атрибут rowspan (спајање редова)

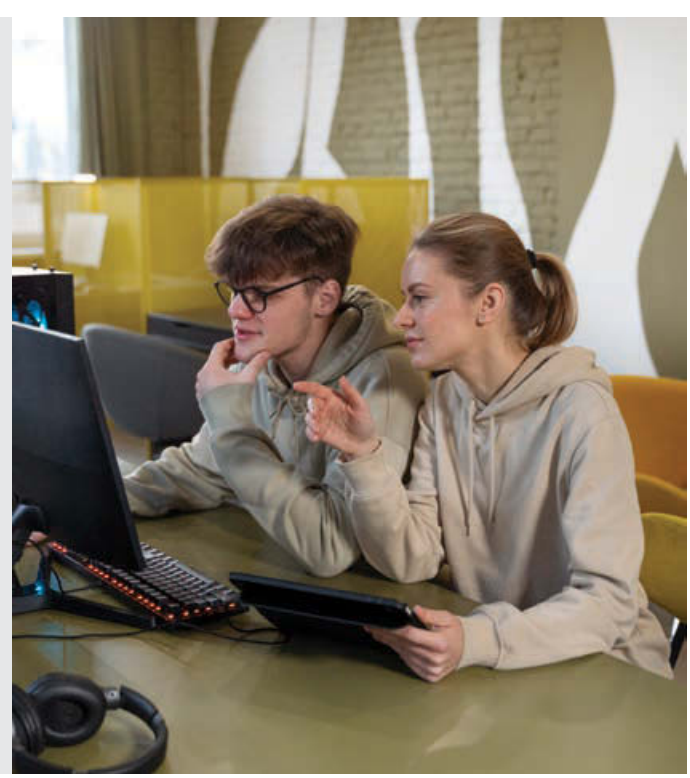
Овај атрибут „растеже“ поље вертикално, односно надоле кроз редове.

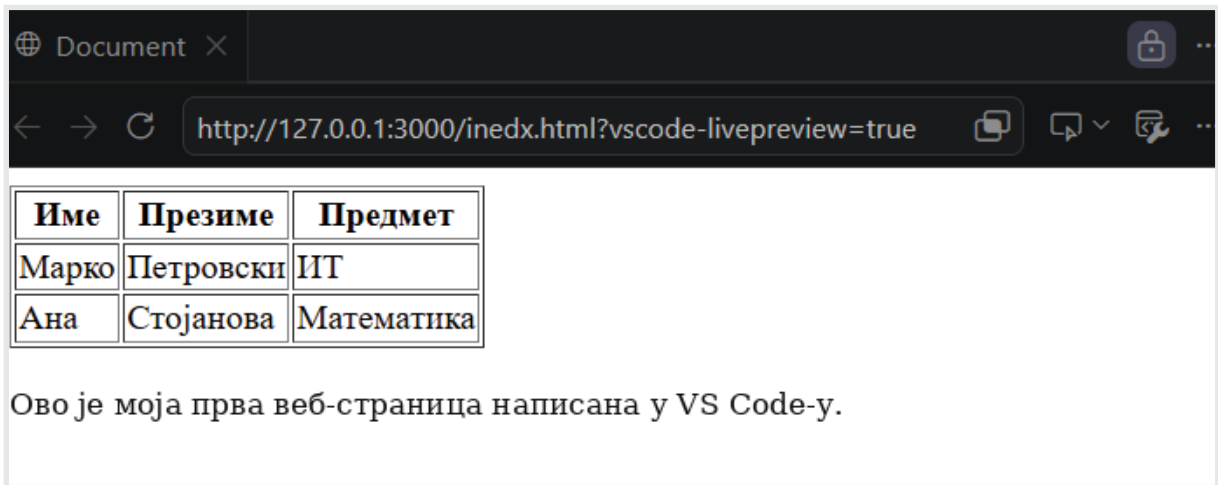
- **Пример:** Ако сваког дана у исто време имаш „Велики одмор“, можеш да спојиш редове за дане:

```
<td rowspan="3">Велики одмор</td>
```

Комплетан код за повезивање свих ових ознака и њихових атрибута:

```
<table border="1"> <tr>
<th>Име</th>
<th>Презиме</th>
 <th>Предмет</th>
</tr>
<tr>
<td>Марко</td>
<td>Петровски</td>
<td>ИТ</td>
</tr>
<tr>
<td>Ана</td>
<td>Стојанова</td>
<td>Математика</td>
</tr>
</table>
```





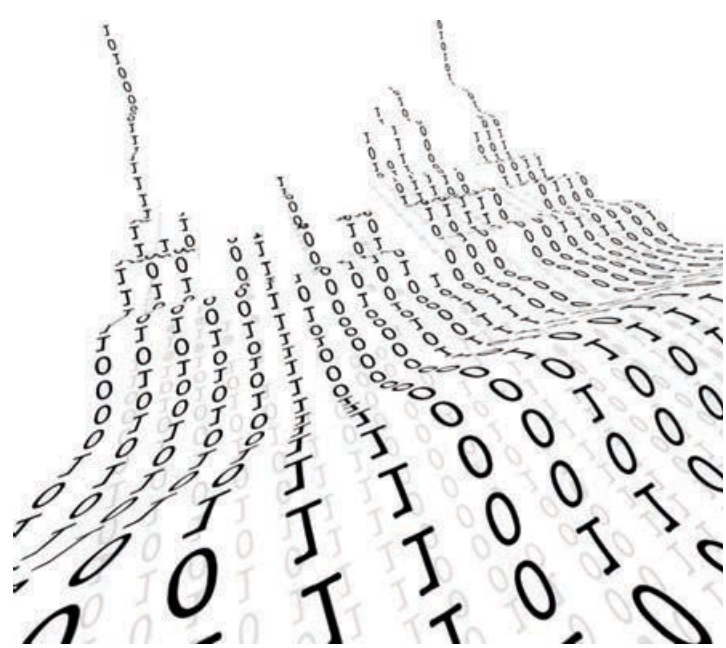
Слика 13: Распоред креиран помоћу HTML-а

Знање о креирању табела можемо искористити за прављење форме за пријаву на курс која ће садржати табелу, уносе и спајање. Овај пример је још напреднији јер користи табелу без оквира (невидљиву мрежу) да савршено поравна поља за унос, а на крају користи colspan да прошири дугме преко обе колоне.

<h2>Пријава за курс HTML ознака</h2>

Молимо попуните обавезна поља да бисте резервисали своје место.

```
<table>
 <tr>
  <td><b>Име и презиме:</b></td>
  <td><input type="text" placeholder="Унесите своје име..." required></td>
 </tr>

 <tr>
  <td><b>Е-пошта:</b></td>
  <td><input type="email" placeholder="пример@е-пошта.ком" required></td>
 </tr>

 <tr>
  <td><b>Датум рођења:</b></td>
  <td><input type="date"></td>
 </tr>

 <tr>
  <td colspan="2" style="text-align: center;">
   <br>
   <input type="submit" value="Пошаљи пријаву">
  </td>
 </tr>
</table>
```



У табелама испод приказане су све ознаке и атрибути са њиховим описима који се користе у HTML-у.

Ознака	Шта означава?	Опис / Како се користи?	Има ли завршну ознаку?
<h1> до <h6>	Наслови	<h1> је највећи главни наслов, <h6> је најмањи поднаслов.	Да (нпр. </h1>)
** **	Пасус	Користи се за писање обичног текста.	Да ( )
<a>	Хиперлинк	Прави линк на који може да се кликне.	Да ()
<img>	Слика	Умеће слику на страницу.	**Не** (самостална је)
<table>	Табела	Започиње и завршава целу табелу.	Да (</table>)
<tr>	Ред у табели	Прави нов хоризонтални ред унутар табеле.	Да (</tr>)
<td>	Поље (ћелија)	Обично поље са подацима унутар реда.	Да (</td>)
<th>	Насловно поље	Поље за наслов колоне (текст је подебљан).	Да (</th>)
** **	Нови ред	Спушта текст или елемент у нови ред.	**Не** (самостална је)

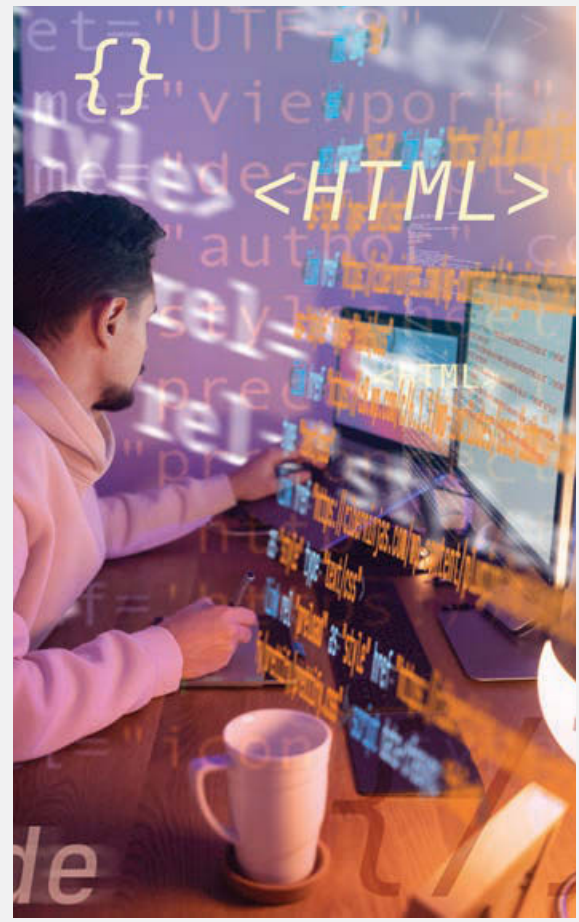
Табела 2: Ознаке у HTML-у

Атрибут	На коју ознаку се поставља?	Шта контролише? (Објашњење)	Пример како се пише
href	На ознаку за линк (<a>)	Говори **куда** (на који линк) да нас одведе када кликнемо.	href="https://google.com"
target	На ознаку за линк (<a>)	Вредност "_blank" говори да се линк отвори у **новом табу**.	target="_blank"
src	На ознаку за слику (<img>)	Наводи тачну путању или линк **одакле** се узима слика.	src="kuche.jpg"
alt	На ознаку за слику (<img>)	**Алтернативни текст** који описује слику речима.	alt="Црно штене"
width	На слику (<img>) или видео	Одређује **ширине** величину елемента у пикселима.	width="300"
colspan	На поља у табели (<td> / <th>)	Спаја више **колоне** колона хоризонтално (нпр. за двочас).	colspan="2"
rowspan	На поља у табели (<td> / <th>)	Спаја више **редова** вертикално у табели.	rowspan="2"
border	На ознаку за табелу (<table>)	Додаје видљив **оквир** (линије) око табеле.	border="1"
title	На било коју ознаку	Приказује мали текстуални облачић када задржиш миш изнад.	title="Више информација"

Веома једноставно може да се креира презентација производа или књиге. Користе се наслов, пасус са описом, слика корице, табела са техничким детаљима о књизи и линк за куповину који се отвара у новом табу.

```
<h1>Књига: Дигитални свет</h1>
<p title="Кратак опис садржаја">Ово је узбудљива књига о почецима компјутерске технологије и развоју интернета.</p>
<img src="https://images.unsplash.com/photo-1543002588-bfa74002ed7e" alt="Корица књиге о технологији"гија" width="250">
```

```
<h3>Технички детаљи</h3>
<table border="1">
 <tr>
  <th>Карактеристика</th>
  <th>Информација</th>
 </tr>
 <tr>
  <td><b>Аутор</b></td>
  <td>Марко Петровић</td>
 </tr>
 <tr>
  <td><b>Језик</b></td>
  <td>Српски</td>
 </tr>
 <tr>
  <td><b>Број страна</b></td>
  <td>240 страна</td>
 </tr>
</table>

<br>
<a href="https://www.amazon.com" target="_blank" title="Купи књигу онлајн">Кликни овде да наручиш књигу</a>
```



ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА:

1. Где се УВЕК пишу HTML атрибути?
   - А) На самом крају документа, после ознаке </html>
   - Б) Унутар почетне ознаке (opening tag)
   - В) Унутар завршне ознаке (closing tag)
   - Г) Између почетне и завршне ознаке, заједно са текстом
2. Која од следећих ознака за табелу користи се за креирање новог хоризонталног реда?
   - А) <table>
   - Б) <td>
   - В) <tr>
   - Г) <th>
3. Својим речима објасни разлику између HTML ознаке (tag) и HTML атрибута (attribute). Како они сарађују да би нешто приказали на екрану?

4. Ако слици задамо атрибут width="300", а изоставимо атрибут height, шта ће се догодити са висином слике у прегледачу?

- А) Слика се уопште неће приказати јер недостаје висина.
- Б) Висина ће постати 0 пиксела и слика ће бити невидљива.
- В) Прегледач ће сам пропорционално прилагодити висину како се слика не би изобличила слика.
- Г) Слика ће се развући вертикално преко целог екрана.

5. Желиш да поставиш слику (logotip.png) која ће, када корисник кликне на њу, отворити веб-страницу https://www.google.com у потпуно новом табу. Напиши тачан HTML код за ту намену.

6. Који од следећих кодова правилно приказује линк са додатним описом када се мишем пређе преко њега?

- А) Иди на сајт
- Б) Иди на сајт
- В) Иди на сајт
- Г) <link href="https://sajtedu.mk" title="Школа">Иди на сајт</link>

7. Размотри следећи HTML код за табелу:

```HTML
<table border="1">
  <tr>
    <td>Српски</td>
    <td colspan="2">Математика (двочас)</td>
  </tr>
  <tr>
    <td>Енглески</td>
    <td>ИТ</td>
    <td>Ликовно</td>
  </tr>
</table>
```

Анализирај код и објасни зашто ће ова табела визуелно бити изобличена у прегледачу. Где је грешка у бројању поља?

8. Замисли да правиш веб-страницу за особе са оштећеним видом које користе софтверске „читаче екрана" (Screen Readers). Који од следећа два корака је кључан за њих и зашто?

- А) Увек користити атрибут target="_blank" на линковима.
- Б) Увек написати прецизан и описан текст у атрибуту alt ознаке <img>.
- В) Табеле увек треба да имају border="1".
- Г) Сви наслови треба да буду искључиво <h1>.

9. Дизајнирај HTML код за најједноставнију онлајн визит-карту (профил). Код треба да садржи: твој омиљени наслов, један пасус о теби, твоју слику и један линк до омиљеног сајта. Води рачуна о правилном отварању и затварању ознака и тачној синтакси атрибута.

10. Замисли да треба да направиш распоред за сутрашњи дан у школи, где су 3. и 4. час блок-часови (двочас) истог предмета. Напиши само HTML код за тај конкретни ред (<tr>) табеле, користећи одговарајући атрибут за спајање.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Ознака <a> са атрибутом href не служи само за прелазак са једне веб-странице на другу. Она може да комуницира са оперативним системом компјутера или телефона корисника.

Веза за директно слање е-поште (mailto:): Ако желиш да се кориснику након клика на линк одмах отвори програм за е-пошту (као Outlook или Gmail) са већ уписаном адресом, користиш следеће:

Пошаљи нам е-пошту

Веза за директно позивање (tel:): Ово је веома корисно за кориснике који сајт гледају са мобилног телефона. Када кликну на линк, телефон аутоматски почиње да позива број:

Позови одмах

Ако имаш веома дугачку веб-страницу и желиш да корисник једним кликом „скочи" до контакта при дну, користиш комбинацију href и атрибута id:

Скочи до контакта

<h2 id="kontakt-sekcija">Овде су наши подаци</h2>

Експериментиши са паметним сликама помоћу атрибута loading="lazy", са ознакама за напредни текст који рачунар „разуме", атрибутом download и другим.

5.2.6 CSS – СТИЛОВИ ВЕБ-СТРАНИЦЕ

Док је HTML одговоран за структуру (скелет) веб-странице, CSS (Cascading Style Sheets) је језик одговоран за њен изглед и дизајн. Без CSS-а све веб-странице би изгледале као обичан црно-бели текст у Notepad-у са плавим линковима.

CSS описује како HTML ознаке треба да се прикажу на екрану. Постоје три начина на које можемо да додамо CSS у HTML документ:

- **Инлајн (Inline):** Директно унутар HTML ознаке преко атрибута style. (Пример: <p style="color: red;">).
- **Унутрашњи (Internal):** Унутар самог HTML документа, у посебној ознаци <style> смештеној у делу <head>.
- **Спољашњи (External) - професионални начин:** Код се пише у посебном фајлу са наставком .css (на пример: style.css), који се затим повезује са HTML-ом преко ознаке <link>.

Да бисмо обојили или променили неки елемент, прво треба да кажемо компјутеру који елемент мењамо. То радимо помоћу **селектора**.

Основни селектори:

- **Селектор по ознаци (Element selector):** Бира све ознаке тог типа на страници.

```
p {
  color: blue; /* Све пасусе чини плавим */
}
```

- **Селектор по класи (Class selector):** Циља групу елемената који имају атрибут class. Увек почиње **тачком (.)**.

```
.crveni-tekst {
  color: red;
}
```

- **Селектор по идентификатору (ID selector):**Циља само један јединствени елемент са атрибутом id. Увек почиње **тарабом (#)**.

```
#glavni-naslov {
  font-size: 30px;
}
```

У CSS-у боја текста мења се наредбом color. Боје можемо дефинисати на три начина:

1. Са **називом боје**: red, blue, green, orange.
2. Са **HEX кодом** (са тарабом): #ff0000 (црвена), #ffffff (бела), #000000 (црна).
3. Са **RGB вредностима**: rgb(255, 0, 0).

```
h2 {
  color: #2c3e50; /* Тамноплава нијанса */
}
```

Уређивање текста је кључно за леп изглед веб-странице. Најважније наредбе су:

- **font-family:** Избор стила слова (нпр. Arial, Verdana, Times New Roman).Препоручује се да се увек наведе генеричка породица као sans-serif (без украса) или serif (са украсима) као замена ако компјутер нема тачан фонт.
- **font-size:** Величина слова, најчешће изражена у пикселима (px).
- **font-weight:** Подебљавање текста (bold за подебљан, normal за обичан).

```
body {
  font-family: Arial, sans-serif;
  font-size: 16px;
}
```

Позадина целе странице (ознака <body>) или неког одређеног елемента (као табела или пасус) уређује се помоћу:

- **background-color:** Мења боју позадине.
- **background-image:** Поставља слику као позадину преко линка (url('slika.jpg')).

```
body {
  background-color: #f4f4f4; /* Нежна светлосива позадина */
}
```

Сваки елемент у HTML-у (наслов, слика, пасус) компјутер види као једну **невидљиву правоугаону кутију**. Распоред и простор око тих кутија контролишемо помоћу три кључне наредбе:

- **padding (Унутрашња маргина):** Простор **унутар** кутије, између текста и њеног оквира.
- **border (Оквир):** Линија око елемента (може имати дебљину, стил и боју, нпр. 2px solid black).
- **margin (Спољашња маргина):**Простор **изван** кутије, који прави размак између овог елемента и других елемената око њега.

```
.kutija {
```

```
    border: 1px solid #ccc;
    padding: 20px; /* Текст унутра биће одвојен од оквира */
    margin: 15px;  /* Кутија ће се одмакнути од других елемената */
  }
```

У следећем примеру приказано је како се користе наредбе за промену боје слова и позадине.

```
<!DOCTYPE html>
<html lang="mk">
<head>
  <meta charset="UTF-8">
  <title>Едноставан CSS пример</title>
</head>
<body style="background-color: #f0f0f0; font-family: Arial;">

  <h1 style="color: blue;">Мој прв стилизован наслов</h1>

  <p style="background-color: red; color: white; padding: 10px;">
    Овој текст има црвена позадина и бела слова!
  </p>

</body>
</html>
```

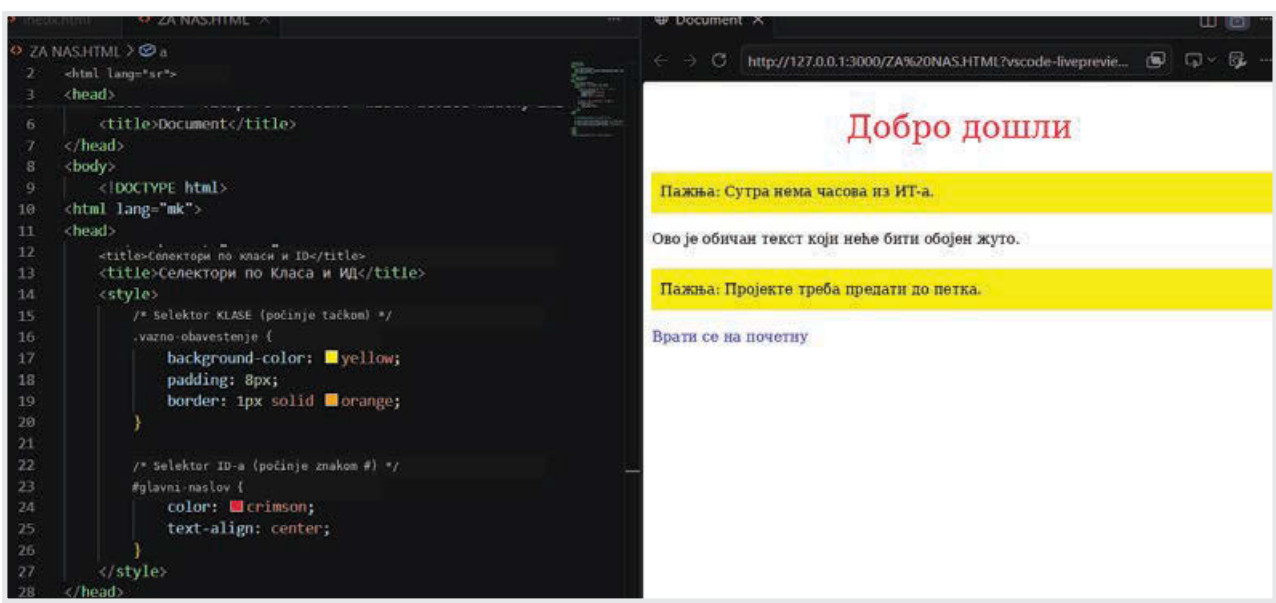
Која од наредби дефинише језик на ком ће текст бити приказан? На који начин је додат CSS?

Када желимо да стилизујемо конкретне (специфичне) елементе странице, боље је да их дефинишемо појединачно, уместо све одједном. За ту сврху користе се две врсте „ознака" (селектора):

Класа (Class) и **Идентификатор (ID)**.

```
<!DOCTYPE html>
<html lang="mk">
<head>
  <meta charset="UTF-8">
  <title>Селектори по класи и ID-у</title>
  <style>
    /* Селектор за КЛАСУ (почиње тачком) */
    .vazno-izvestuvanje {
      background-color: yellow;
      padding: 8px;
      border: 1px solid orange;
    }
      border: 1px solid orange;
    }
```

```
        /* Селектор за ID (почиње тарабом) */
        #glaven-naslov {
            color: crimson;
            text-align: center;
        }
    </style>
</head>
<body>

    <h1 id="glaven-naslov">Добро дошли</h1>

    <p class="vazno-izvestuvanje">Пажња: Сутра нема часова ИТ-а.</p>
    <p>Ово је обичан текст који неће бити обојен жутом бојом.</p>
    <p class="vazno-izvestuvanje">Пажња: Пројекте треба предати до петка.</p>

</body>
</html>
```



Слика 13: Стилизовање помоћу класе и идентификатора

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА:

1. Шта означава скраћеница CSS?
- А) Computer Style Sheets
- Б) Creative Style Systems
- В) Cascading Style Sheets
- Г) Colorful Style Software

2. Којим симболом почиње селектор за КЛАСУ (class) у CSS-у?
- А) Тарабом (#)
- Б) Тачком (.)
- В) Узвичником (!)
- Г) Упитником (?)

3. Објасни разлику у дејству између селектора p и селектора #glaven-paragraf. Који елемент на страници ће променити сваки од њих?

4. Ако неком елементу доделиш background-color: black; и намерно изоставиш наредбу color, зашто ће текст у том елементу постати невидљив кориснику?

5. Напиши тачан CSS код којим ће сви наслови типа <h2> на веб-страници добити зелену боју текста, величину 24 пиксела и фонт Arial.

6. Који од следећих CSS кодова правилно поставља црни оквир дебљине 2 пиксела око слике?
- А) img { margin: 2px black; }
- Б) img { border: 2px solid black; }
- В) img { padding: 2px solid; }
- Г) img { line: 2px black; }

7. Размотри следећу ситуацију: имаш две кутије једну испод друге. Желиш да направиш веће растојање (празан простор) између њих. Коју наредбу из „Box Model"-а треба да употребиш и зашто наредба padding неће решити овај проблем?

8. Твој друг из разреда написао је цео CSS код помоћу инлајн стилова (директно у свакој HTML ознаци са style="..."). Пројекат има 5 различитих веб-страница. Процени овај приступ: шта ће се догодити ако наставник сутра затражи да промени позадину на свих 5 страница из сиве у белу? Да ли би спољашњи CSS фајл био бољи избор?

9. Дизајнирај комплетан CSS стил за класу .karticka-proizvod. Кутија треба да има светлосиву позадину, унутрашњи размак (padding) од 15 пиксела, танак оквир и текст у фонту Verdana.

10. Напиши кратак HTML и CSS код у ком ћеш креирати један главни наслов са јединственим идентификатором id="naslov-akcija". У CSS делу подеси да овај наслов буде подебљан (bold) и да има жуту позадину (background-color).

5.2.7. РАСПОРЕД И АДАПТИВНИ (RESPONSIVE) ДИЗАЈН

Када креирамо веб-страницу, није довољно само обојити елементе. Важно је знати како да их распоредимо на екрану (лево, десно, једно поред другог) и како да се тај распоред аутоматски прилагоди када се страница гледа на компјутеру, таблету или мобилном телефону. То се постиже одговарајућим распоредом и адаптивним дизајном.

Распоред представља структуру или „скелет" веб-странице. Одређује где ће бити логотип, где мени, где главни текст, а где бочна трака (sidebar).

У CSS-у се елементи подразумевано ређају један испод другог (блок елементи као , <h1>, <p>). Да бисмо направили другачији распоред, користимо наредбу **display**. Најсавременији и најједноставнији начин за ређање елемената је **Flexbox** (Flexible Box). Када родитељској кутији доделимо наредбу display: flex;, сви њени унутрашњи елементи аутоматски се ређају **један поред другог (у колоне)**, уместо један испод другог.

```
.kontejner {
  display: flex;
  justify-content: space-between; /* Равномерно раздваја елементе лево и десно */
}
```

Респонзивни веб-дизајн (Responsive Web Design) је приступ креирању веб-страница који омогућава да се њихов изглед и садржај аутоматски прилагођавају величини екрана уређаја са ког се приступа – било да је то паметни телефон, таблет, лаптоп или велики десктоп монитор.

Уместо прављења посебних верзија странице за мобилне телефоне и компјутере, респонзивни дизајн користи исти код који се флексибилно прераспоређује и скупља или шири у зависности од расположивог простора.

Главни принципи респонзивног дизајна:

1. Флуидне мреже (Fluid Layouts): Уместо фиксних димензија у пикселима (нпр. width: 900px;), користимо проценте (width: 100%; или width: 50%;). Тако се елементи шире или сужавају у зависности од величине екрана.
2. Флуидне слике: Да слика никада не изађе ван екрана телефона, задајемо јој следеће златно правило у CSS-у:

```
img {
  max-width: 100%;
  height: auto;
}
```

Зашто је кључан за савремене веб-странице?

- Доминација мобилног саобраћаја: Више од половине глобалног интернет саобраћаја долази са мобилних уређаја. Ако ваша страница изгледа лоше или је тешка за навигацију на телефону, губите велики број потенцијалних посетилаца.
- Боље корисничко искуство (UX): Корисници не желе стално да зумирају, хоризонтално скролују или притискају ситна дугмад. Респонзивни дизајн нуди чист приказ, лако читљив текст и једноставну навигацију на сваком уређају.

- Значај за SEO (оптимизацију за претраживаче): Претраживачи као Google користе „mobile-first indexing“. То значи да Google првенствено оцењује мобилну верзију ваше странице када одлучује на ком месту ће је рангирати у резултатима претраге.
- Лакше одржавање и економичност: Уместо трошења времена и новца на развој и ажурирање две различите веб-странице (једне за десктоп, друге за мобилни), одржава се само један систем који ради свуда.

Данас респонзивни дизајн више није само „додатак“ или луксуз, већ основни стандард за сваку професионалну веб-страницу која треба да буде успешна и видљива на интернету.

Најважнија алатка у CSS-у за прилагођавање веб-странице назива се **Media Queries** (медијски упити). Они раде као филтери или услови: „Ако је екран мањи од 600 пиксела, примени ова нова CSS правила“. Замисли да на компјутеру имаш три кутије постављене једна поред друге (flex). На мобилном уређају екран је узак и кутије ће се „згурати“. Помоћу Media Query кажемо им да се поређају **једна испод друге** на мобилном уређају.

```
/* Стандардни CSS за компјутере (елементи су један поред другог) */
.glavno-meni {
  display: flex;
  flex-direction: row;
}

/* Посебно правило за мобилне телефоне (екрани ужи од 600px) */
@media (max-width: 600px) {
  .glavno-meni {
    flex-direction: column; /* Окрећемо елементе да се поређају један испод другог */
  }
}
```

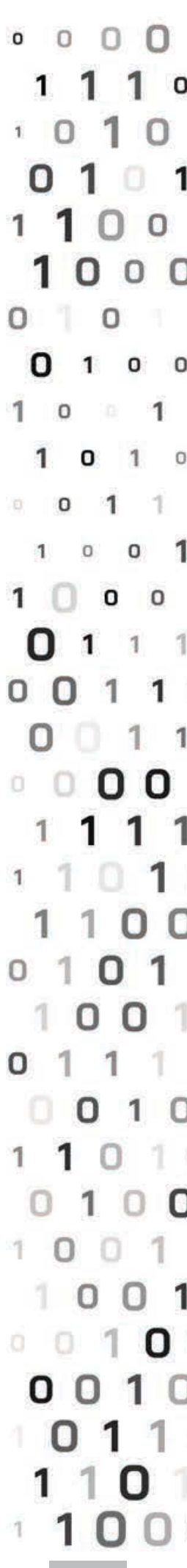
Сваки елемент/кутија у HTML-у има такозвани Box Model (модел кутије). Чак и ако не видиш оквир, компјутер за сваки елемент израчунава четири ствари:

- **Садржај (content):** Сам текст или слика.
- **Унутрашња маргина (padding):** Празан простор око текста како не би био приљубљен уз ивице.
- **Оквир (border):** Линија око елемента (може бити невидљива, а можемо је и обојити).
- **Спољашња маргина (margin):**Простор који одваја ту кутију од суседних кутија како се не би додиривале.

Када дизајнирамо веб-страницу, заправо се играмо овим кутијама – ширимо их, сужавамо, бојимо њихове позадине и одлучујемо да ли ће стајати у истом реду или једна испод друге!

У примеру испод приказан је адаптивни дизајн веб-странице. Искористи код у VSC едитору и тестирај величину прозора смањивањем и повећавањем да уочиш како функционишу display: flex; и @media.

```
<!DOCTYPE html>
<html lang="mk">
<head>
    <meta charset="UTF-8">
    <title>Кратак адаптивни дизајн</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            padding: 20px;
        }

        /* КОНТЕЈНЕР: На компјутеру ређа картице хоризонтално (у ред) */
        .grupacija-kutii {
            display: flex;
            flex-direction: row;
            gap: 15px; /* Размак између кутија */
        }

        /* КАРТИЦА: Стил за сваку појединачну кутију */
        .karticka {
            background-color: white;
            padding: 20px;
            border: 1px solid #ddd;
            flex: 1; /* Кутијама задаје да равномерно поделе ширину */
        }

        /* МЕДИЈСКИ УПИТ: За екране мање од 600 пиксела (мобилне) */
        @media (max-width: 600px) {
            .grupacija-kutii {
                flex-direction: column; /* РЕЂА КУТИЈЕ ЈЕДНУ ИСПОД ДРУГЕ */
            }
        }
    </style>
</head>
<body>

    <h2>Наш адаптивни распоред</h2>

    <div class="grupacija-kutii">
        <div class="karticka">
```



```
        <h3>Кутија 1</h3>
        <p>Ово је садржај прве кутије.</p>
    </div>
    <div class="karticka">
        <h3>Кутија 2</h3>
        <p>Ово је садржај друге кутије.</p>
    </div>
    <div class="karticka">
        <h3>Кутија 3</h3>
        <p>Ово је садржај треће кутије.</p>
    </div>
</div>
```

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА:

Која CSS наредба се користи за активирање флексибилног модела распореда (Flexbox)?

- А) display: block;
- Б) display: flex;
- В) position: static;
- Г) float: left;

2. Шта се постиже коришћењем CSS правила @media (Media Queries)?

- А) На страницу се додају видео и аудио датотеке.
- Б) Примењују се различити CSS стилови у зависности од величине екрана.
- В) Мења се језик веб-странице.
- Г) Страница се повезује са друштвеним мрежама.

3. Зашто је боље користити проценте (%) уместо фиксних пиксела (px) за ширину главних контејнера када дизајнирамо респонзивну веб-страницу?

4. Објасни разлику у понашању елемената када је flex-direction постављен на row у односу на column.

5. Напиши CSS правило које ће омогућити да све слике (<img>) на страници буду респонзивне (да не прелазе ширину свог родитељског елемента и да задрже однос димензија).

6. Који од следећих кодова правилно циља уређаје чија је максимална ширина екрана 768 пиксела (као таблети)?

- А) @media (width: 768px)
- Б) @media (max-width: 768px)
- Г) @media (min-width: 768px)

7. Ученик је направио распоред у ком навигациони мени и главни садржај стоје једно поред другог на компјутеру. Када отвори сајт на мобилном, садржај се „збије“ и текст не може да се прочита. Анализирај проблем и наведи који принцип респонзивног дизајна је прекршен и како треба променити распоред на мобилном.

8. Замисли да треба да изабереш стратегију за развој новог школског веб-сајта. Једна опција је да направиш две посебне верзије (једну потпуну за компјутер и посебну за мобилни са другим линком), а друга да направиш једну респонзивну страницу са Media Queries. Процени ефекат обе опције са аспекта одржавања и будућих измена информација.

9. Креирај једноставан CSS код за класу .kutija-blok која ће на компјутерима заузимати тачно 33.3% ширине екрана, али помоћу Media Query подеси да на мобилним уређајима (испод 600 пиксела) њена ширина аутоматски постане 100%.

10. Дизајнирај HTML структуру и CSS код за једноставно заглавље (Header) сајта које садржи Наслов (лево) и Линкове (десно). На компјутеру треба да стоје у истом реду (једно поред другог) помоћу Flexbox-а, а када се екран сузи испод 480 пиксела, наслов и линкови треба да се центрирају и поређају један испод другог.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Када правимо респонзивни дизајн, проценти (%) често нису довољни јер зависе од величине „родитељске“ кутије. Зато професионалци користе мерне јединице директно везане за величину целог екрана (Viewport):

vw (Viewport Width): 1vw је једнако 1% **ширине** екрана. Ако поставиш width: 50vw;, елемент ће увек заузимати тачно половину екрана, без обзира на уређај на ком се отвара.

vh (Viewport Height): 1vh је једнако 1% **висине** екрана. Ово је одличан начин за прављење почетних („Hero“) секција које треба да испуне цео екран по висини када се сајт учита:

```
.pocetna-sekcija {
    height: 100vh; /* Заузима целу висину екрана */
}
```

Један од највећих изазова у респонзивном дизајну је фонт. Ако је наслов превелик на компјутеру, на мобилном ће се „преломити“ у три реда. Уместо да пишеш десетине Media Queries само за фонтове, у модерном CSS-у постоји функција clamp().

Она прима три вредности: минималну, препоручену (флуидну) и максималну величину.

```
h1 {
    font-size: clamp(20px, 5vw, 45px);
}
```

Док је Flexbox одличан за ређање елемената у једном смеру (само у реду или само у колони), CSS Grid је намењен комплетном дводимензионалном распореду (истовремено редови и колоне). Помоћу Grid-а можеш да направиш распоред као у правим новинама или магазину.

```
.magazin-raspored {
    display: grid;
    grid-template-columns: repeat(3, 1fr); /* Прави тачно 3 једнаке колоне */
    grid-gap: 20px;
}
```

А на мобилном, само једним медијским упитом једноставно мењаш број колона:

```
@media (max-width: 600px) {
    .magazin-raspored {
        grid-template-columns: 1fr; /* На мобилном се све претвара у 1 колону */
    }
}
```

Понекад велику табелу или сложену слику нема смисла приказивати на мобилном телефону, јер ће само успорити интернет корисника и „згурати" простор. Помоћу Media Queries можеш потпуно да сакријеш одређени елемент на мобилном, а да га оставиш видљивим на компјутеру:

```
/* На мобилном сакриј бочну траку са огласима */
@media (max-width: 600px) {
    .stranicna-reklama {
        display: none;
    }
}
```

5.2.8. ГЕНЕРАТИВНА ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА ЗА ВЕБ-РАЗВОЈ

Технологија генеративне вештачке интелигенције (Generative AI) направила је револуцију у начину на који се креирају веб-странице. Алатке као што су ChatGPT, Claude, Microsoft Copilot и специјализовани додаци у VS Code-у (као GitHub Copilot) сада могу да пишу сложен HTML и CSS код за неколико секунди.

Овај садржај ће ти помоћи да разумеш како процес функционише, како правилно да задајеш упутства компјутеру и како критички да анализираш код који ће вештачка интелигенција направити за тебе.

1. Појам инструкције (prompt) и формулисање јасних инструкција

Један од најважнијих концепата у раду са вештачком интелигенцијом је **prompt (промпт)**. Промпт је текстуална инструкција или наредба коју корисник уноси у AI алатку да би добио одређени резултат (у нашем случају веб-код).

Ако AI-ју кажеш:„Направи ми веб-страницу за школу“, резултат ће бити превише општи и једноставан. Да би добио професионалан код, треба да користиш структуриран и детаљан промпт.

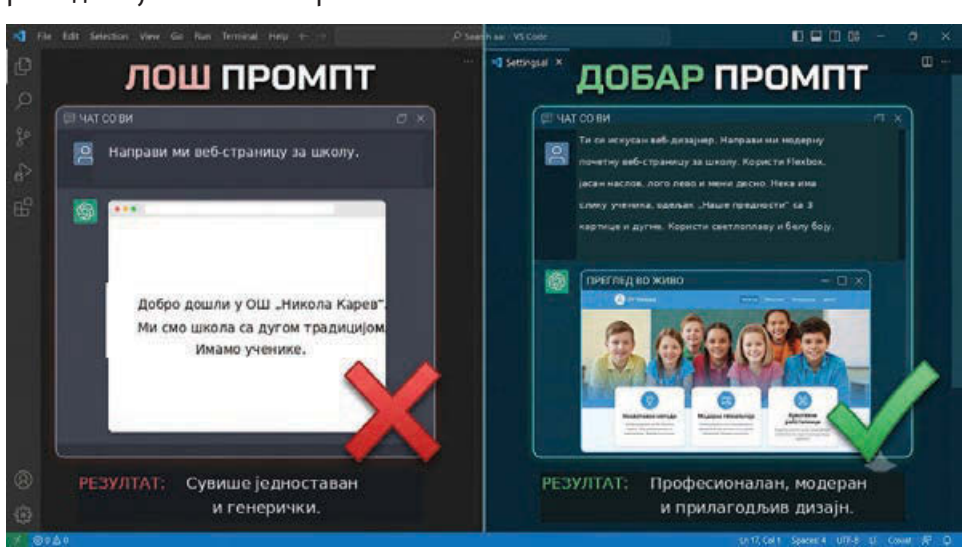
Како формулисати јасан и прецизан промпт за веб-развој?

Квалитетан промпт треба да садржи следеће елементе:

- **Улога (Role):** Дефиниши улогу AI-ја (нпр. „Ти си искусан веб-дизајнер“).
- **Контекст/Циљ:** Шта је тачно замисао странице.
- **Технички захтеви:** Које језике треба користити (HTML, CSS са унутрашњим/спољашњим стилом, Flexbox распоред).
- **Специфични детаљи:** Боје, број секција, респонзивност за мобилне уређаје.

Пример лошег промпта: „Напиши ми HTML за контакт веб-страницу.“

Пример одличног промпта: „Ти си професионални веб-програмер. Напиши ми комплетан HTML и унутрашњи CSS код за адаптивну веб-страницу ‹Контакт›. Страница треба да садржи форму са пољима за Име, Е-пошту и Поруку, као и дугме за слање. Користи модеран дизајн са тамноплавом позадином (#2c3e50), белим текстом и користи Flexbox да центрираш форму на средини екрана. На мобилним уређајима (испод 600px) форма треба да заузима 95% ширине.“



Слика 14: Пример општог и специфичног промпта

2. Аутоматско креирање веб-страница и анализа резултата

Поступак рада са асистентом вештачке интелигенције (ВИ) при креирању веб-страница одвија се у неколико кључних корака. Да би добио квалитетан код, процес није само „копирај-налепи", већ обухвата планирање, инструкције и проверу.

Ево целог процеса, корак по корак:

- Планирање структуре (пре него што почнеш да пишеш код)

Пре него што уопште отвориш ВИ алатку, треба да имаш јасну представу о томе шта желиш да направиш. Скицирај распоред (layout), где желиш да буде главни мени, колико кутија са садржајем ће бити и које боје ћеш користити.

- Формулисање јасног промпта (инструкције)

Ово је најважнији корак. Уместо кратких и нејасних реченица, ВИ асистенту треба формулисати детаљан опис.

- **Постави улогу:** „Ти си искусан веб-програмер."
- **Дефиниши циљ:** „Направи ми веб-страницу за..."
- **Наведи техничка правила:**„Користи HTML5 и унутрашњи CSS у ознаци <style>. Распоред направи помоћу flexbox-а. Код треба да буде адаптиван (responsive) за мобилне уређаје помоћу медијских упита."

- Генерисање и преузимање кода

ВИ асистент ће написати код за неколико секунди. Твој задатак је:

1. Да прегледаш код још у самом прозору за разговор.
2. Да копираш одговарајући блок кода.
3. Да га пребациш у своју нову датотеку (на пример index.html) у VS Code-у и сачуваш.

- Анализа и тестирање у прегледачу (провера резултата)

Када сачуваш фајл, отвори га у прегледачу (Chrome, Edge). Сада критички анализираш урађено:

- **Изглед:** Да ли су боје и фонтови онакви какве си тражио?
- **Структура:** Да ли су HTML ознаке правилно и логички постављене (наслови, пасуси, кутије)?
- **Адаптивност (Responsive):** Смањи прозор да видиш да ли се елементи на мањем екрану правилно ређају један испод другог.

- Итерација и ручна дорада (исправљање грешака)

Готово никада први одговор ВИ није 100% савршен. Овде наступаш ти као програмер:

- **Ако постоји мала грешка:** Исправи је сам ручно у VS Code-у (на пример, промени боју из црвене у плаву у CSS делу). То је најбољи начин за учење.
- **Ако постоји већа грешка или недостаје цела секција:** Врати се ВИ асистенту и дај му додатни промпт, на пример: „Код је добар, али додај још једно зелено дугме испод текста."
-

Овим кружним процесом вештачка интелигенција ти помаже да брзо завршиш тежак и механички посао, а ти задржаваш потпуну контролу над логиком и коначним изгледом веб-странице.

Да би постао добар корисник ВИ, треба да познајеш предности и недостатке оба приступа. Ево директног поређења:

Карактеристика	Рачно напишан код (Hand-coded)	ВИ-генериран код (AI-generated)
Брзина израде	Споро (захтева ручно писање сваке линије и загради).	Изузетно брзо (целе странице су готове за неколико секунди).
Контрола и разумевање	**Максимална**. Тачно знаш зашто је свака линија кода ту и лако је исправљаш ако постоји грешка.	**Ограничена**. Ако је код предугачак, тешко је разумети где је настао проблем ако се поквари нешто.
Оптимизација и логика	Код је чист, написан тачно за твоје потребе, без непотребних понављања.	Може да садржи „сувишан код" (непотребне наредбе које AI су додате без стварне потребе).
Учење градива	Најбољи начин за учење. Напредак долази кроз прављење и исправљање сопствених грешака.	Не помаже у учењу ако се само слепо копира и лепи без читања.

Вештачка интелигенција је **изванредан асистент, али лош господар**. Најбоље је да искористиш њену брзину за прављење брзог скелета (шаблона) странице, а затим да ти ручно прегледаш, очистиш и прилагодиш код својим прецизним жељама и правилима!

- Анализа постојећег HTML/CSS кода према структури и изгледу

Када вештачка интелигенција генерише код на основу твог промпта, посао није завршен. Као будући корисник ВИ, треба да анализираш резултат. Вештачка интелигенција није непогрешива – често генерише код који изгледа лепо, али у позадини може да садржи озбиљне пропусте.

На шта треба обратити пажњу при анализи ВИ кода?

- **Функционалност:**Да ли раде сви линкови? Да ли форма дозвољава унос текста?
- **Прилагођеност (адаптивност):** Мењај ширину прозора прегледача. Да ли је ВИ додао одговарајуће @media упите за уређаје или се код изобличава на мобилном?
- **Чистоћа кода:** Да ли су класе јасно именоване (нпр. .karticka-proizvod) или су збуњујуће?

Да би знао да ли је ВИ добро урадио посао, треба да „прочиташ" код и анализираш његову **структуру (HTML)**и **изглед (CSS)**.

Замислимо да је ВИ генерисао следећи део кода:

HTML

```
<div class="karticka">
   <h2 class="naslov">Добро дошли</h2>
   <p>Прочитајте најновије вести овде.</p>
</div>
</div>
```

CSS

```
.karticka {
   background: #fff;
```

```
    padding: 20px;
    border-radius: 5px;
}
.naslov {
    color: red;
}
```

Анализа структуре (HTML): Код је логичан. Имамо контејнер-правоугаоник (div) који садржи један наслов и један пасус. То омогућава да се сви елементи померају и уређују заједно као једна целина.

Анализа изгледа (CSS): Кутија има белу позадину, заобљене углове и унутрашњи размак (padding) како текст не би био прилепљен уз ивице. Наслов има специфичну црвену боју. Структура и изглед су лепо раздвојени.

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА:

1. Који појам се користи за текстуалну инструкцију или наредбу коју корисник уноси у ВИ алатку да би добио код?
   - А) Скрипта (Script)
   - Б) Промпт (Prompt)
   - В) Таг (Tag)
   - Г) Коментар (Comment)

2. Која је главна улога медијских упита (@media) које ВИ асистент генерише у CSS коду?
   - А) Да омогуће страници да се прилагођава различитим величинама екрана.
   - В) Да убрзају учитавање слика на сајту.
   - Б) Да повежу веб-сајт са друштвеним мрежама.
   - Г) Да пронађу логичке грешке у HTML структури.

3. Зашто се промпт „Направи ми веб-страницу за патике“ сматра непотпуним и лошим промптом када радимо са ВИ асистентом?

4. Објасни шта се догађа са изгледом веб-странице на мобилном телефону ако је ВИ асистент генерисао код са фиксним ширинама у пикселима (px) уместо процената (%) или флуидног распореда.

5. Напиши детаљан и структуриран промпт којим ћеш од ВИ асистента затражити да креира навигациони мени (Header) са три линка, тамнозеленом позадином и који користи Flexbox за распоређивање.

6. Ако у коду који је генерисао ВИ асистент уочиш следећу линију: <meta name="viewport" content="width=device-width, initial-scale=1.0">, за који уређај је ВИ асистент обезбедио да се страница правилно приказује?
   - А) Само за десктоп компјутере са великим мониторима.

- Б) Само за штампаче када се страница штампа на папиру.
- В) За све уређаје, укључујући и мобилне телефоне.
- Г) Само за претраживаче који не подржавају CSS.

7. Ученик користи ВИ асистента да генерише образац за регистрацију. Након што копира код у VS Code, примећује да дугме „Пошаљи" стоји на врху странице уместо на дну, испод поља. Анализирај у којем делу кода (HTML или CSS) је направљена логичка грешка у структури и како би је исправио.

8. Размотри следећи CSS код који је генерисала ВИ: .tekst { color: blue; font-size: 16px; font-style: italic; }. Ученик је тражио да код нема искошена слова. Опиши коју наредбу из овог кода треба избрисати или променити да би се испунио захтев.

9. Процени предности и недостатке коришћења кода генерисаног помоћу ВИ у поређењу са ручно написаним кодом, са становишта учења и разумевања програмирања код почетника.

10. Замисли да ти је ВИ асистент генерисао одличан распоред са две картице постављене једна поред друге на компјутеру, али када тестираш код у претраживачу, схватиш да је заборавио да дода респонзивност за мобилне уређаје. Креирај (напиши) CSS медијски упит који недостаје како би картице са класом .karticka спустио једну испод друге на екранима ужим од 600 пиксела.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Вештачка интелигенција не види само промпт који јој тренутно пишеш већ памти и претходни разговор. Тај простор за памћење назива се контекстни прозор. Ако желиш да знаш више, најбољи начин да то искористиш јесте да ВИ даш унапред дефинисана правила пре него што уопште затражиш код.

На пример, први промпт у дану може да буде формулисан овако:

„Током данашње сесије тражићу од тебе HTML и CSS код. Твој задатак је да никада не користиш застареле CSS наредбе (као што је float), да увек користиш Flexbox и да коментари у коду увек буду на српском језику."

Овим обучаваш асистента да ради по твојим стандардима још пре него што почне са креирањем.

5.2.9. УРЕЂИВАЊЕ И ПРИЛАГОЂАВАЊЕ ВИ КОДА

РАЗМОТРИМО ЈЕДАН ПРИМЕР КРЕИРАЊА ВЕБ-СТРАНИЦЕ ПОМОЋУ ВИ АЛАТКЕ. ВЕЋ СМО ПОМЕНУЛИ ДА VS CODE ИМА ВИ АСИСТЕНТА. ОТВОРИ VS CODE И НАПИШИ СЛЕДЕЋУ ИНСТРУКЦИЈУ:

Ти си професионални веб-дизајнер и кодер. Напиши ми комплетан, чист и лако разумљив HTML и унутрашњи CSS код за веб-страницу школског клуба под називом „ИТ и Гејминг Клуб Код-Елит".

Страница треба да испуни следеће техничке и дизајнерске стандарде:

1. Структура (HTML):

- Да има заглавље (header) са главним насловом и поднасловом који описује клуб.

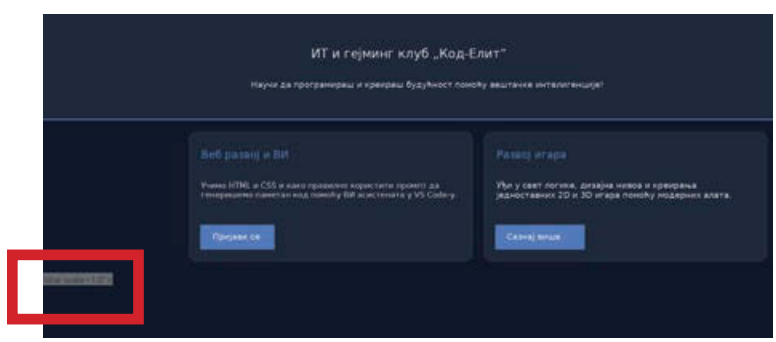
- Да има главни контејнер са две картице (кутије) за садржај: једну за „Веб-развој и ВИ“ и једну за „Развој игара“.
- Свака картица треба да има наслов, кратак текст и једно дугме (линк) за акцију.

2. Изглед и распоред (CSS):

- Користи модерну тамну тему (тамносива или тамноплава позадина) са веома светлим словима ради доброг контраста и лаког читања.
- Главни распоред картица направи помоћу Flexbox-а (display: flex), тако да на компјутерском екрану кутије стоје хоризонтално, једна поред друге, једнаке ширине и са лепим размаком између њих.
- Картице треба да имају заобљене углове (border-radius) и дискретне оквире.

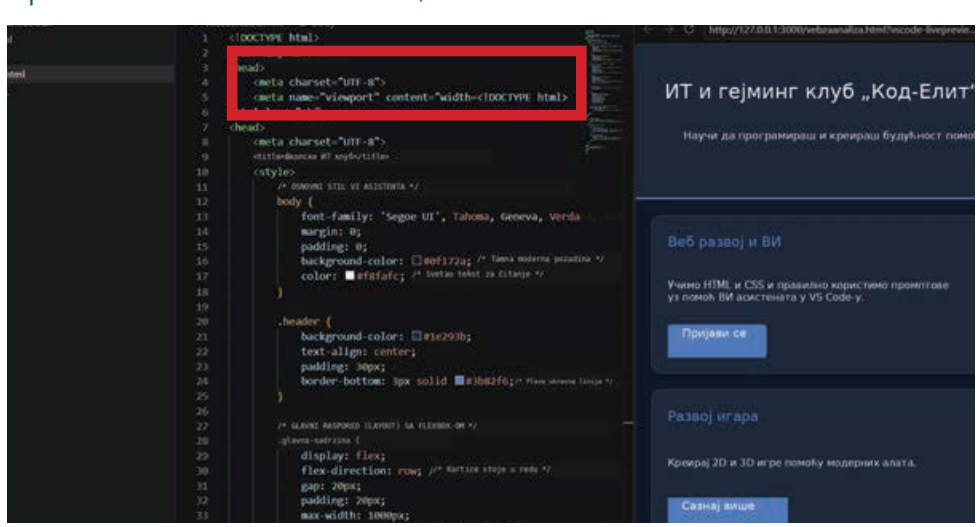
3. Адаптивност (Responsive):

- Укључи медијски упит (@media) за мобилне уређаје са максималном ширином од 768 пиксела.
- Када се страница отвори на мобилном уређају, Flexbox распоред треба да се промени у колону (column), тако да се картице аутоматски спусте једна испод друге и буду прегледне на уском екрану.
- Генерисани код сачувај са наставком index.html у VS Code-у и отвори га у претраживачу да видиш резултат. Примећујеш да се једна линија кода појављује на веб-страници, initial-scale=1.0”>.

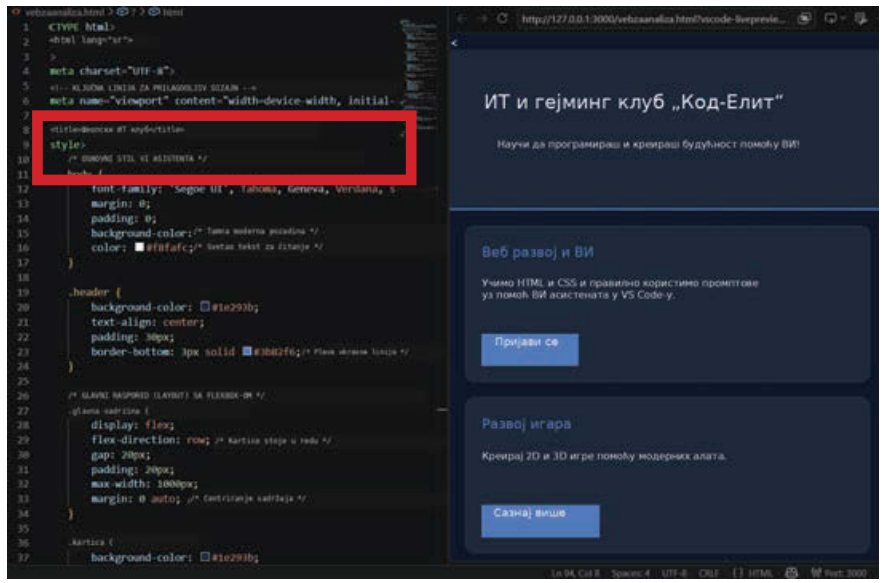


То значи да је ВИ асистент негде погрешио. Анализирај чему служи и где треба да се напише ова наредба. Затражи од ВИ асистента да исправи код. Ако не успе, исправи га ручно испод <head> линије наредбом

<meta name=”viewport” content=”width=device-width, initial-scale=1.0”>



Сачувај промене, кликни F5 и веб-страница ће изгледати исправно.



Затражи нове измене веб-странице, на пример: „У овом коду желим да променим главну тему. Преправи CSS тако да позадина буде тамнозелена, а главна плава дугмад и линије постану неонски жути.“ или: „Анализирај структуру кутија у овом коду и додај још једну, трећу кутију са насловом 'Сајбер-безбедност' која ће имати исти стил и уклопити се у Flexbox распоред.“ Може се затражити и провера адаптивности веб-странице: „Погледај медијски упит на дну. Како могу да променим код тако да на мобилним уређајима текст у заглављу (header) постане мањи и центриран?“

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА:

1. Ученик је затражио од ВИ асистента да направи респонзивну страницу. У коду који је добио линија за viewport написана је са грешком: <meta name="viewport" content="width=device-width", initial-scale=1.0">"> Шта ће се догодити када се ова страница отвори у претраживачу?

- А) Страница се уопште неће учитати и приказаће празан екран.
- Б) На дну странице појавиће се вишак текста као initial-scale=1.0">.
- В) Цео CSS стил ће се избрисати и текст ће постати црвен.
- Г) Контакт форма ће се аутоматски сакрити.

2. Размотри овај CSS код који треба да распореди картице једну поред друге:

```
.grupacija-kutii {
    display: block;
    flex-direction: row;
    gap: 15px;
}
```

3. Која наредба је погрешна и спречава да Flexbox распоред правилно функционише?

- А) gap: 15px;
- Б) flex-direction: row;
- В) display: block;
- Г) Име класе .grupacija-kutii има словну грешку.

4. Погледај следећи HTML код који је генерисао ВИ асистент:

```
<div class="karticka">
    <h3>Веб Развој</h3>
    <p>Учимо да кодирамо уз помоћ ВИ.
</div>
```

Објасни која структурна грешка постоји у овом делу кода и како она може да утиче на изглед осталих елемената испод њега.

5. У делу за медијске упите ВИ асистент је направио следећу грешку у синтакси:

```
@media max-width: 768px {
    .glavna-sodrzina {
        flex-direction: column;
    }
}
```

Зашто овај медијски упит не ради у претраживачу и који знаци недостају у првој линији да би код био исправан?

6. Ученик жели да прилагоди ВИ код како би променио боју позадине картице. ВИ асистент је написао следећи код:

```
.karticka {
    background-color: #0f172a
    padding: 20px;
    border-radius: 12px;
}
```

Који синтаксички знак недостаје у овом коду и које CSS правило се уопште неће применити (претраживач ће га игнорисати)?

7. Коју наредбу треба да избришеш или да промениш како би испунио захтев да код нема искошена слова (италик)?

```
.kopce {
    display: inline-block;
    background-color: #3b82f6;
    color: white;
    font-style: italic;
}
```

8. Размотри овај покушај ВИ асистента да направи респонзиван распоред за мобилне телефоне:

```
.karticka {
    width: 400px;
}
```

```
@media (max-width: 600px) {
    .karticka {
        width: 400px;
    }
}
```

Анализирај код и објасни зашто овај дизајн неће бити адаптиван на екранима ширине 360 пиксела (стандардни мобилни) и како треба уредити вредност width у медијском упиту.

9. Ученик користи следећи код да креира наслов странице:

```
<h1 id="glaven-naslov" class="glaven-naslov">ИТ Клуб Код-Елит</h1>
```

Ако у CSS делу постоје два правила: једно циља са .glaven-naslov { color: red; }, а друго са #glaven-naslov { color: blue; }, којом бојом ће се приказати наслов и зашто? Објасни принцип специфичности селектора.

10. ВИ асистент је генерисао код у којем две картице треба да буду једна поред друге на компјутеру, а једна испод друге на мобилном. Међутим, код изгледа овако:

```
.glavna-sodrzina {
    display: flex;
    flex-direction: column;
}

@media (max-width: 768px) {
    .glavna-sodrzina {
        flex-direction: column;
    }
}
```

Процени функционалност овог кода за оба уређаја (компјутер и мобилни) и предложи тачне вредности за flex-direction како би исправио распоред.

11. Ово је HTML код за две картице у којем је ВИ асистент направио логичку грешку приликом копирања класа:

```
<div class="glavna-sodrzina">
    <div class="karticka">
        <h3>Веб развој</h3>
        <p>Учимо да кодирамо уз помоћ ВИ.</p>
    </div>
    <div class="glavna-sodrzina">
        <h3>Развој игара</h3>
        <p>Креирамо 2D и 3D игре.</p>
    </div>
</div>
```

Креирај тачан HTML код тако што ћеш пронаћи и исправити погрешну класу у другој кутији како би она добила исти стил као прва картица.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Приступ „Прво мобилни“ (Mobile-First) уз помоћ ВИ

Већина почетника тражи од ВИ да направи страницу за компјутер, а затим је прилагођава за мобилни. Међутим, у модерној ИТ индустрији користи се приступ Mobile-First. То значи да се страница прво дизајнира за најмањи екран, а затим се проширује за велике мониторе.

Када тражиш код од ВИ овим приступом, инструкције мењаш овако:

Тражиш да основни CSS буде написан у једној колони (за мобилни телефон).

Тражиш да медијски упити користе min-width (минималну ширину) уместо max-width.

На тај начин, са @media (min-width: 1024px), вештачка интелигенција ће додати правила која ће се активирати тек када екран постане велики као на компјутеру, што доводи до много бржег учитавања странице на телефонима.

Аутоматско чишћење „AI кода-смећа“ (Refactoring)

Вештачка интелигенција има тенденцију да пише превише кода. Често ће за једну исту ствар измислити три различите класе или додати CSS наредбе које претраживач ионако подразумева. Процес чишћења и оптимизације тог кода назива се рефакторисање (Refactoring).

Након што ти ВИ генерише код, напредна техника је да му вратиш тај исти код са промптом за оптимизацију:

„Ово је код који си генерисао. Сада га рефакториши: споји дуплиране CSS класе, уклони непотребне маргине које се понављају и учини HTML структуру краћом и једноставнијом за читање.“

Интеграција у VS Code са паметним додацима

Напредни програмери не копирају код из претраживача (као што је ChatGPT), већ користе додатке директно у самом уређивачу VS Code (као што су GitHub Copilot или Codeium).

Ове алатке имају приступ свим датотекама у твојој фасцикли одједном. То значи да оне не само што генеришу нови код већ могу да анализирају твој стил писања и аутоматски ти предложе следећу линију кода још пре него што почнеш да је пишеш, притиском на само једно дугме (Tab).

5.2.10 КРЕИРАЊЕ ВЕБ-ПРОЈЕКТ

Пошто си научио како вештачка интелигенција ради и како се праве прилагођавања у коду, време је да погледамо цео процес креирања једне веб-странице, од прве идеје на папиру па све до коначног изгледа.

Професионално креирање веб-странице увек се одвија кроз 4 кључна корака који морају да се прате прецизним редоследом.

1. Планирање (Шта ћемо радити и за кога?)

Планирање је темељ веб-странице. Пре него што се напише иједна линија кода, мора се одговорити на следећа питања:

- **Који је циљ странице?** (Да ли је то портфолио, школски сајт или онлајн продавница?)
- **Која је циљна публика?** (Да ли је правимо за тинејџере, наставнике или гејмере?)
- **Шта ће све садржати?** Прави се списак потребних делова: заглавље, картице са производима, галерија слика, контакт форма.

2. Структура (Како ће код бити организован?)

- Структура се односи на скелет странице, односно на логичко распоређивање HTML ознака. Професионални дизајнери у овој фази цртају **жичани оквир (Wireframe)** – једноставну скицу са правоугаоницима која показује где ће стајати сваки елемент.

У коду ово претварамо у чист, семантички HTML5:

- Користимо <header> за мени на врху.
- Користимо контејнере (или <section>) са јасним класама да групишемо елементе.
- Водимо рачуна о хијерархији наслова (<h1> за главни наслов, <h2> и <h3> за поднаслове картица).

3. Дизајн (Како ће страница изгледати?)

Када имамо чврсту структуру, прелазимо на CSS стилизовање. Дизајн одређује визуелни идентитет и корисничко искуство.

Овде примењујемо сва изучена CSS правила:

- **Распоред (Layout):** Користимо Flexbox како би се елементи ширили и паметно распоређивали.
- **Типографија и боје:** Бирамо фонтове који се лако читају и палету од 2 до 3 усклађене боје (на пример, тамну модерну тему са контрастним плавим детаљима).
- **Простор:** Користимо маргине (margin) и унутрашње размаке (padding) како елементи не би били збијени и „залепљени“ једни уз друге.
- **Адаптивност (Responsive):** Помоћу медијских упита (@media) обезбеђујемо да се дизајн прилагоди са три колоне на компјутеру на једну колону на мобилном уређају.

4. Интеграција садржаја (Попуњавање странице стварним подацима)

Интеграција је завршни корак у којем „лажни“ или генерички текст који нам је дао ВИ асистент замењујемо стварним, реалним информацијама:

- Пишемо коначан текст у пасусима (<p>).
- Привремене линкове замењујемо тачним путањама до стварних слика и спољних страница.
- Повезујемо дугмад са одговарајућим радњама.

Тек када сва 4 корака буду завршена овим редоследом, можемо рећи да имамо комплетну, функционалну и професионално израђену веб-страницу.

Да би боље усвојио фазе припреме, најбоље је да израдиш своју личну веб-страницу (портфолио) на којој ћеш приказати све вежбе, игре или веб-странице које си направио или планираш да правиш у будућности. То је нешто што професионални програмери увек имају.

Архитектура и елементи које веб-страница треба да садржи:

1. Структура и семантички HTML тагови

Страница треба да буде подељена на јасне логичке целине:

- <header>: На врху, са његовим именом и кратким поднасловом (нпр. „Будући веб-програмер“).
- <main>: Главни део у којем ће бити садржај.
- <section> са класом .proekti: Овде ће стајати картице за његове пројекте.
- <footer>: На дну, са текстом о ауторским правима.

2. Распоред (Layout) помоћу Flexbox-a

У секцији за пројекте треба да постоје **3 картице** (на пример: „Школски сајт“, „Моја прва игра“, „ВИ вежба“).

- Ове 3 картице морају бити смештене у један родитељски контејнер који ће имати display: flex; и flex-direction: row; како би на компјутеру биле лепо распоређене у једном реду.

3. Адаптивни дизајн (@media упити)

Страница мора да изгледа савршено и на мобилном телефону.

- Треба додати @media (max-width: 768px) где ће се Flexbox распоред променити у flex-direction: column; како би се картице спустиле једна испод друге.
- У <head> делу обавезно треба поставити исправну viewport линију како се на екрану не би појављивао вишак текста.

Како искористити ВИ асистента за овај пројекат? (Процес рада)

Ученик треба да води пројекат кроз следећа 3 корака са ВИ у VS Code:

- **Први промпт (Генерисање скелета):** Ученик пише ВИ: „Ти си мој ментор. Напиши ми HTML5 структуру и почетни CSS за мој лични ИТ портфолио. Желим тамносиву позадину, са заглављем, главним делом са 3 картице за пројекте постављене помоћу Flexbox-a и подножјем. Укључи и респонзивност за мобилне уређаје испод 768px.“
- **Тестирање и проналажење грешака (Анализа):** Ученик покреће код у претраживачу. Проверава да ли је ВИ заборавио неку тачку и запету (;), да ли се текст изобличава или се случајно појавио неки искошени текст (италик) који не желимо.

- **Други промпт (Прилагођавање и дорада):** Након што анализира код, ученик тражи измену: „Код је одличан, али сада у другој картици промени наслов у 'Мој пројекат са ВИ асистентом', а боју дугмади на свим картицама учини светлоплавом са заобљеним угловима."
- **Личан је:** Ученик представља себе и осећа се као прави програмер.
- **Садржи све стандарде:** Захтева размишљање о структури (HTML), естетици и респонзивности (CSS), као и правилну комуникацију са технологијом (промпт инжењеринг).
- **Лако се проверава:**Као наставник/оцењивач, одмах можете видети да ли Flexbox ради када смањите прозор претраживача.

ПИТАЊА ЗА ПРОВЕРУ ЗНАЊА:

1. Ученик жели да направи веб-страницу за ИТ клуб, али пре него што почне са кодирањем, истражује које су информације најпотребније његовим школским друговима и прави списак секција. У којој фази израде пројекта се ученик налази у том тренутку?

- А) Дизајн
- Б) Структура
- В) Планирање
- Г) Интеграција садржаја

2. Када кажемо да правимо „жичани оквир" (Wireframe) или једноставну скицу са правоугаоницима како бисмо одредили где ће стајати наслови, слике и картице, ми заправо дефинишемо:

- А) Боју позадине сајта
- Б) HTML структуру странице
- В) Брзину учитавања сервера
- Г) Везу са спољним PDF документима

3. Размотри следећу ситуацију: Ученик је завршио HTML код и сада жели да распореди картице са пројектима помоћу Flexbox-а, да заобли углове кутија и дода медијске упите за мобилни телефон. Објасни коју фазу израде пројекта ученик обавља у овом кораку и који језик користи.

4. Шта представља фаза „Интеграција садржаја" и шта тачно ученик треба да уради са линковима које је ВИ асистент генерисао као href="#" да би његов портфолио постао функционалан?

5. Ученик је у свом коду направио следеће повезивање да би приказао контакт форму: Отвори. Анализирај зашто је ова путања погрешно постављена као апсолутна уместо локална и како ће то утицати ако неко други покуша да отвори страницу на свом компјутеру.

ЗА ОНЕ КОЈИ ЖЕЛЕ ДА ЗНАЈУ ВИШЕ

Планирање у веб-развоју укључује и креирање детаљне корисничке мапе (User Flow) која тачно предвиђа свако кретање и клик посетиоца кроз страницу. У овој напредној фази професионалци користе специјализоване дигиталне алатке као што су Figma или Adobe XD да би направили интерактивни прототип који изгледа и реагује као прави сајт још пре него што се напише прва линија HTML кода.

Додатно се планира и такозвана SEO стратегија (оптимизација за претраживаче), којом се унапред одређују кључне речи у насловима и мета-ознакама како би страница касније била лако пронађена помоћу претраживача.