Марина Лакчевић
Јасмина Алексић

ИНФОРМАТИКА И РАЧУНАРСТВО 6

уџбеник за шести разред основне школе

АрхиКњига - Завод за уџбенике

Рецензенти
др Драгана Сајферт, наставник Основне школе „Соња Маринковић”, Земун
Александра Иванов, професор Архитектонско-техничке школе у Београду
Драгана Станисављевић, наставник Основне школе „Радоје Домановић” у Београду

Уредник
др Жељко Станковић

Одговорни уредник
Слободанка Ружичић

Коректура
Завод за уџбенике
Обим: 7 штампарских табака
Формат: 20,5 × 26,5 cm

Главни уредник
Драгољуб Којчић

За издавача
Драгољуб Којчић, директор

Министар просвете, науке и технолошког развоја Републике Србије решењем број 650-02-00350/2018-07 од 21. 1. 2019. године одобрио је овај уџбеник за издавање и употребу у шестом разреду основне школе.

Марина Лакчевић
Јасмина Алексић

ИНФОРМАТИКА И РАЧУНАРСТВО 6

уџбеник за шести разред основне школе

АрхиКњига - Завод за уџбенике

Рецензенти
др Драгана Сајферт, наставник Основне школе „Соња Маринковић”, Земун
Александра Иванов, професор Архитектонско-техничке школе у Београду
Драгана Станисављевић, наставник Основне школе „Радоје Домановић” у Београду

Уредник
др Жељко Станковић

Одговорни уредник
Слободанка Ружичић

Главни уредник
Драгољуб Којчић

За издавача
Драгољуб Којчић, директор

Министар просвете, науке и технолошког развоја Републике Србије решењем број 650-02-00350/2018-07 од 21. 1. 2019. године одобрио је овај уџбеник за издавање и употребу у шестом разреду основне школе.

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд

37.016:004(075.2)

ЛАКЧЕВИЋ, Марина, 1979-
Информатика и рачунарство 6 : уџбеник за шести разред основне школе / Марина Лакчевић, Јасмина Алексић. - 1. изд. - Београд : Завод за уџбенике, 2019 (Београд : АМД Систем). - 125 стр. : илустр. ; 27 cm

Тираж 3.000. - Речник: стр. 123-124. - Буиблиографија: стр. 125.

ISBN 978-86-17-20154-6

1. Алексић, Јасмина, 1976- [аутор]

COBISS.SR-ID 277931020

ISBN 978-86-17-20154-6

Садржај

Кратак водич кроз уџбеник

Кључне речи:

Најважнији појмови издвојени на почетку лекције

Непознате речи:

Непозната реч
Објашњење непознате речи

Запамти

Издвојени појмови које би требало да запамтиш, научиш и разумеш.

Вежбе и питања за проверу стеченог знања

✓ Вежбе и питања којима ћеш проверити своје знање.

Користан линк

https://www. ...

Желим да знам више.

1

ИНФОРМАЦИОНО--КОМУНИКАЦИОНЕ ТЕХНОЛОГИЈЕ

1.1. Појам, значај и улога ИКТ-а у савременом животу

Кључне речи:

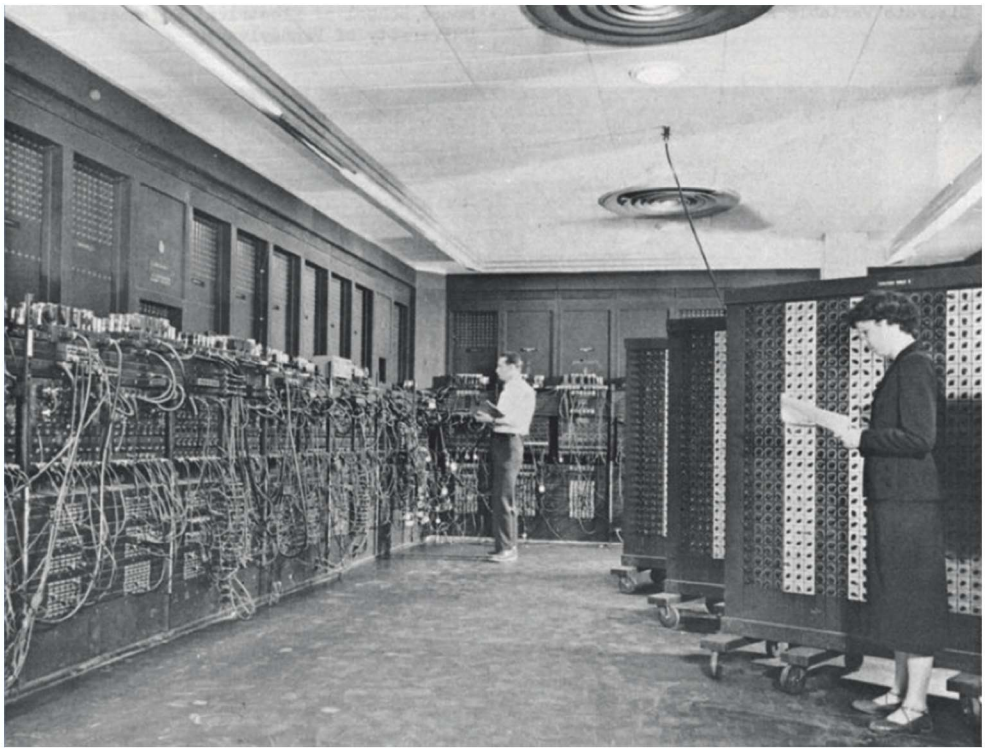
Информационо-комуникационе технологије

У овом поглављу сазнаћеш како да правилно користиш ИКТ уређаје, како да креираш и уређујеш дигиталне садржаје са тебелама у програму за рад са текстом и за рад са мултимедијалним презентацијама.

Научићеш како да креираш и обрађујеш дигиталне слике, знаћеш самостално да снимиш и обрадиш аудио и видео записе као и да их користиш при изради презентација.

У петом разреду научили сте да људи у 21. веку користе рачунаре у свим областима живота. Употребљавају их да би лакше и брже обавили неки посао или да би слушали музику и гледали филмове. Рачунар је присутан у свакодневом животу ученика, студената, полицајаца, војника, инжењера, наставника, лекара, банкара...

Рачунар треба схватити као алат који човеку омогућава лакши, једноставнији и ефикаснији рад. Први електронски рачунар појавио се 1946. године и назван је **ENIAC**. Од тада па до данас, рачунарска технологија се непрестано и убрзано развија. Први електронски рачунар приказан је на слици 1.1.



Слика 1.1. **ENIAC** – први електронски рачунар

Филип Драјфус (1962) је од енглеске речи **information** (инфомација) и француске **automatique** (аутоматски) створио појам информатика. Ова наука је настала из потребе овладавања новим знањима, техникама и технологијама у циљу бржег и свеобухватнијег разрешавања насталих друштвених проблема и противречности.

Информатика је наука о систематској и рационалној обради информација као носилаца људског знања и комуникација у техничком, економском и друштвеном контексту, првенствено помоћу аутоматских машина. Она подразумева прикупљање, обрађивање, чување и пружање информација кориснику.

Информациона технологија (**ИТ**) је технологија која користи рачунаре за прикупљање, обраду, чување, заштиту и пренос информација. Брз развој ИТ-а, умрежавање рачунара и нарочито повезивање рачунара и људи путем интернета допринели су томе да се ИТ-у придруже и комуникационе технологије. Тако је настао појам информационо- комуникационе технологије, на енглеском **Information and Communications Technology** – **ICT**. **Информационо-комуникационе технологије** (**ИКТ**) обухватају разноврсне технолошке алате и ресурсе који се користе за пренос, складиштење, стварање, дељење и размену информација.

Назив **ИКТ** користи се уопштено да означи све уређаје, мрежне компоненте, апликације, као и интернет, интернет сервисе и системе који комбиновано дозвољавају људима и организацијама да међусобно сарађују и комуницирају у дигиталном свету.

ИКТ повезују људе, организације и државе широм света као ниједна технологија до сада. Она се најбрже развија, а њене границе још увек нису до краја познате.

Информатички писмена особа користи ИКТ да пронађе информације, уме да процени њихову тачност, зна како да дође до поуздане информације, да је повеже са већ постојећим знањима и схвата значај тачне и комплетне информације за доношење исправних одлука.

Запамти

ИКТ се дефинише као комбинација информационих технологија са другим технологијама, посебно комуникационим технологијама. Информационо-комуникационе технологије (ИКТ) обухватају разноврсне технолошке алате и ресурсе који се користе за пренос, складиштење, стварање, дељење и размену информација.

1.2. Рачунарски систем

Кључне речи:
рачунарски систем, хардвер, софтвер

Рачунарски систем представља јединство **хардвера** и **софтвера**.

Хардвер (hardware) чине делови рачунара које можемо да додирнемо – то су физички делови рачунара: матична плоча, процесор, хард-диск, монитор, миш, тастатура, микрофон, звучници...

У табели 1.1. приказана је само једна од могућих подела хардвера, док су на слици-1.2. приказани поједини уређаји (миш, монитор, штампач и тастатура).

ХАРДВЕР

Улазни уређаји	Излазни уређаји	Уређаји за чување података
✓ Миш ✓ Тастатура ✓ Веб-камера ✓ Микрофон	✓ Штампач ✓ Монитор ✓ Звучници ✓ Пројектор	✓ Хард-диск ✓ Флеш меморија ✓ Оптички уређаји

Табела- 1.1. Подела хардвера

Слика 1.2. Делови рачунара – миш, монитор, штампач, тастатура

Софтвер (software) је скуп свих програма инсталираних на рачунару. Софтвер се дели на две основне групе програма:

– **системске програме,** који представљају скуп програма одговорних за контролу и управљање уређајима и рачунарским компонентама, као и за обављање основних системских радњи и
– **корисничке програме**, тј. оне програме које користимо у свакодневном раду за обављање одређеног задатка, на пример за писање текста, прављење табела, презентацију, музику, израду слика.

Кориснички програми могу бити комерцијални и бесплатни. Програми који се корисницима испоручују без надокнаде и ограничења и које корисник може да прилагођава и дорађује према сопственим потребама називају се **open source** програми.

Оперативни систем је скуп програма који управља радом рачунара. Данас се најчешће користи оперативни систем Microsoft Windows, који постоји у више различитих верзија као што су **Windows 7**, **Windows 8**, **Windows 10**. У употреби су и оперативни системи **Linux**, **Mac OS** итд. Паметни телефони најчешће користе **Андроид** и **IOS** оперативне системе. Логои различитих оперативних система приказани су на слици 1.3.

Слика 1.3. Лого различитих оперативних система

Запамти

Рачунарски систем представља јединство **хардвера** и **софтвера**.
Хардвер (hardware) чине физички делови рачунара.
Софтвер (software) је скуп свих програма инсталираних на рачунару.

1.3. Графичко радно окружење оперативног система

Кључне речи:

оперативни систем, графичко радно окружење

Оперативни систем **Windows** је графички оперативни систем који омогућава кориснику да задаје наредбе рачунару кликом на иконе (сличице) на радној површини. Иконе представљају програме или докумената која се чувају на рачунару.

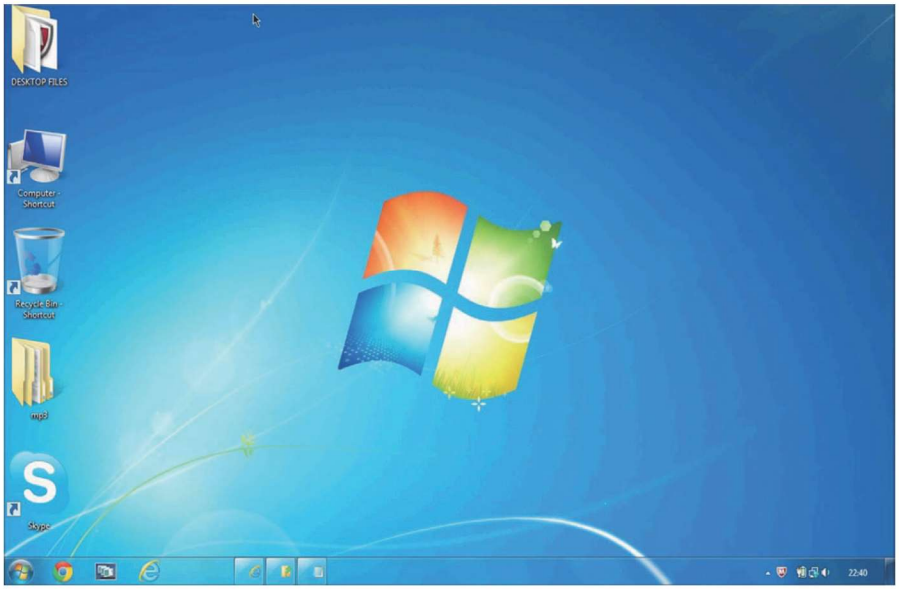
Битно олакшан рад у графичком окружењу довео је до брзог развоја различитих верзија **оперативног система Windows**: Windows 98, Windows Millenium, Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10...

Сматра се да је оперативни систем Windows инсталиран на око 80% персоналних рачунара, док мобилни телефони најчешће користе Android и IOS оперативне системе.

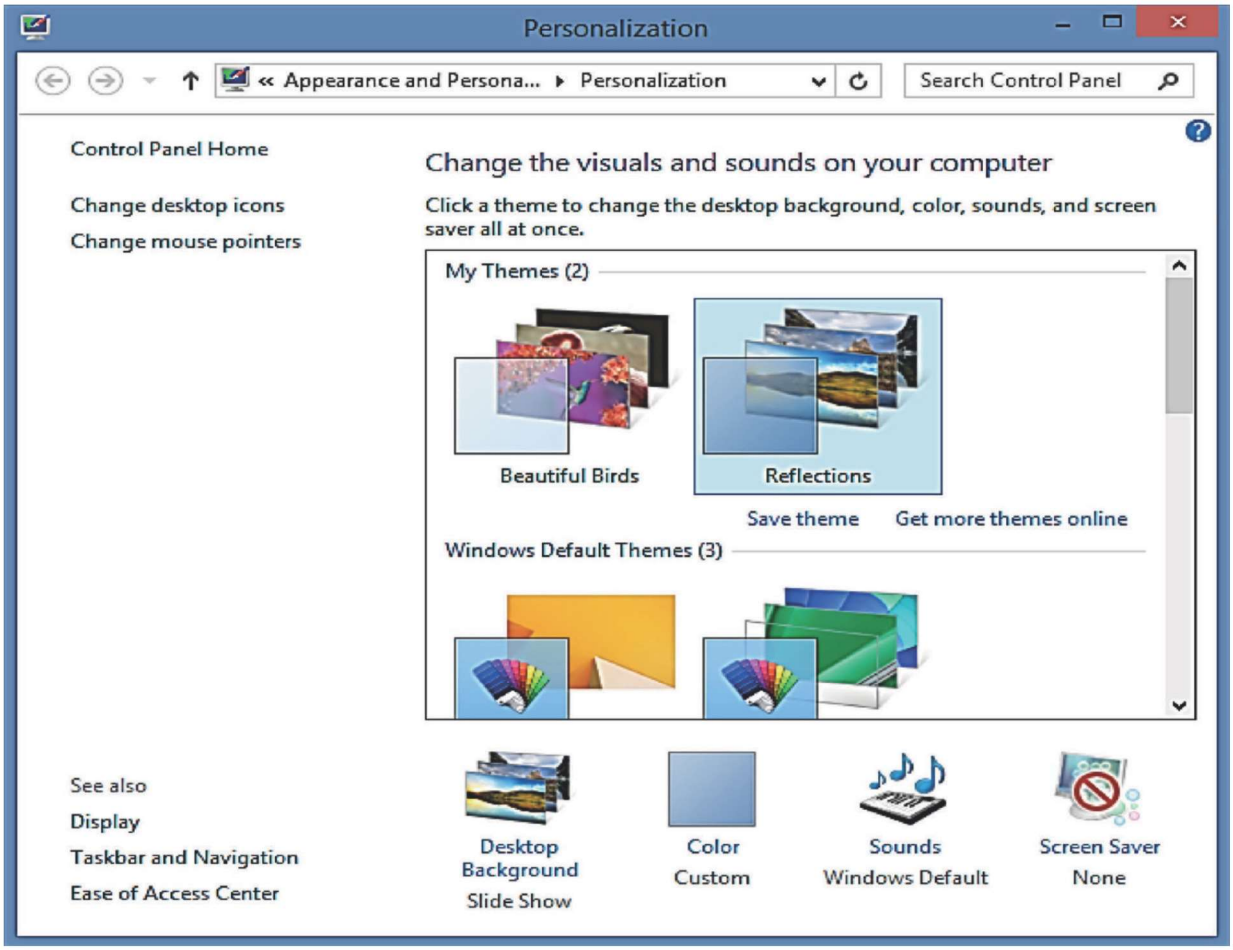
Непознате речи:

Персонализација
Оличење сваког појединца

Десктоп или радна површина је све оно што видимо на екрану када укључимо рачунар (сл. 1.4). Позадина, иконе и линија задатака саставни су део сваког десктопа. Позадина може бити једнобојна или прекривена неком сликом, иконица је сличица која визуелно представља неки програм или документ, док је линија задатака (Taskbar) линија која се налази на дну екрана, садржи дугме Start са леве стране и сличице које представљају покренуте програме. У зависности од афинитета и послова које обавља корисник сам одлучује шта жели да му се приказује на радној површини и како ће она изгледати. Ово је могуће постићи персонализацијом десктопа. Десним кликом миша на празан простор радне површине отварамо картицу **Personalization** (сл. 1.5), а у понуђеним опцијама бирамо ону која нам највише одговара.



Слика 1.4. Изглед радне површине – десктопа оперативног система **Windows 7**



Слика 1.5. Картица **Personalization**

Запамти

Оперативни систем Windows омогућава кориснику да задаје наредбе рачунару кликом на иконе (сличице) на радној површини или избором из листе са наредбама.
Десктоп или радна површина је све оно што видимо на екрану када укључимо рачунар.
Радно окружење је могуће прилагођавати потребама корисника.

1.4. Организација података на рачунару

Кључне речи:
датотека, фолдер, Windows explorer

Подаци у рачунарима чувају се у електронским документима који се називају датотеке, тј. **фајлови** (**File**).

Датотека (фајл) је именовани, уређени скуп података који су садржајно повезани и смештени на медију за меморисање.

Подаци меморисани у датотекама чувају се и након престанка рада одређеног програма коме припадају.

Сваки фајл има свој **назив** и **екстензију**. Назив фајла и екстензија раздвојени су тачком. На пример: pera.txt, где је pera назив, а .txt екстензија.

Непознате речи:
Екстензија
Додатак у називу датотеке који указује на врсту њеног садржаја

Назив може бити било који низ слова и бројева. Могу да се користе и латиничко и ћириличко писмо.

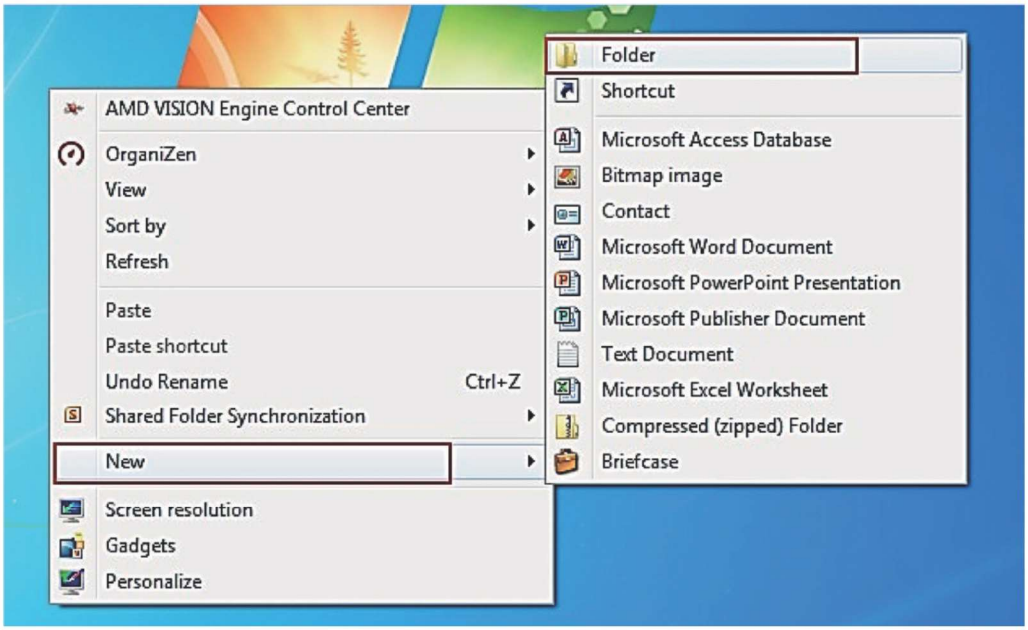
У називу се не могу користити следећи знаци: ? * < > / \ : ” |

Екстензија представља додатак на име датотеке и указује на врсту података.

– екстензије у вези са текстуалним фајловима: txt, rtf, doc, docx;
– екстензије у вези са видео фајловима: avi, wmv, mkv, 3gp;
– екстензије у вези са сликовним фајловима: bmp, jpg, gif, png, tif;
– екстензије у вези са аудио фајловима: mp3, wav, mid, wma.

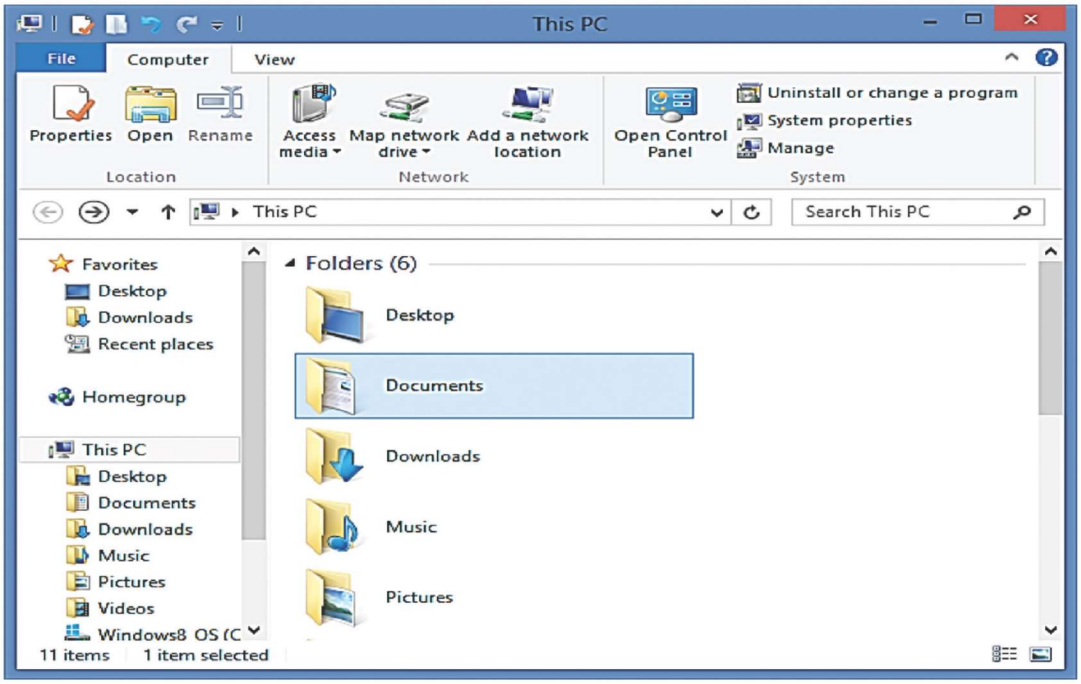
Фајлови се даље организују у **фолдере** (**Folder**), тј. фасцикле. За фасцикле се користе још и називи **директоријуми** или **каталози**. У фолдеру може да се налази један или више фајлова, а могуће је и да се у оквиру једног фолдера нађе још један фолдер или неколико других фолдера, тј. **потфолдера.**

У петом разреду учили сте како се креира нови фолдер и тај поступак приказаћемо на слици 1.6.



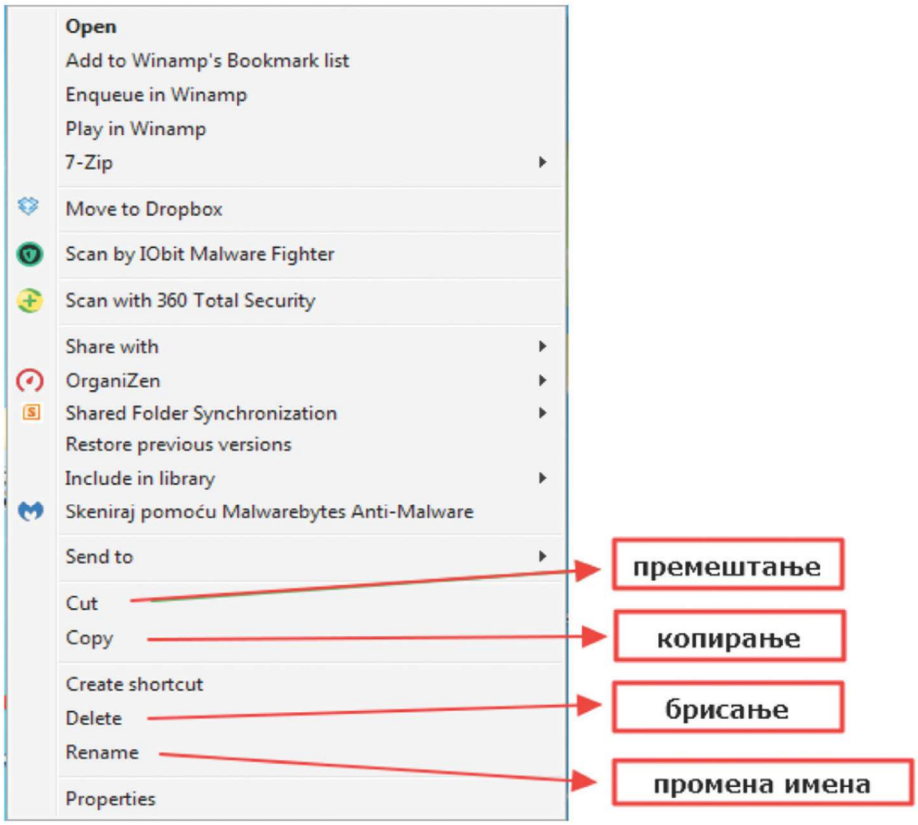
Слика 1.6. Креирање новог фолдера

Windows Explorer је програм који се користи од стране оперативног система Windows за организовање датотека и фолдера, сл. 1.7.



Слика 1.7. Организација података на рачунару

Датотеке је могуће копирати с једног места на друго место у рачунару, премештати их, брисати и преименовати; најједноставнији начин за извршавање тих операција је обележавање датотеке, и то тако што се десним кликом миша изабере нека од поменутих операција из контекстног менија (сл. 1.8).



Слика 1.8. Контекстни мени

Запамти

Датотека (фајл) је именовани, уређени скуп података који су садржајно повезани и смештени на медију за меморисање.
Сваки фајл има свој **назив** и **екстензију**.
Фајлови се организују у **фолдере,** тј. фасцикле.
Windows Explorer је програм који се користи од стране оперативног система Windows за организовање датотека и фолдера.

Вежбе и питања за проверу стеченог знања

- ✓ Креирај на радној површини два фолдера, Вежба 1 и Вежба 2.
- ✓ Фолдер Вежба 1 преименуј у Припрема 1.
- ✓ Обриши фолдер Вежба 2.
- ✓ Објасни чему служи Windows Explorer.
- ✓ Шта означава екстензија фајла?
- ✓ Објасни разлику између поступка премештања и поступка копирања фолдера.

1.5. Рад са текстуалним документима

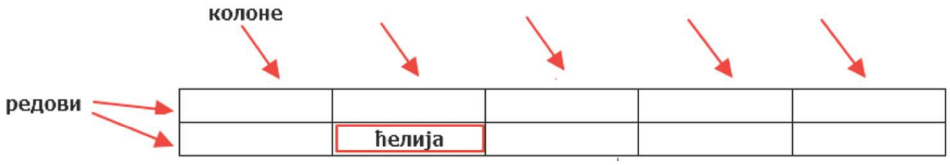
Кључне речи:
табела, ћелија, форматирање

Од свих програма, најчешће и највише се користе програми за обраду текста. Најпознатији и најзаступљенији је програм **Microsoft Word**. Представићемо верзију Microsoft Word 2013 .

Од раније ти је познато како се текст уноси, мења и исправља ако је потребно, како се форматира (обликује) на одређен начин и како га је могуће сачувати, а у шестом разреду научићеш да податке представљаш табеларно.

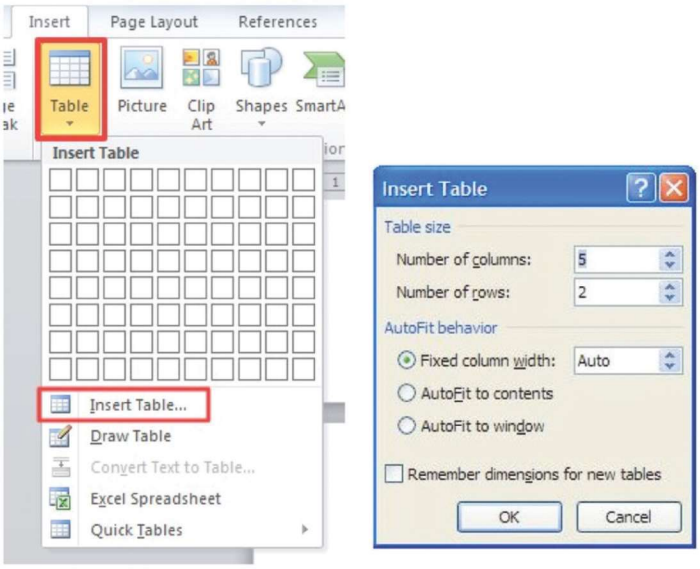
Креирање и форматирање табела

Сваку табелу карактерише одређени број **редова** и **колона**. Пресек реда и колоне представља **ћелију** у коју је могуће унети одређене податке (текстуалне, нумеричке), сл. 1.9.



Слика 1.9. Приказ табеле

Једноставан начин за прављење табеле подразумева да се у оквиру картице **Insert** изабере алат за уметање табела **Table / Insert table** и након тога број редова и колона. На слици 1.10. приказано је креирање табеле.



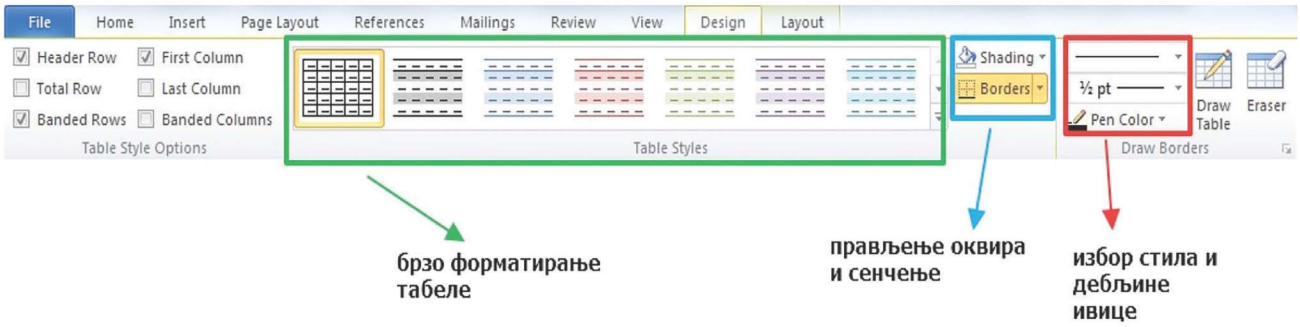
Слика 1.10. Креирање табеле

Направљену табелу је могуће даље **форматирати**, тј. **уређивати**: одредити боју и тип линија оквира, обликовати текст унутар ћелија, додавати или брисати редове и колоне.

Слика 1.11. Картице **Design** и **Layout**

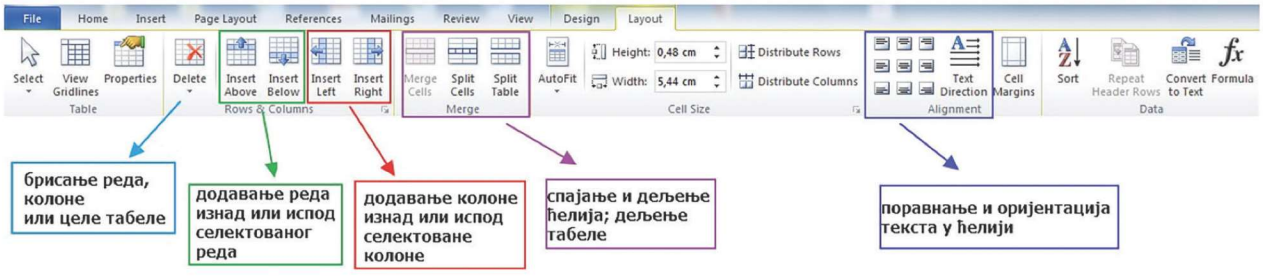
За рад са табелама, у пракси се често користе још две додатне картице **Design** и **Layout**, сл. 1.11.

Картице се појављују тек када је курсор позициониран у табели. Уколико је курсор изван табеле, картице неће бити доступне. На следећим сликама приказаћемо могуће опције из катрице **Design** (сл.1.12.) и картице **Layout** (сл.1.13).



Слика 1.12. Доступне опције из картице **Design**

Често се дешава да је потребно додати или избрисати неки ред или колону, делити или спајати ћелије, поравнати текст унутар табеле. Најлакши начин да се то уради јесте да селектујемно ред или колону над којом желимо да извршимо неку промену и изаберемо жељену опцију из картице Layout (сл.1.13). Важно је напоменути да показивач миша има карактеристичан облик приликом селектовања редова (стрелица налево), колона (стрелица надоле) а да приликом обележавања целе табеле курсор поприма облик крстића.



Слика 1.13. Доступне опције из картице **Layout**

Запамти

Сваку табелу карактерише одређени број редова и колона.
Пресек реда и колоне представља ћелију.
Направљену табелу је могуће даље форматирати, тј. уређивати.

Вежбе и питања за проверу стеченог знања

- ✓ Креирај табелу која има четири реда и пет колона.
- ✓ Направи табелу која приказује твој распоред часова.

Користан линк

https://www.youtube.com/watch?v=I1JkfpQ10rg

1.6. Рачунарска графика

Кључне речи:

рачунарска графика, резолуција, векторска и растерска графика

Област визуелног рачунарства која омогућава креирање слика и њихову обраду назива се рачунарска графика.

Два основна начина за приказ и представљање графике у рачунару јесу **векторски и растерски** начин.

Како ће слика бити представљена у рачунару зависи од програма у ком се слика ствара и обрађује.

Код програма који **векторски** представљају слике, рачунар памти елементе слике и њихове особине. Количина података коју треба запамптити зависи од сложености слике.

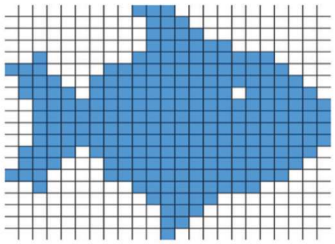
Векторски програми који се најчешће користе јесу: **CorelDRAW, Adobe Illustrator, Inkscape, Karbon** (сл. 1.14. приказује лого векторског програма Inkscape i CorelDraw).

Слика 1.14. Лого векторских програма Inkscape i CorelDraw

Програми који растерски приказују слике деле слику на мрежу квадратића. Ти квадратићи називају се пиксели (piksel – од речи picture и elements), а рачунар памти особине сваког пиксела посебно. Пиксел на монитору има свој положај, боју и интензитет или дубину боје. Од броја пиксела и интензитета боје зависи и квалитет растерске слике.

Растерски пограми који се најчешће користе јесу: **Microsoft Paint, Adobe Photoshop, Macromedia Fireworks, Microsoft Photo Editor, GIMP, CinePaint, Picasa** и др. (сл. 1.15. приказује лого растерских програма Photoshop i MS Paint). На слици 1.16. приказана је слика у растеру.

Слика 1.15. Лого растерсих програма Photoshop і MS Paint Слика



1.16. Растерски приказ слике

Векторска графика се темељи на математици, тј. користи математику за цртање облика помоћу тачака, правих и кривих линија. Њене предности се огледају у томе што приказ можемо бесконачно увећати или умањити, а квалитет се неће променити. Једна од значајних предности јесте и та да је величина њеног фајла мања у односу на растерску графику, будући да садржи много мању количину информација.

Растерска графика се темељи на пикселима. Предности растерске графике у односу на векторску графику јесу у томе што она садржи много више детаља, па можемо вршити прецизне измене (боје) на нивоу сваког појединачног пиксела.

Резолуција представља број пиксела по хоризонтали и вертикали. Што је резолуција већа бољи је квалитет слике. Начин на који је цртеж снимљен у датотеку зове се формат података.

Постоје технике чувања слика у рачунару које применом разних алгоритама штеде меморију потребну за њихово чување. Процес сажимања података, односно претварање података у облик који заузима мање меморијског простора назива се компресија.

Најзначајнији формати за чување дигиталних слика јесу: BMP (bitmap), GIF (graphics interchange format), JPG или JPEG (joint photographers experts group), SVG (scalable vector graphics), PNG (portable network graphics), TIFF (tagged image file format).

Елементи и принципи графике

Рачунарска графика има огромну примену у дизајну, па је неопходно да се упознамо са њеним основним елементима и принципима.

Основне елементе графике чине: линија, облик, величина, боја, итд., док основне принципе графике чине: баланс боја, њихов контраст, пропорција, итд. Најбитније и основне елементе и принципе графике приказаћемо у табели 1.2.

Основни елементи и принципи графике

Елементи графике	Принципи графике
✓ линија ✓ облик ✓ величина ✓ простор ✓ боја ✓ текстура ✓ вредност	✓ баланс ✓ контраст ✓ наглашеност ✓ пропорција ✓ узорак ✓ градација ✓ композиција

Табела 1.2. Елементи и принципи графике

Извори дигиталних слика

Дигиталне слике углавном настају цртањем у неком од рачунарских програма.

Програми за израду дигиталних слика могу да учитавају већ постојеће слике и да их даље обрађују.

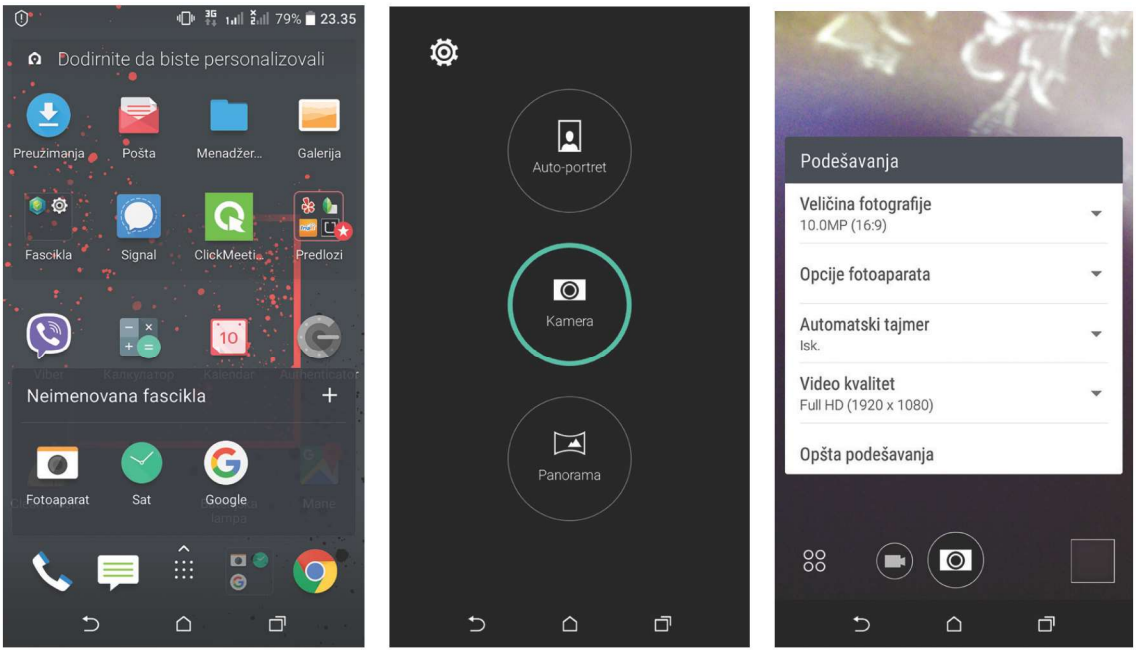
Дигиталним фото-апаратом (сл. 1.17) је прилично лако руковати а представља један од извора дигиталних слика које се бележе у неком од познатих формата и погодне су за даљу обраду.

Слика 1.17. Дигитални фотоапарат

Слика 1.18. Скенер

Скенер је уређај намењен преношењу садржаја са папира у дигитални облик (сл. 1.18). **Скенирање** је поступак при ком се слика с папира преноси на рачунар и тако добијену слику могуће је даље едитовати.

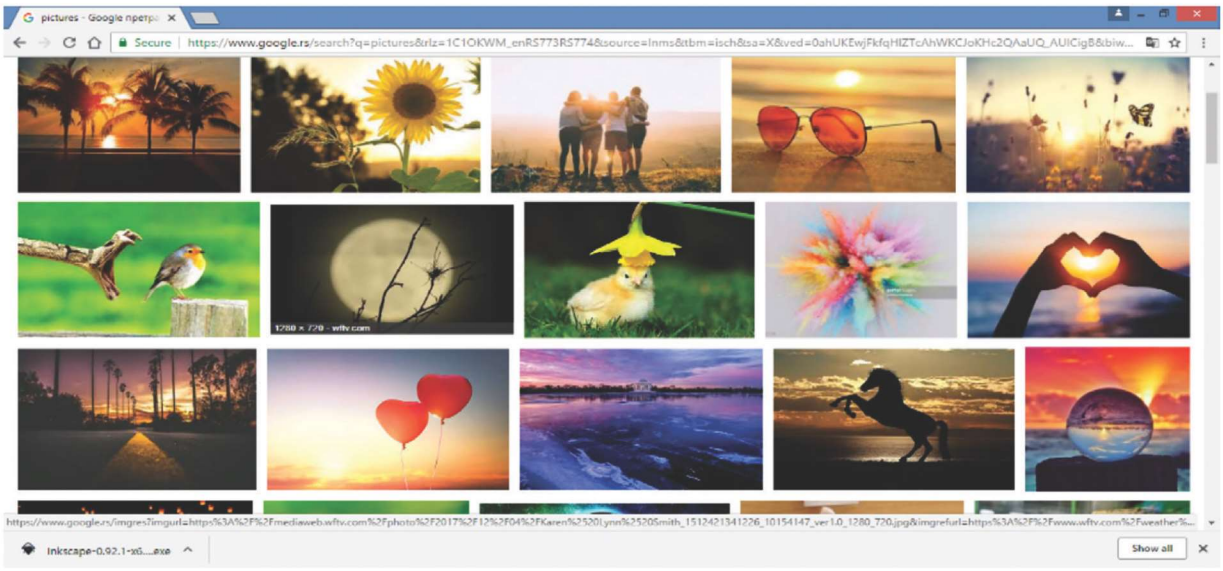
Мобилни телефони, односно камере мобилних телефона сваким даном су све квалитетније. Резолуција фотографија је све већа, а самим тим, повећава се и могућност стварања лепих дигиталних фотографија на крајње једноставан начин (сл. 1.19).



1.19. Фотогрфисање мобилним телефоном

Преузимање слика са интернета

Слике других аутора настале цртањем, фотографисањем или скенирањем могуће је преузети са интернета при чему треба водити рачуна о ауторским правима (сл. 1.20).



Слика 1.20. Претрага слика на интернету

Запамти

Област визуелног рачунарства која омогућава креирање слика и њихову обраду назива се **рачунарска графика**.
Два основна начина за представљање графике у рачунару су: векторски и растерски.
Резолуција представља број пиксела по хоризонтали и вертикали. Што је резолуција већа, бољи је квалитет слике.
Рачунарска графика има огромну примену у дизајну.
Дигитални фото-апарат и скенер могу бити извори дигиталних слика.

Вежбе и питања за проверу стеченог знања

- ✓ Направи неколико фотографија дигиталним фото-апаратом или мобилним телефоном.
- ✓ Направљене слике пребаци у фолдер Слике на десктопу.
- ✓ Пронађи на интернету слике свог града, одабери три, преузми их и сачувај у фолдеру Слике на десктопу.
- ✓ Креирај табелу која има три колоне и потребан број редова за твоје слике.
- ✓ У сваку ћелију табеле убаци по једну слику из фолдера Слике.

Корисни линкови

https:/inkscape.org/en
https://coreldraw.com/en/free-trials/?topNav=en

1.7. Програм Inkscape

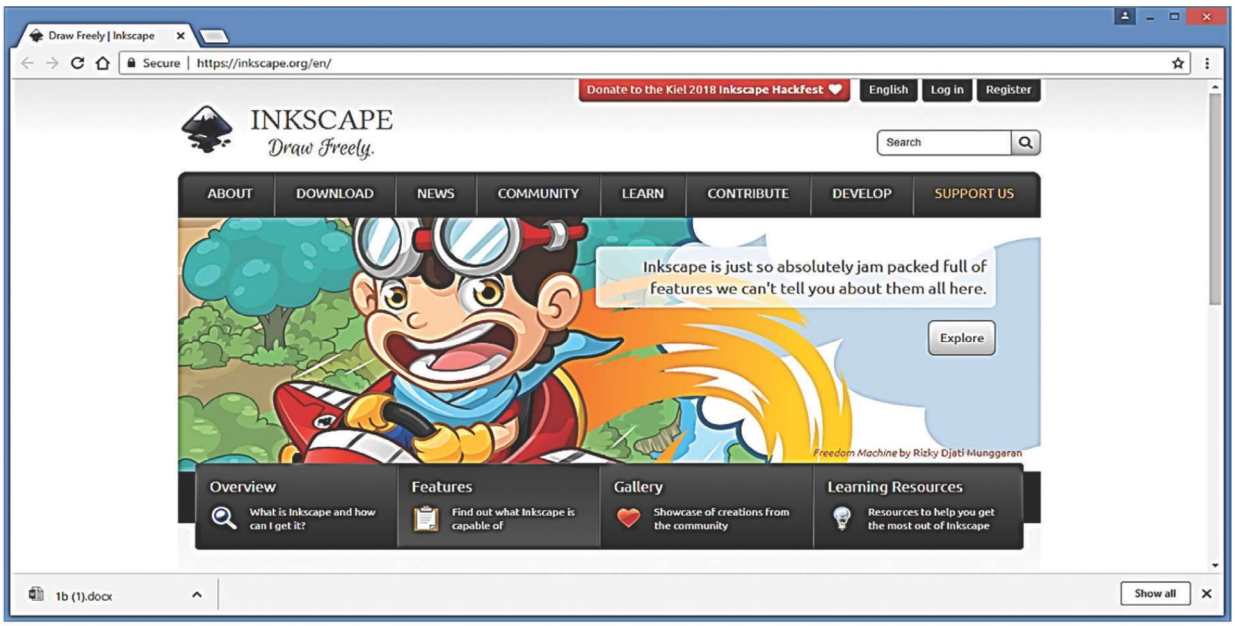
Кључне речи:
програм Inkscape, векторска графика

Типични представник програма заснованог на векторској графици је програм **Inkscape**.

Програм је бесплатан, даје велики број могућности у раду, врло је једноставан и лак за коришћење.

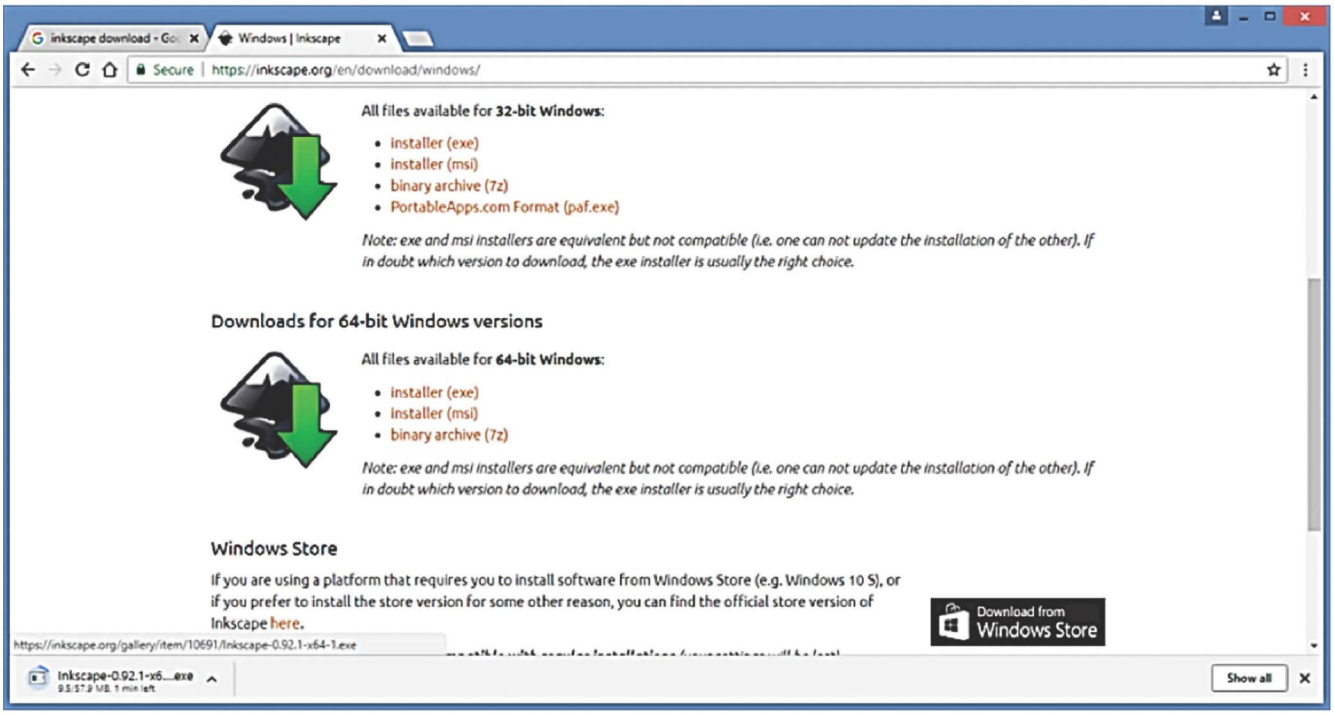
Инсталација програма

Програм **Inkscape** бесплатно преузми са веб-странице https://inkscape.org/en/. Кад посетиш веб-страницу, појавиће се приказ као на слици 1.21.

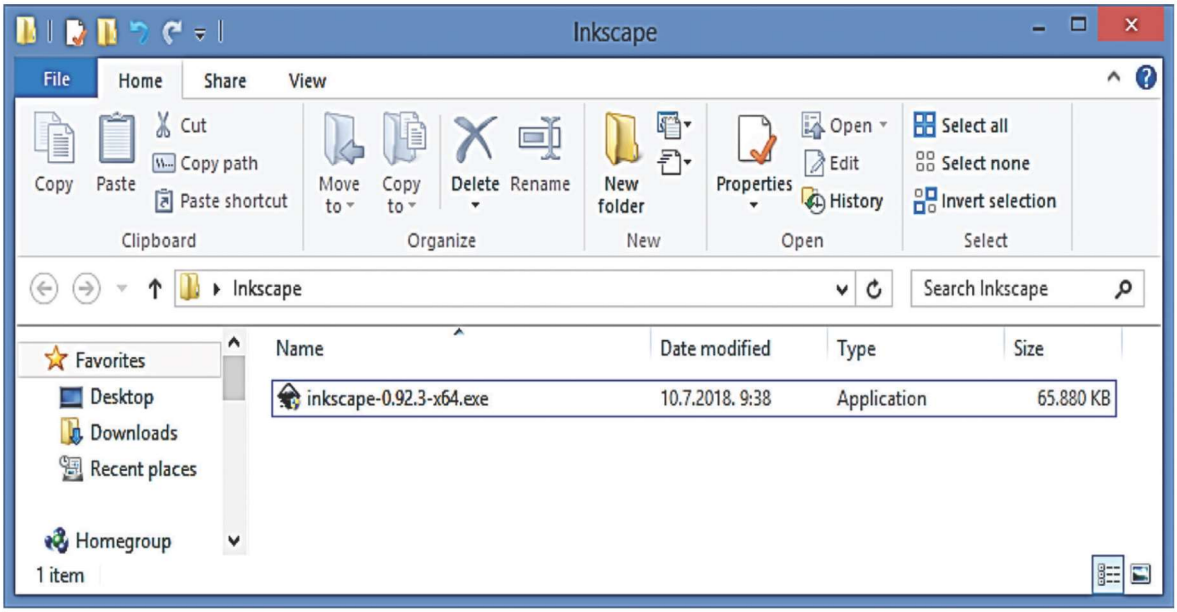


Слика 1.21. Изглед веб-странице с које преузимамо програм

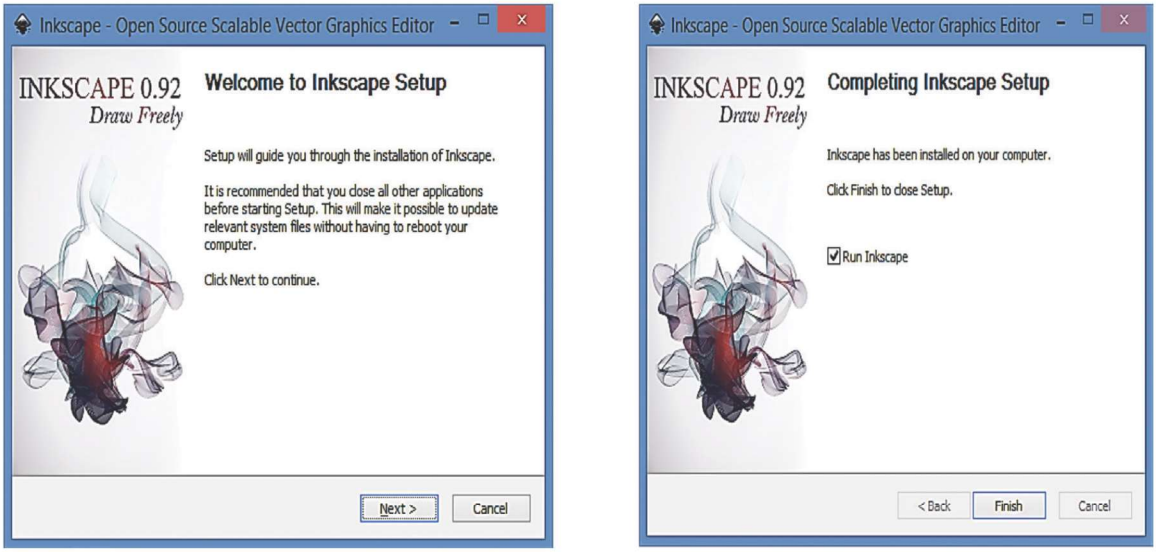
У зависности од тога који оперативни систем имаш инсталиран на свом рачунару, преузимаш одговарајућу верзију програма, коју потом инсталираш покретањем извршног (.exe) фајла (сл. 1.22, сл. 1.23. и сл. 1.24).



Слика 1.22. Преузимање одговарајуће верзије програма

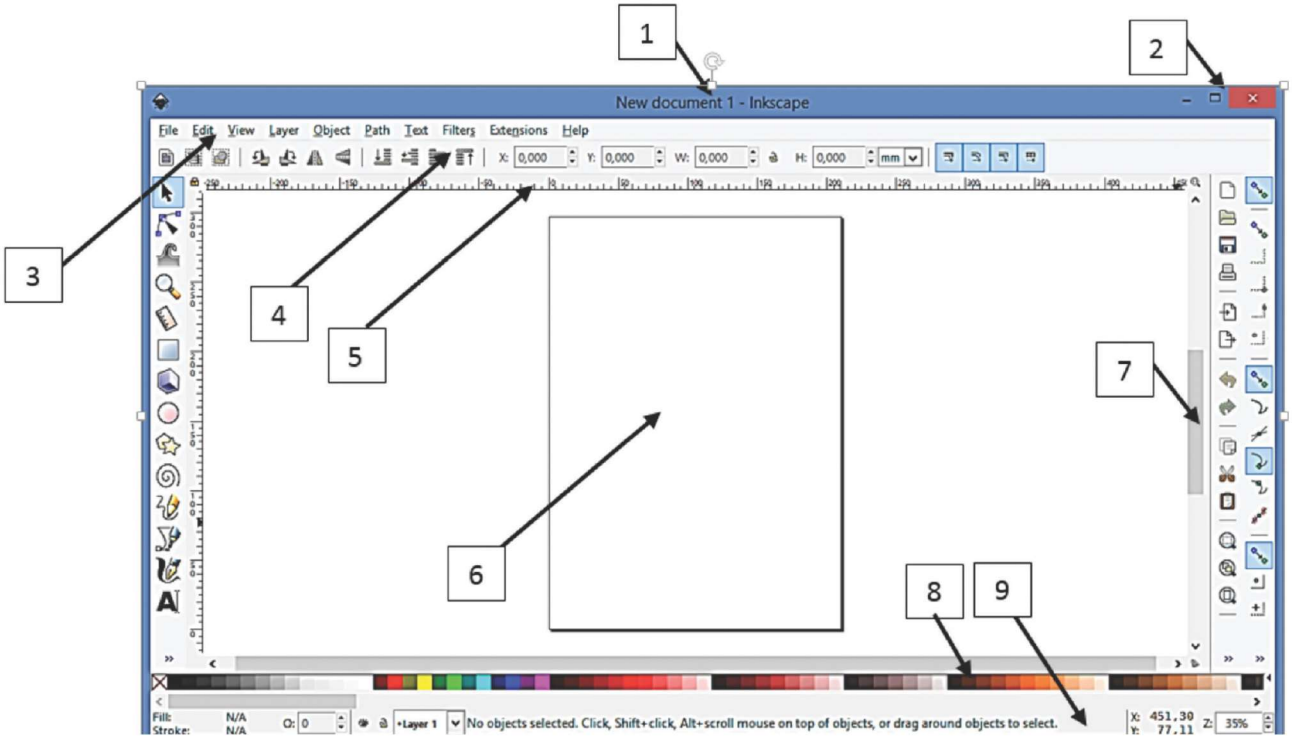


Слика 1.23. Инсталациони фајл **Inkscape**-0.92.3-x64.exe



Слика 1.24. Почетак и крај инсталације програма

Након завршене инсталације програма, на десктопу се појављује пречица, двоструким кликом на њу покрећемо програм. У стандардном прозору уочавамо насловну линију са именом документа (1), тастере за промену величине прозора и излаз из програма (2), линију менија (3), палете са алатима (4), лењире (5), површину за цртање видљиву при штампању цртежа „папир" (6), скрол траке (7), палету са бојама(8), статусну линију (9)... Сви ти елементи чине радни прозор **Inkscape**-а (сл. 1.25).



Слика 1.25. Радни прозор **Inkscape**-а

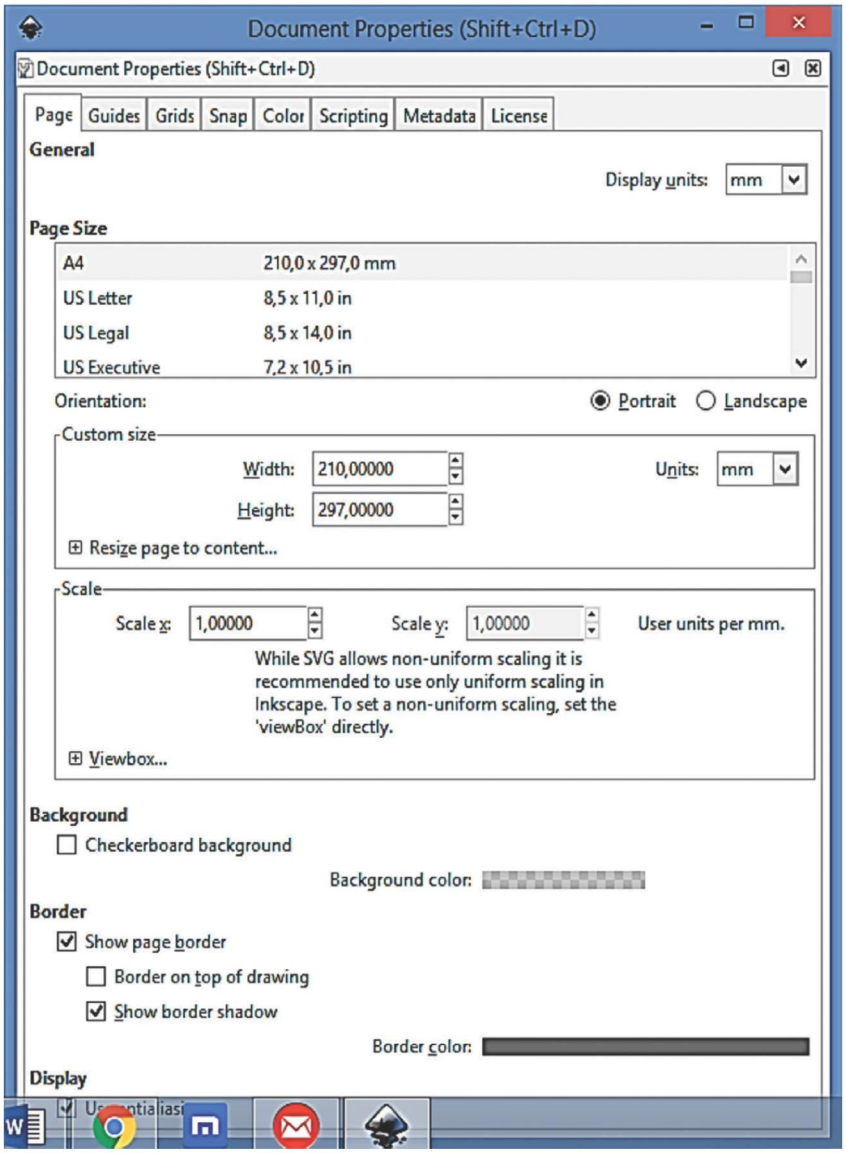
У радном прозору **Inkscape**-а могуће је цртати и по површини која се налази ван површине "папира" али те делове цртежа није могуће одштампати.

Курсор може да има различите облике; уколико селектујемо неки објекат имаће облик стрелице, али ако користимо неки од алата курсор ће имати облик примењеног алата.

Палете са алатима могу бити распоређене на различитим местима на радној површини.

Пре самог цртања добро је подесити радно окружење према сопственим потребама. Такође, важно је одмах изабрати величину и оријентацију папира.

Подешавање се врши бирањем менија **File/ Document properties**. У том оквиру за дијалог подешавају се и једнице мере за дужину. Након избора јединице, промена се извршава и на лењирима који онда показују вредности у m, cm, mm или у инчима и пикселима (сл. 1.26).

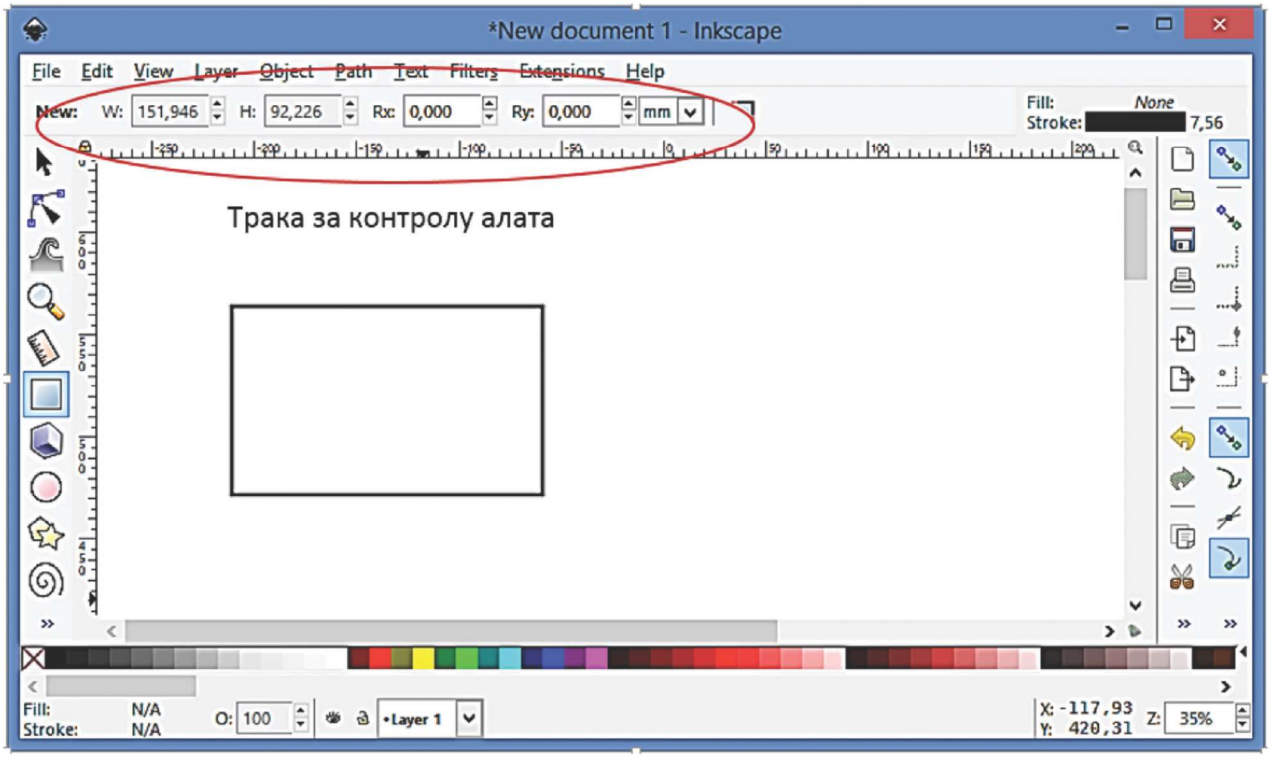


Слика 1.26. Поставке документа у **Inkscape**-у

Цртање основних графичких елемената

Правоугаоник и квадрат

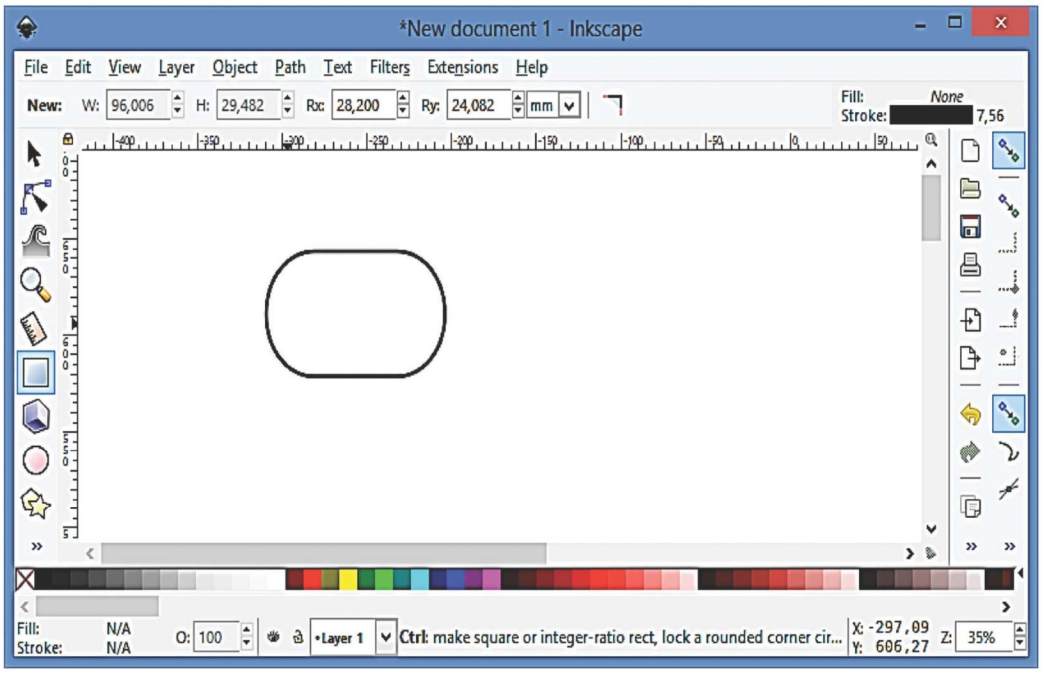
Правоугаоник и квадрат цртају се помоћу алата за цртање правоуганика  . Кад изаберемо тај алат, показивач миша добија облик правоуганика са знаком + . Превлачењем преко радне површине добијамо жељени објекат (сл. 1.27). На статусној линији добијамо корисна обавештења, као и помоћ при цртању (сл. 1.28).



Слика 1.27. .Цртање правоуганика

Слика 1.28. Статусна линија при цртању правоугаоника

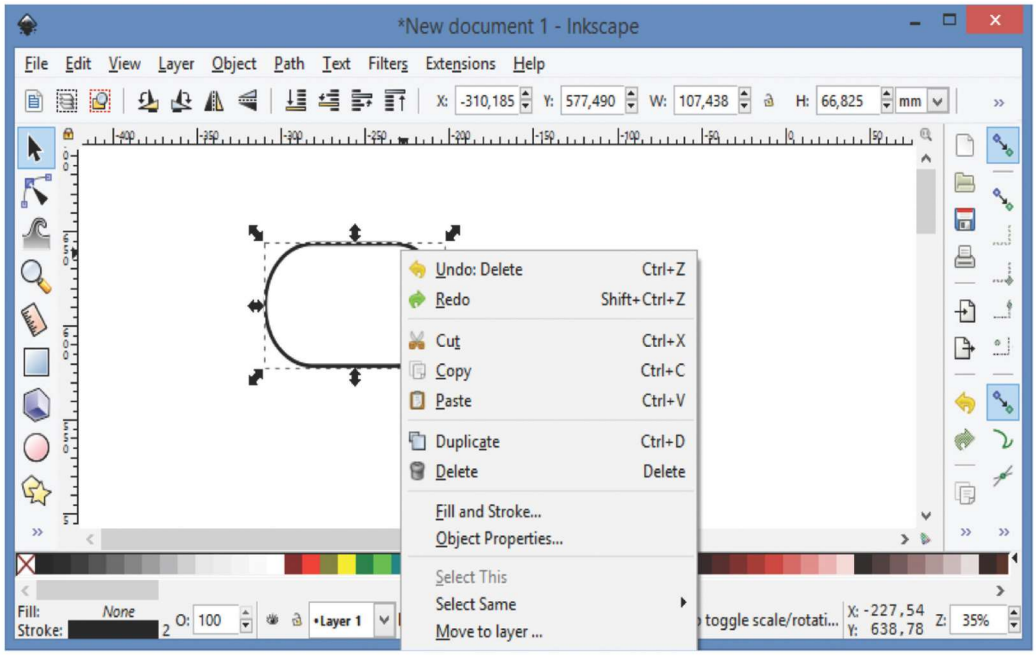
У траци контроле алата која се налази изнад хоризонталног лењира добијамо информацију да ли се црта нови објекат или се мења већ нацртани. У поља за ширину и висину правоуганика уписане су димензије правоуганика, које је увек могуће изменити, уколико се укаже потреба за променом. У продужетку траке за контролу алата налазе се поља с полупречницима заобљења углова по X- оси (Rx) и Y- оси (Ry), као и изабрана јединица мере. На крају ове траке налази се алат који заобљене углове претвара у оштре. Са десне стране ове траке приказане су и информације о томе којом бојом је испуњена унутрашњост правоугаоника, каква је линија којом је правоугаоник нацртан и дебљина линије у пикселима (сл. 1.29).



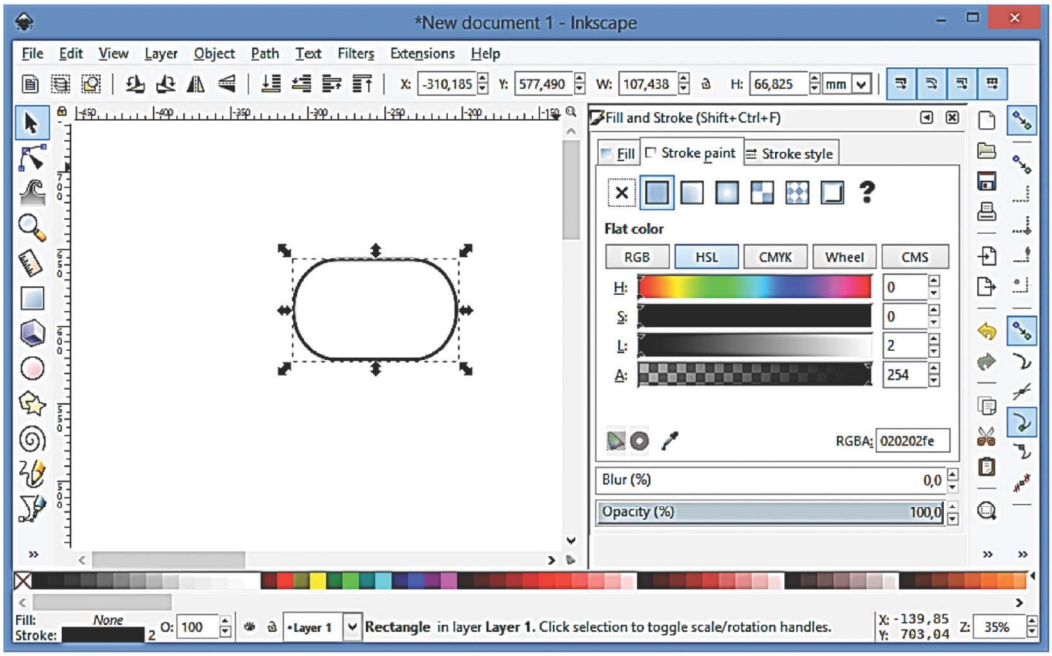
Слика 1.29. Измењен правоугаоник са заобљеним ивицама

Квадрат се црта на идентичан начин као и правоугаоник, само је потребно држати притиснут тастер **Ctrl** и истовремено превлачити курсор преко радне површине.

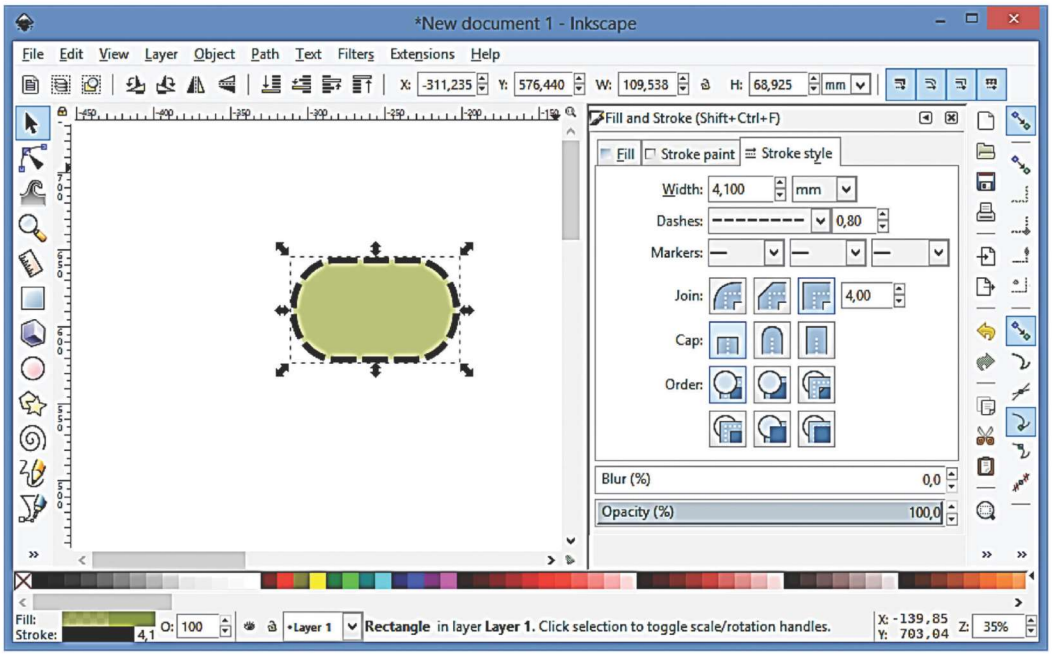
Уколико желимо да променимо боју линије, унутрашњости или дебљину и врсту линије на цртежу, објекат обележимо алатом за селектовање који има облик стрелице, отворимо контекстни мени и изаберемо опцију **Fill** and **Stroke** (сл. 1.30, сл. 1.31. и сл. 1.32).



Слика 1.30. Отварање контекстног менија **Object Properties**



Слика 1.31. Подешавање унутрашњег поља објекта и линија

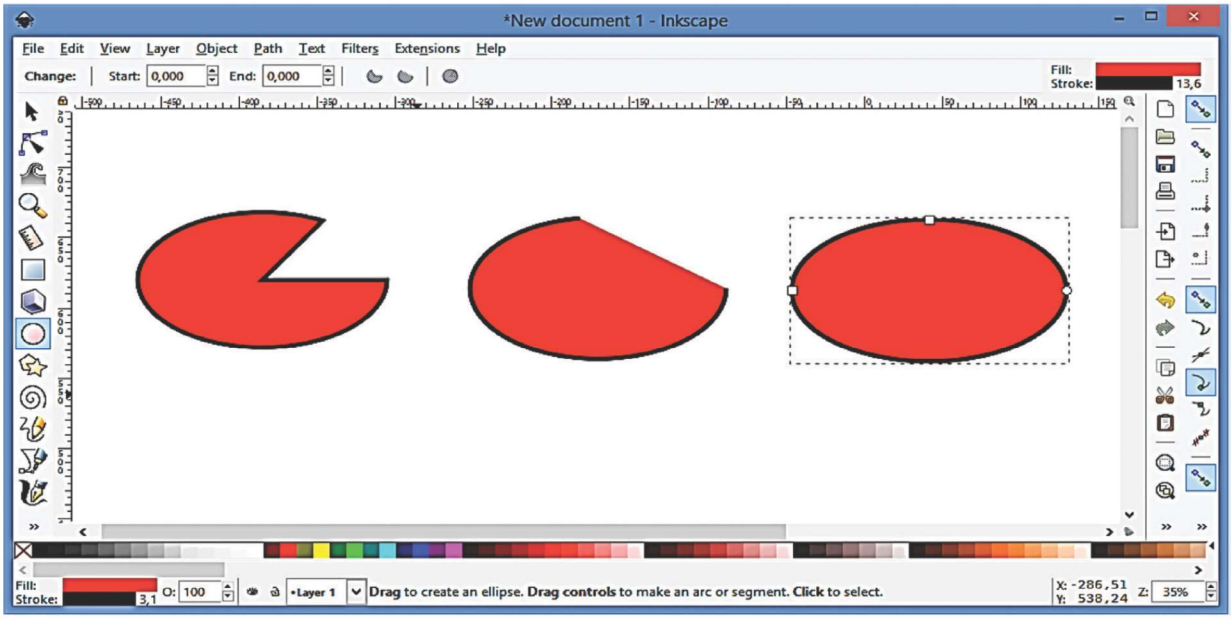


Слика 1.32. Измењен правоугаоник

У отвореном оквиру за дијалог картица **Fill** служи за подешавање боје унутрашњости објекта; у картици **Stroke paint** подешавамо боју линије, а картица **Stroke style** служи за подешавање стила линије. **Blur** означава замућеност боје објекта, а **Opacity** провидност, односно непровидност објекта. Уколико је непровидност подешена на 0%, нацртани објекат се неће видети. Нацртани објекат се брише притиском на тастер **Delete**, а претходно га је потребно обележити. Наредбе **Cut**, **Copy** и **Paste** функционишу на исти начин као и у свим осталим програмима које смо до сада обрађивали.

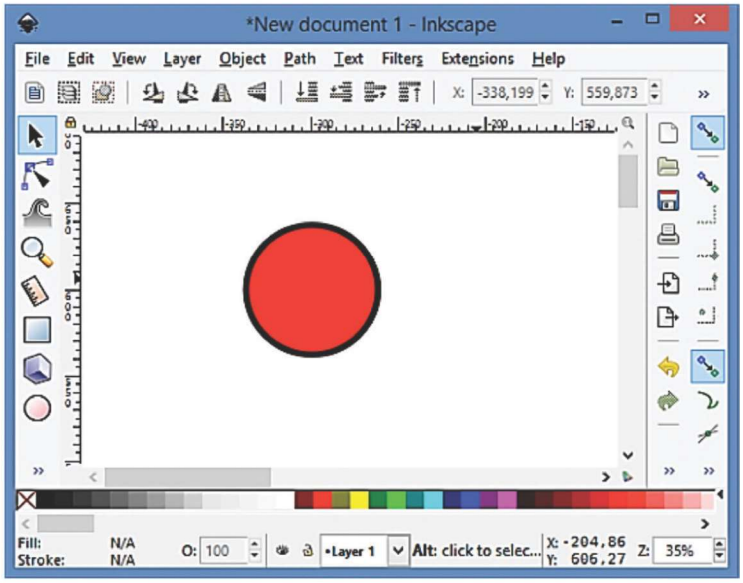
Елипса, круг, лук

Алатка за прављење кружнице, елипсе и лукова  функционише слично као и алат за цртање правоугаоника и квадрата. Изаберемо алат и превлачењем тог алата преко радне површине добијамо облик објекта. У траци за контролу алата добијамо информацију да ли се црта нов објекат или мења већ нацртан, а у продужетку се налазе опције за промену. Прва од њих мења елипсу у сегмент  , друга пребацује у лук  и трећа затвара линију у елипсу  (сл.1.33).



Слика 1.33. Коришћење алата за цртање елипсе

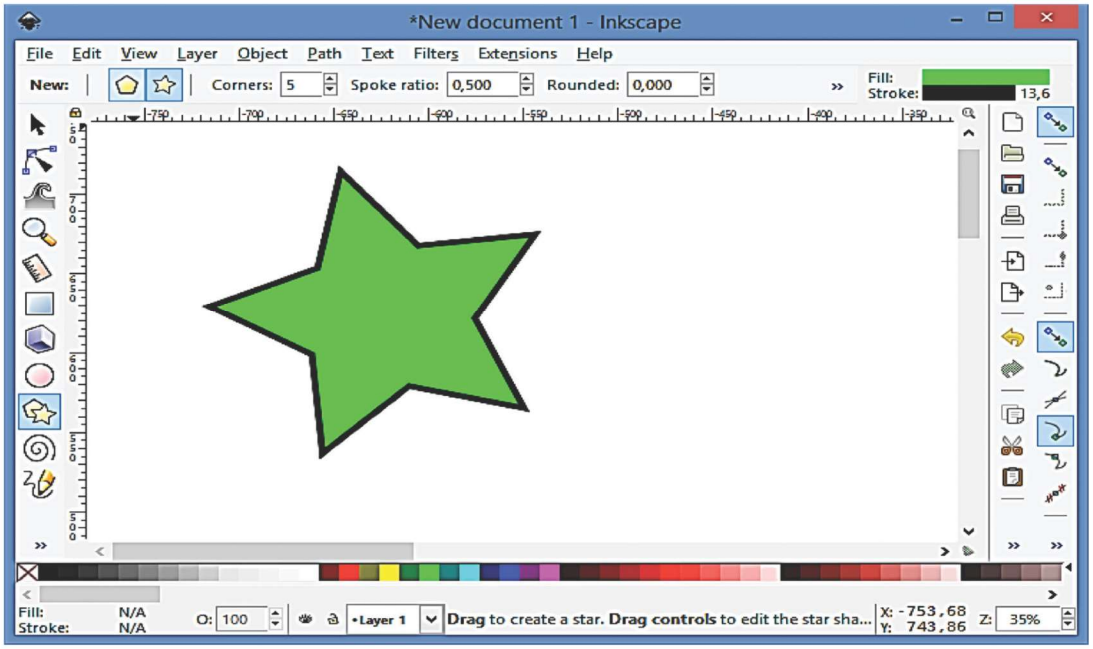
Користећи алат за цртање кружница, елипса и лукова могуће је нацртати круг и то тако што ћемо истовремено држати притиснут тастер **Ctrl** и превлачити курсор преко радне површине (сл.1.34).



Слика1.34. Цртање круга

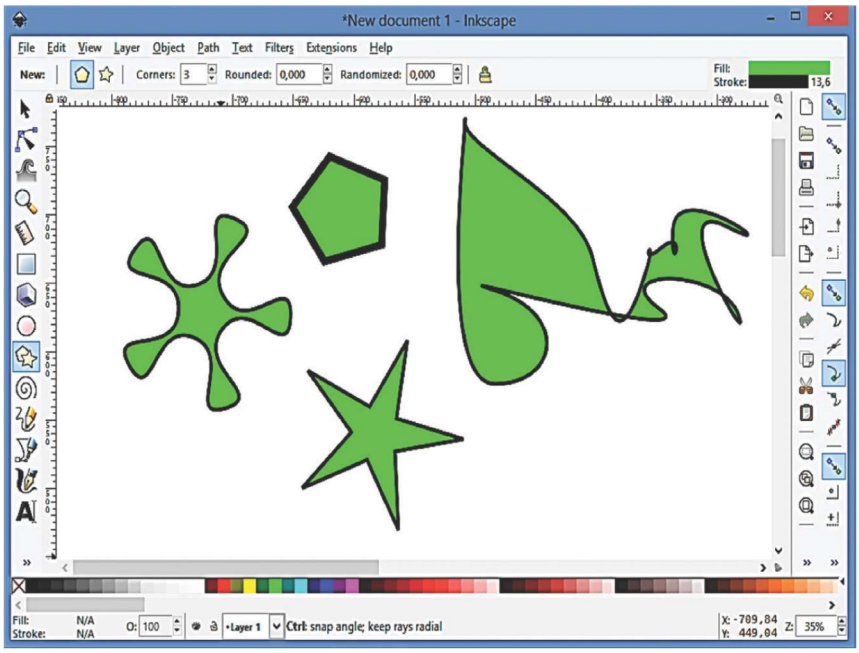
Полигон и звезда

За цртање полигона и звезде користи се исти алат  , а у траци за контролу алата бирамо који облик желимо, подешавамо однос кракова звезде или број страна полигона (сл. 1.35).



Слика 1.35. Коришћење алата за цртање звезде или полигона

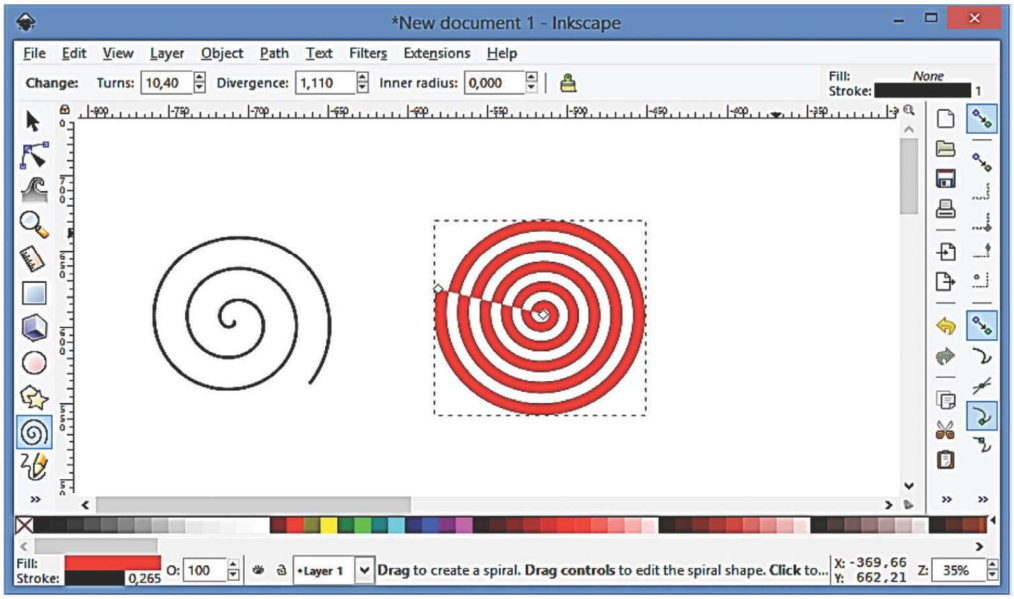
Кад у траци за контролу алата извршимо промену вредности параметара у пољу за број страна, заобљеност, као и произвољне промене, добија се могућност цртања најразличитијих облика (сл. 1.36).



Слика 1.36. Различити облици добијени коришћењем алата за цртање полигона и звезде

Цртање спирале

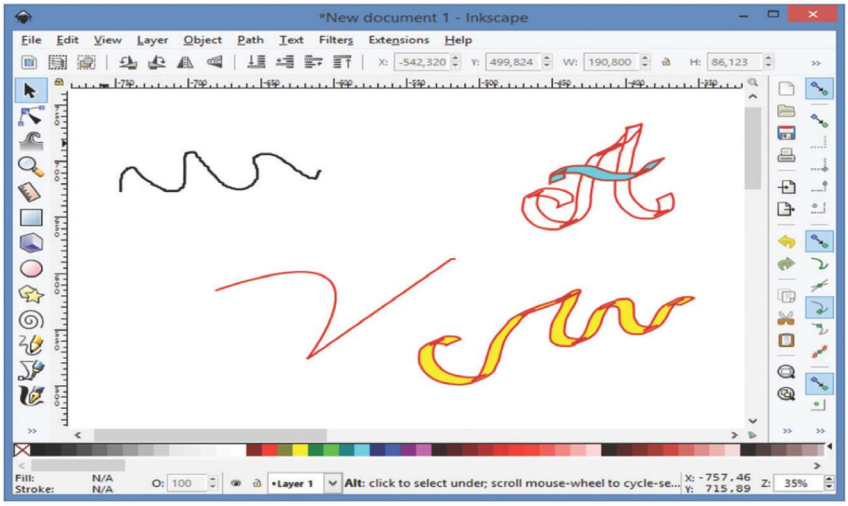
Избором алата за цртање спирале  показивач миша добија облик спирале. Активирањем тог алата, активира се и трака за контролу алата у којој је могуће подесити број завоја спирале, одступања, као и унутрашњи полупречник спирале. Подешавања у вези с линијама, бојом унутрашњости, типом линија итд. идентична су као код претходно објашњених алата (сл. 1.37).



Слика 1.37. Коришћење алата за цртање спирала

Цртање линија

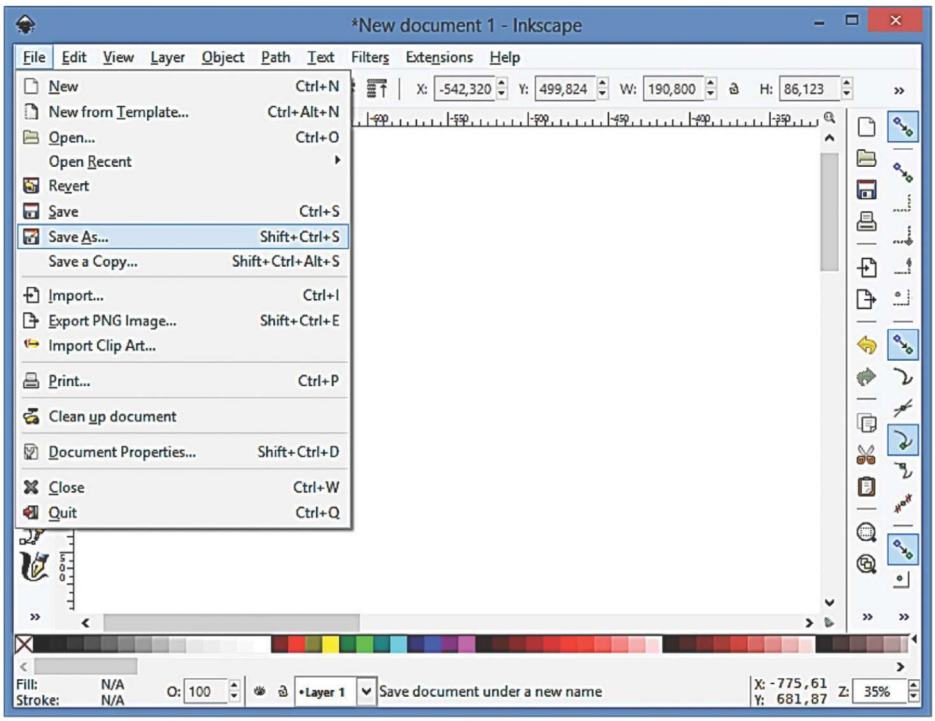
Програм **Inkscape** поседује три алата за цртање линија: алат за цртање слободом руком  , алат за цртање Безиерове криве  , односно параметарске криве која се задаје одређеним бројем контролних тачака, чијим се померањем мења облик криве и алат за цртање калиографских линија  (сл. 1.38).



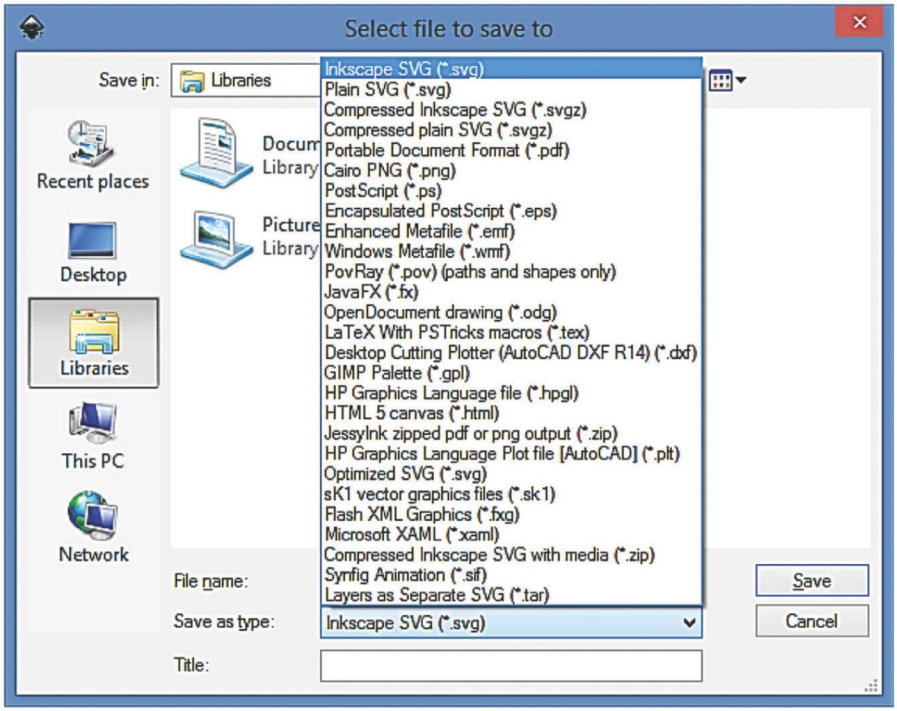
Слика 1.38. Цртање различитих врта линија

Снимање цртежа

Програм **Inkscape** омогућава чување цртежа у векторском облику, али је могуће сачувати цртеже и у олику битмапе (сл. 1.39. и сл. 1.40).



Слика 1.39. Снимање цртежа



Слика 1.40. Могући формати цртежа

Запамти

Inkscape је бесплатан програм заснован на векторској графици.
У радном прозору **Inkscape**-а, могуће је цртати по површини која се налази ван површине "папира" али те делове цртежа није могуће одштампати.
Курсор може да има различите облике; уколико селектујемо неки објекат имаће облик стрелице, али ако користимо неки од алата курсор ће имати облик примењеног алата.
Палете са алатима могу бити распоређене на различитим местима на радној површини.
Програм **Inkscape** омогућава чување цртежа у векторском облику.

Вежбе и питања за проверу стеченог знања

- ✓ Преузми са интеренета програм **Inkscape** и инсталирај на свом рачунару.
- ✓ Подеси параметре документа, тако да величина папира буде A4, оријентација портрет, а јединица мере милиметар.
- ✓ Нацртај квадрат зелене боје, димензија 6x6 cm, при чему линија треба да буде испрекидана, црвене боје, дебљине 3mm. Сачувај цртеж.
- ✓ Нацртај правоугаоник са заобљеним угловима. Сачувај цртеж.
- ✓ Нацртај круг полупречника 5 cm. Сачувај цртеж.
- ✓ Напиши своје име користећи алат за калиграфско писање. Сачувај цртеж.

Користан линк

https://inkscape.org/en/learn/tutorials

1.8. Обрада звука

Кључне речи:

звук, Power-Sound--Editor

Звук је саставни део живота људи и јавља се као пратилац многих животних активности. Најједноставније звук се може описати као промена притиска ваздуха. Човек често има потребу да сними неке звукове, као и да их даље обрађује на рачунару.

За записивање и чување аудио-датотека користе се различити аудио-формати. Они могу бити компримовани и некомпримовани. Код компримованих датотека подаци су посебним поступцима „збијени" тако да заузимају мање меморијског простора. Компримовани аудио-формати деле се на компримоване формате без губитака и компримоване формате са губицима. Формати са губицима су они код којих долази до смањења квалитета звука. Некомпримовани формати заузимају највише меморијског простора, потом компримовани формати без губитака, а најмање простора заузимају компримовани формати са губицима. Мада су данас хард-дискови великих капацитета, због чега о меморијском простору више не мора да се води рачуна као некад, ипак се у аматерске сврхе најчешће користи формат MP3. Иако овај формат уноси губитке, они се могу чути само кад се репродукција звука обавља на професионалној опреми, што је ретко.

Стога се може рећи да постоје три основне групе аудио-формата:

1. некомпримовани аудио-формати WAV, AU i AIFF
2. компримовани аудио-формати без губитака WMA; WV, APE и
3. компримовани аудио-формат са губицима MP3.

На интернету се може наћи много програма за обраду звука који су бесплатни, а поседују много могућности за обраду звука.

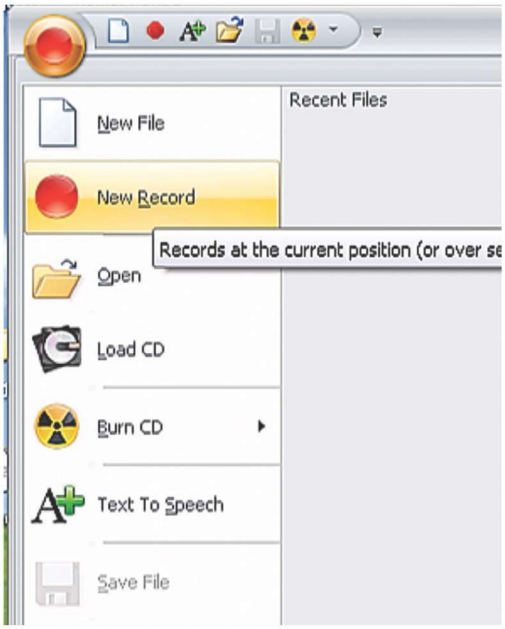
Један од таквих програма је и **Power Sound Editor** који можеш бесплатно да преузмеш са интеренет странице:

http://www.free-sound-editor.com/download.html

Програм има веома једноставно радно окружење за обраду аудио фајлова, са задовољавајућим квалитетом снимања и репродукције звука. Када се програм преузме и инсталира, приликом његовог покретања, појавиће се почетни прозор приказан на слици 1.41.

Слика 1.41. Почетни прозор програма Power Sound Editor

Најважније функције програма груписане су тако да је могуће снимање новог аудио записа помоћу опције **New Recording** (сл. 1.42). Опција **Load from** омогућава да се већ постојећи аудио фајлови унесу у програм са неке спољне меморије, а опција **Text to Speach** откуцан текст претвара у говор.



Слика 1.42. Снимање новог аудио-записа

Запамти

Некомпримовани аудио-формати су WAV, AU и AIFF.
Компримовани аудио-формати без губитака су WMA, WV и APE.
Компримовани аудио-формат са губицима су MP3.
Power Sound Editor један је од програма за обраду звука који се може бесплатно преузети са интернета

Вежбе и питања за проверу стеченог знања

- ✓ Преузми са интернета програм за обраду звука **Power Sound Editor** и инсталирај га на свом рачунару.
- ✓ Отпевај рефрен омиљене песме и сними га.
- ✓ Откуцај три реченице и задај рачунару наредбу да их прочита.

Користан линк

https://free-sound-editor.com

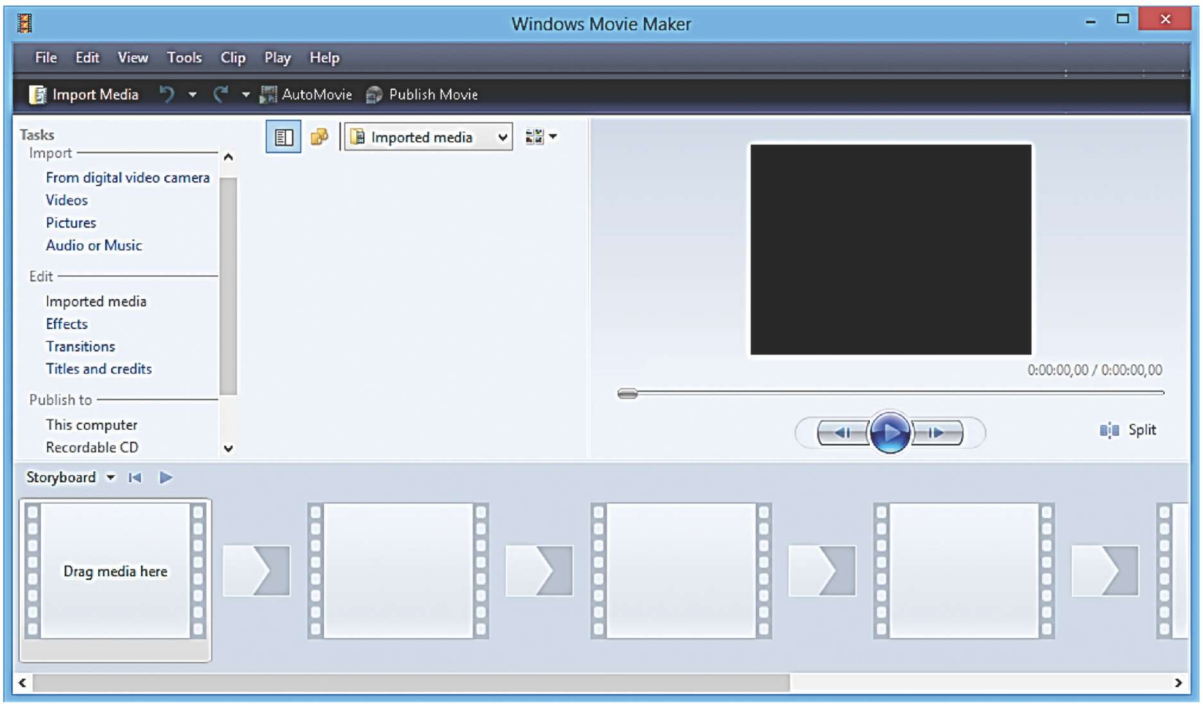
1.9. Обрада видео записа

Кључне речи:
видео запис,
Windows movie maker

Развој технике и технологије омогућио је и људима који се не баве професионално филмом да могу врло једноставно и лако да сниме видео запис, као и да га даље обрађују.

Најчешће се видео материјал снима дигиталним фотоапартом, дигиталном камером или мобилним телефоном. Овако направљени видео записи могу се пребацити на рачунар и даље дорађивати у неком програму.

Windows Movie Maker је један од најчешће коришћених програма за обраду видео записа. Програм се може преузети са инернета, а сама инсталација на рачунару траје врло кратко и сасвим је једноставна.

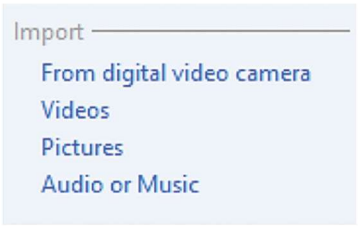


Слика 1.43. Радно окружење програма Windows Movie Maker

Покретањем програма отвара се прозор који поред насловне линије, линије менија и основних управљачких тастера на врху екрана са леве стране има мени са три групе функција:

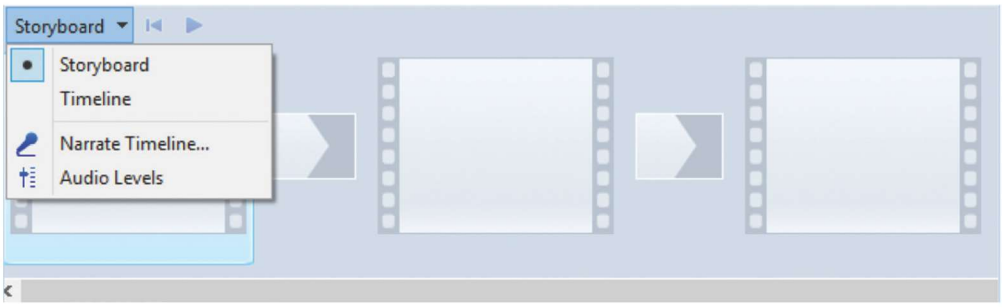
1. за увоз материјала (**Import**),
2. за монтажу (**Edit**) и
3. за снимање и објављивање коначног резултата (**Publish to**).

У средини екрана се налази простор за колекцију мултимедијалних материјала, док се десно налази мултимедијални монитор за преглед свих материјала, са класичним командама за приказ, а у доњем делу екрана трака за обраду и монтажу новог видео материјала.



Сл. 1.44. Увоз мултимедијалног материјала

Прва група функција служи за прикупљање мултимедијских компоненти директно са уређаја или са рачунара. За увоз материјала из фајлова се користи стандардни дијалог, у коме се на уобичајени начин може изабрати један или више фајлова. Увезени материјали се појављују као нове иконе у колекцији. Превлачењем помоћу миша, материјал се премешта на траку за монтажу, која има могућност да прикаже монтирани материјал по компонентама и по временском трајању.



Слика. 1.45. Изглед траке за монтажу

Приликом монтирања материјала често се користе ефекти и прелази за повезивање појединачних видео секвенци у целину. Додавање наслова на почетку и крају, као и титлова током трајања филма могуће је помоћу опције **Titles and credits**.

Коначни видео материјал је потребно снимити у неком од дигиталних видео формата. Најчешће коришћени формати за видео записе су AVI и MPEG-4.

Запамти

Најчешће се видео материјал снима дигиталним фотоапартом, дигиталном камером или мобилним телефоном.
Windows Movie Maker је један од најчешће коришћених програма за обраду видео записа.
Најчешће коришћени формати за видео записе су AVI и MPEG-4 .

Вежбе и питања за проверу стеченог знања

- ✓ Како се најчешће сннима видео материјал?
- ✓ Наведи име програм који се често користи за обраду видео записа.
- ✓ Које су основне функције програма Windows Movie Maker?
- ✓ Чему служе прелази и ефекти у програму Windows Movie Maker?
- ✓ Како додајемо наслов и титлове у филму?

Користан линк

http://vokainformatika.weebly.com/obrada-video-zapisa/obrada-video-zapisa-u-programu-movie-maker

1.10. Мултимедијалне презентације

Кључне речи:
Microsoft PowerPoint, мени Insert

У општем смислу, **презентовање** значи представљање одређене теме. Данас се презентовање идеје, производа, градива, најчешће врши помоћу рачунара, односно **програма за креирање мултимедијалних презентација**.

Један од програма који се често користи је **Microsoft PowerPoint 2013** који је саставни део програмског пакета **Microsoft Office** (сл. 1.46).

Слика 1.46. Икона програма MS Power Point 2013

Презентације направљене у том програму састоје се од слајдова, као основног елемента. Презентација може да садржи текст, табеле, графиконе, слике, звук, видео-записе и тада представља мултимедијалну презентацију. Такве презентације представљају моћну подршку сваком презентеру, а публика запамти више битних информација, што је и крајњи циљ презентовања.

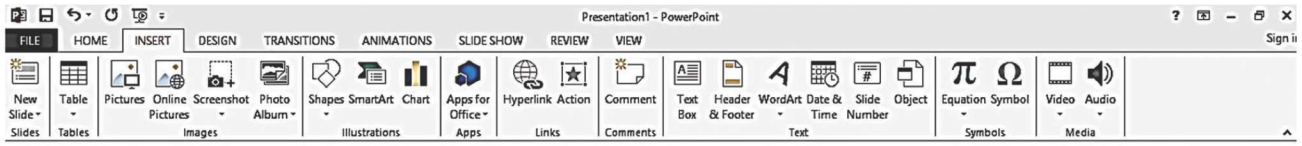
Иако се сматра да је добра она презентација које је оригинална, јединствена, необична и упечатљива, ипак се треба придржавати одређених правила приликом израде:

- први слајд треба да садржи наслов презентације, евентуално поднаслов, име аутора, датум и место презентовања;
- на другом треба представити садржај презентације, док остали слајдови прате наслове садржаја;
- на слајдовима се текст уноси у виду кратких и јасних теза (сматра се да је четири до шест информација довољно);
- треба користити стандардне, лако читљиве фонтове (нпр. Times New Roman, Arial) величине од 24pt do 32 pt;
- потребно је одабрати контрастне боје за позадину и текст и
- пажљиво изабрати анимације за текст и објекте (оне могу да буду врло ефектне, али и да скрену пажњу са важних и битних информација које желимо да пренесемо публици).

Покретањем програма отвара се прозор у ком започињемо израду презентације. Након бирања дизајна и уређивања насловног слајда, може се додати нови слајд и извршити унос садржаја.

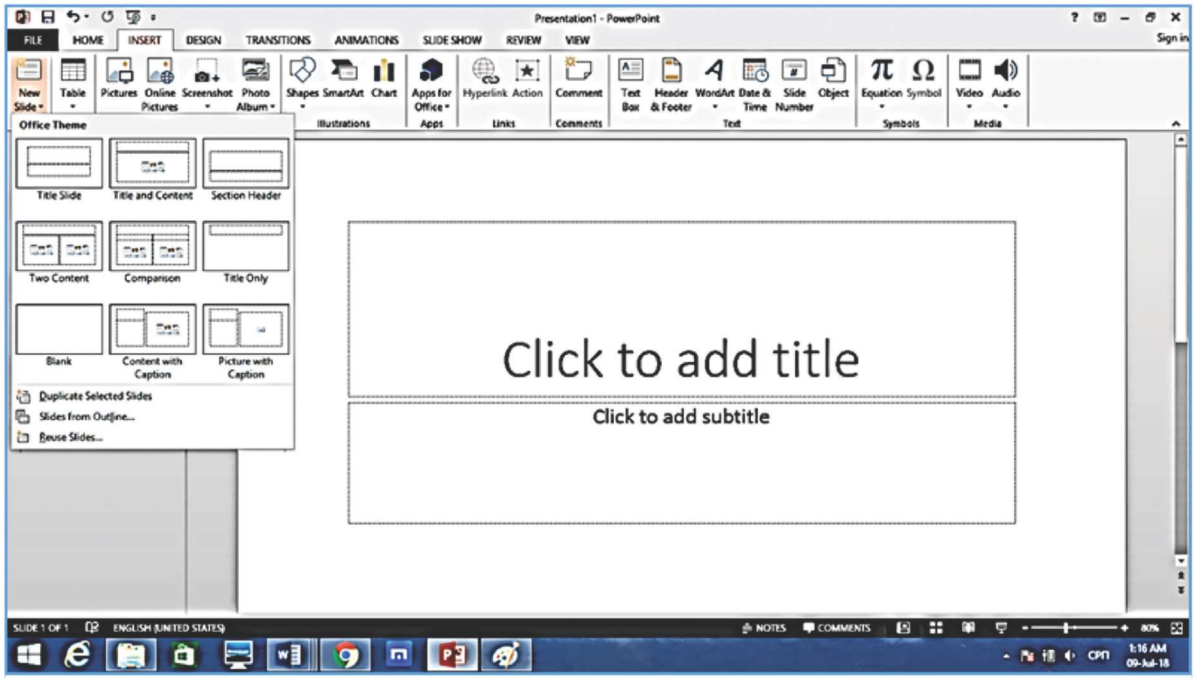
Унос садржаја презентације

Најлакши начин за унос различитог садржаја у презентацију јесте коришћење картице **Insert** (сл. 1.47).



Слика 1.47. Картица Insert

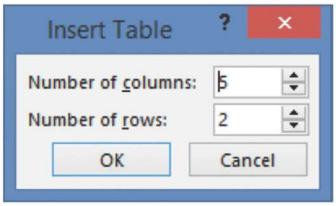
Insert/New slide – ова картица користи се за убацивање новог слајда са жељеним распоредом садржаја. Текст се уноси и обликује на исти начин као и у програму за обраду текста (сл. 1.48).



Слика 1.48. Почетак израде презентације

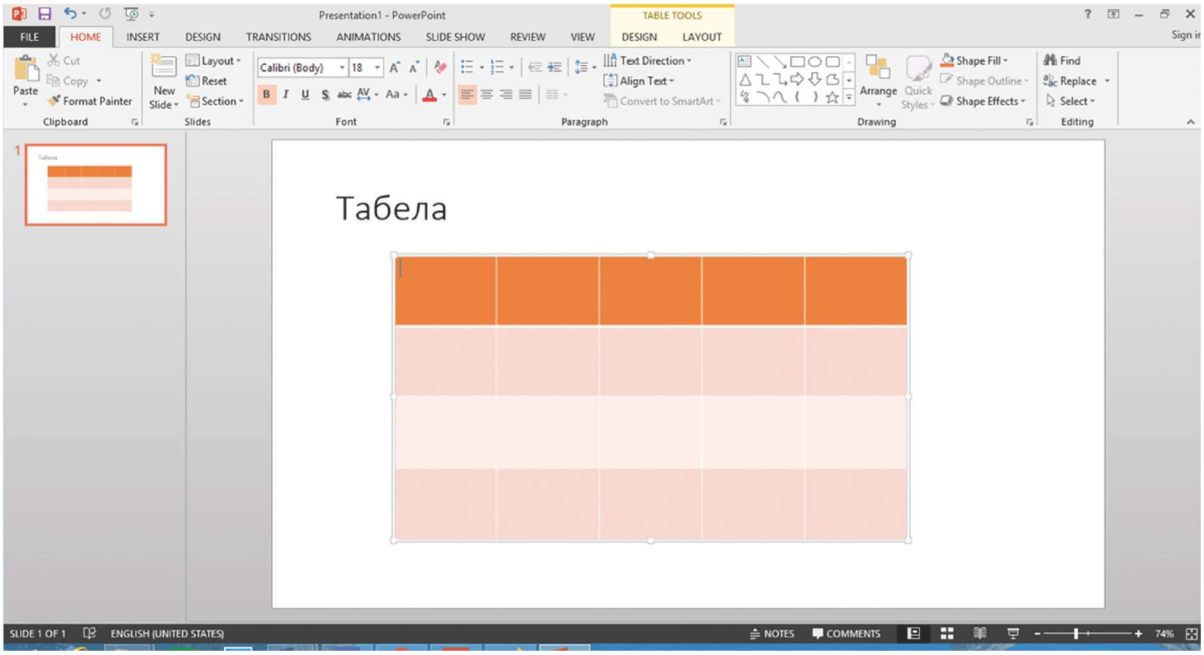
Видео и аудио записе можемо уметнути у презентацију на врло једноставан начин. Видимо да се опције за унос видео и аудио записа налазе на крају реда картице **Insert.** Када на њих кликнемо левим тастером миша отвара се прозор за избор и потврду уноса видео или аудио записа у презентацију.

Insert/ Table/ Insert table отвара оквир за дијалог за унос броја колона и редова табеле (сл. 1.49).



Слика 1.49. Убацивање табеле у презентацију

Insert table/ Draw table – омогућава цртање табела унутар презентација, док коришћењем опција из картице Table Tools можемо даље уређивати табеле (сл. 1.50).



Слика 1.50. Унос табеле у презентацију

Слике, различите облике, украсна слова, аудио и видео материјале можемо додати презентецији користећи опције из картице **Insert**. Садржај који се уноси у презентацију може бити складиштен на рачунару, а такође могуће га је преузети и са интернета.

Запамти

Power Point је комерцијални програм за израду презентација.
Постоје правила којих се треба придржавати приликом израде презентација.
Најједноставнији начин за убацивање мултимедијалног садржаја у презентацију је преко менија **Insert**.

Вежбе и питања за проверу стеченог знања

✓ Направи мултимедијалну презентацију којом представљаш своју школу. Аудио и видео материјале можеш да преузмеш са интернета или да их направиш самостално. Податке о броју ђака, расподели наставника и одељења представи табеларно.

1.11. Рачунарство у облаку

Кључне речи:
Cloud computing, online, Google Drive

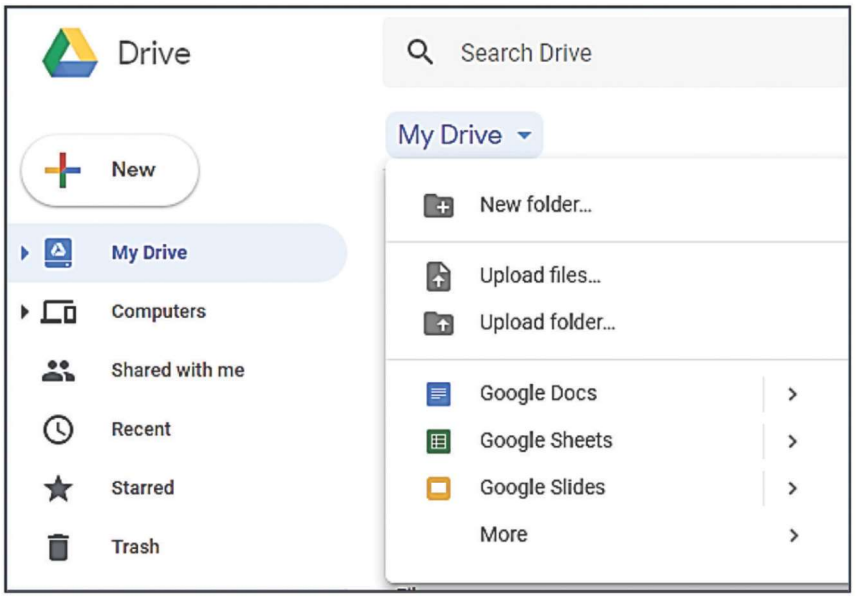
Рачунарство у облаку (**Cloud computing**) у основи представља дељење и чување ресурса (програма, слика, докумената, итд.) путем интернета.

Ресурси који се користе, преузимају и деле у облаку могу да буду постављени на рачунарима који се налазе на удаљеним локацијама. Корисници могу да им приступе помоћу апликација са својих мобилних телефона или десктоп апликација.

Cloud сервиси, тј. сервиси базирани на облаку су имејл сервиси, као и апликације за складиштење података, онлајн игрице, итд.

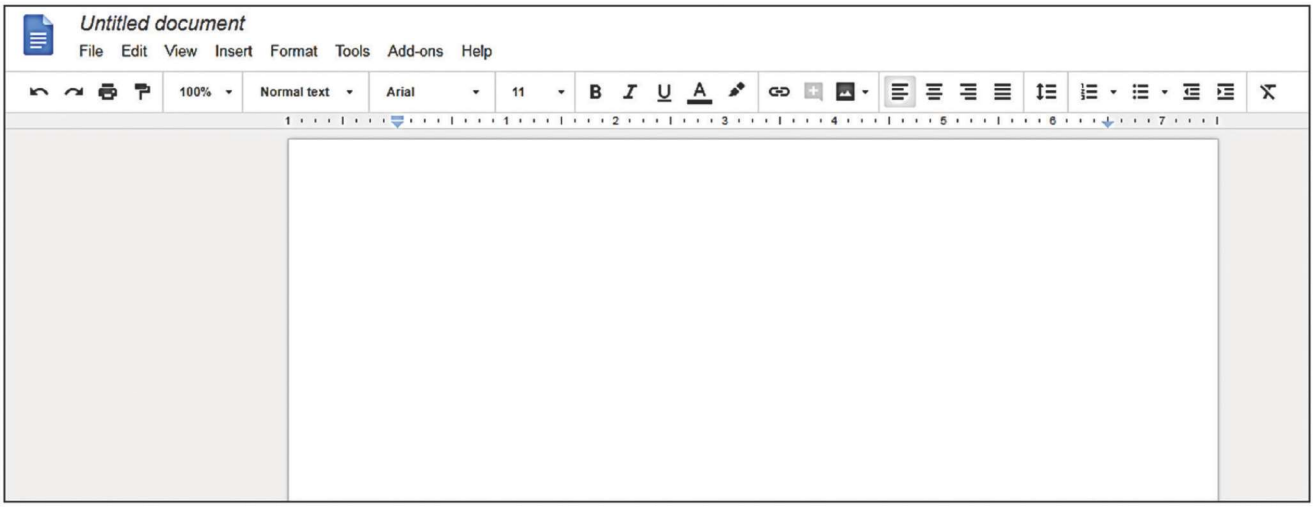
Google Диск

Најраспрострањенији сервис за електронску пошту **Gmail** пружа могућност да преко имејл налога користиш и **Google Диск (Google Drive)** за дељење и чување својих ресурса. Када приступиш **Google Диску**, на картици **Мој диск (My Drive)** појављују се следеће опције: **Креирање новог фолдера (New folder)**, **Отпремање фајла (Upload files)**, **отпремање фолдера (Upload folder)** што је и приказано на слици 1.51.



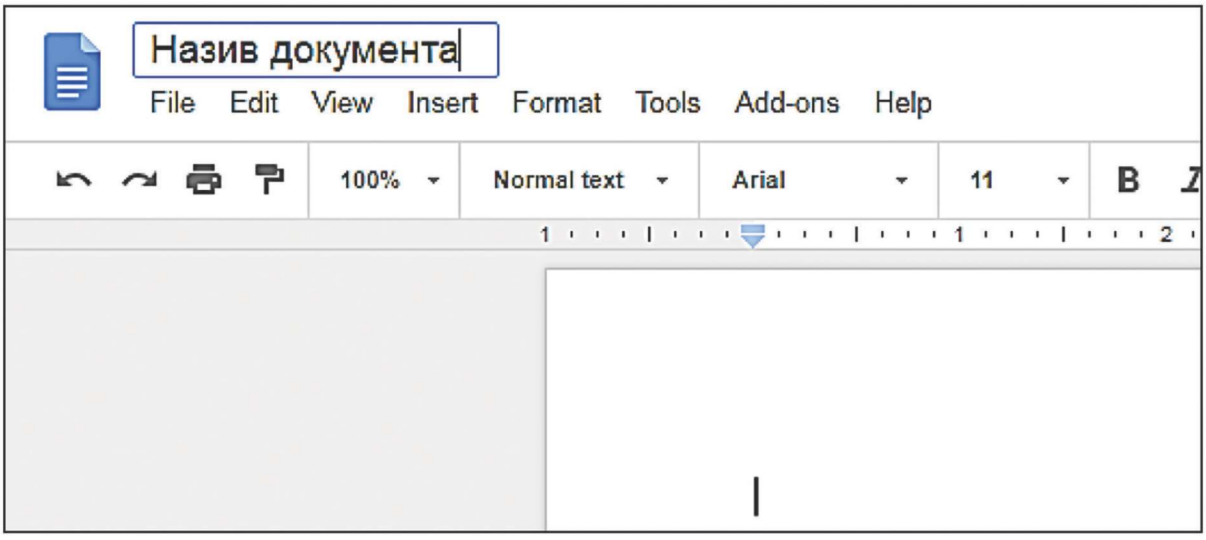
Слика 1.51. Приказ картице Мој диск

Креираћемо нов **Google документ (Google Docs)**. Када кликнемо левим тастером миша на опцију Google документи отвара нам се прозор приказан на сл. 1.52.



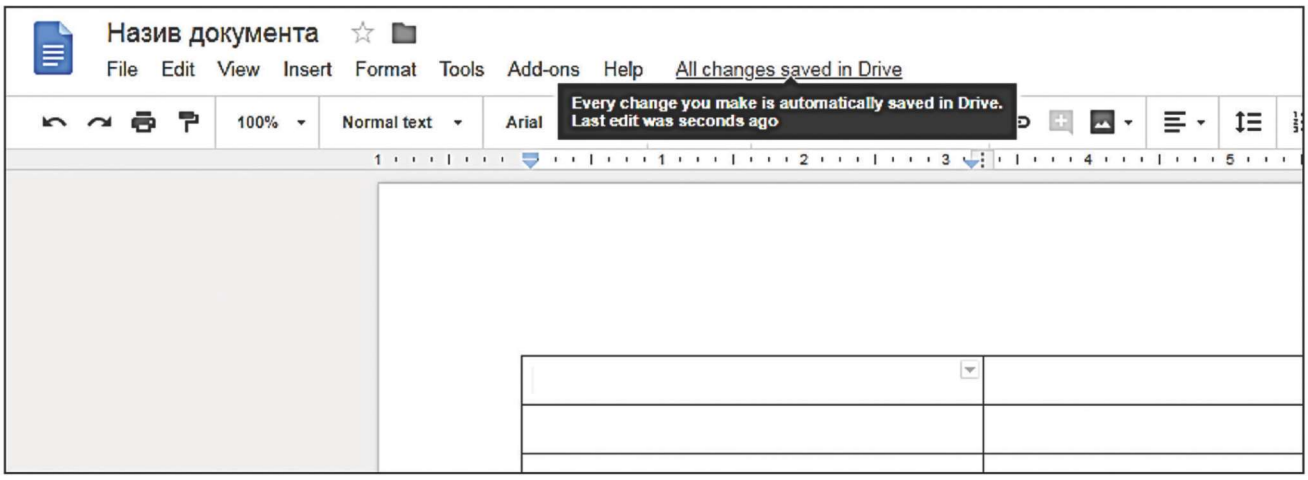
Слика 1.52. Изглед Google документа

Назив документа уносимо у горњи леви угао документа (сл. 1.53).



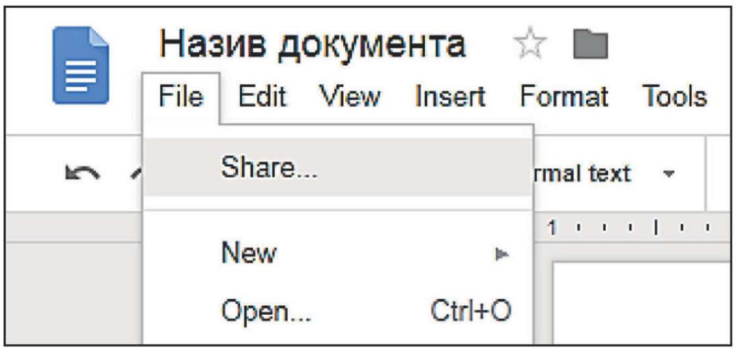
Слика 1.53. Унос назива документа

Потом настављамо са радом у документу, уносећи текст, табеле, слике и остале податке за које желимо да се нађу у нашем документу. Свака промена коју направимо у документу биће аутомтски сачувана на нашем **Google Drive** налогу, (сл. 1.54).



Слика 1.54. Приказ аутоматског чувања документа

Свој документ можеш да поделиш са другима, тако што ћеш отворити картицу **File**, и изабрати прву понуђену опцију **Подели (Share)**, сл. 1.55. Потом ће се отворити прозор **Подели са другима (Share with others)**, приказан на сл.1.56. у ком ћеш укуцати имејл адресу особе с којом желиш да поделиш свој документ. Након тога треба да кликнеш на **Урађено (Done)**. Документ је могуће поделити и прослеђивањем линка. Подељени документ друга особа може да уређује уколико за то добије ваше одобрење, да коментарише или само да га види.

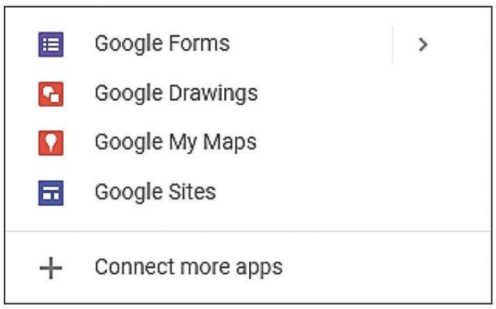


Слика 1.55. Приказ опције Подели (Share)

Слика 1.56. Приказ прозора Подели са другима (Share with others)

Помоћу **Google диска** можеш и да прикупљаш податке који те интересују. Уколико желиш да прикупиш податке о својим другарима, нпр. која је њихова омиљена храна, музика, филм, предмет итд. то ћеш урадити помоћу онлајн упитника који ћеш поделити са њима, а потом на свом **Google диску** сачувати и анализирати податке.

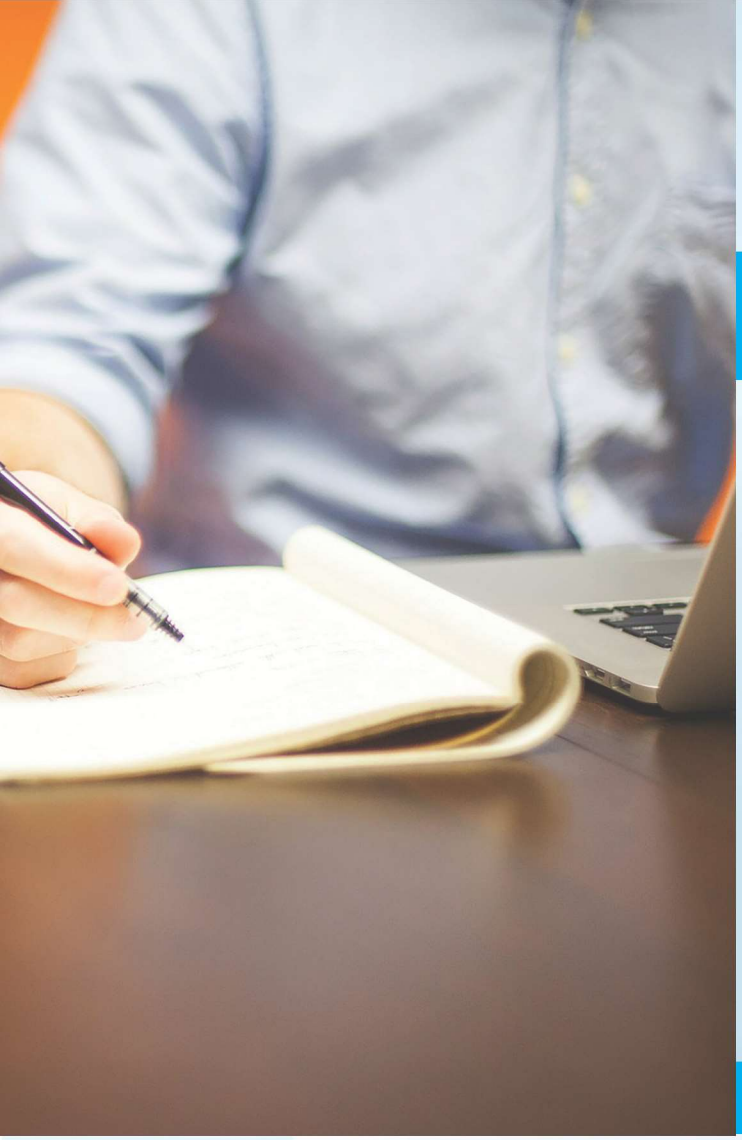
Након отварања картице **Мој диск (My Drive)**, појављује се опција **Још (More)**. Када кликнеш на њу левим тастером миша, отварају се опције приказане на слици 1.57. Избором опције **Google форме (Google Forms)** можеш да отпочнеш са израдом свог онлајн упитника, који ћеш поделити са другима.



Слика 1.57. Приказ опција Још (More)

Питања за проверу знања

1. Шта је рачунарски систем?
2. Шта може да се подешава у картици **Personalization**?
3. Наброј екстензије за текстуалне датотеке?
4. Шта карактерише сваку табелу?
5. Која је разлика између векторске и растерске графике?
6. Који су основни елементи графике?
7. У програму **Inkscape** нацртај цртеж по твојој жељи.
8. Шта је звук?
9. У програму **Power Sound Editor** сними аудио-фајл по твојој жељи.
10. Шта је слајд?
11. Шта омогућава опција **Insert table/ Draw table**?
12. Шта представља рачунарство у облаку?
13. За шта користимо **Google Диск**?
14. Објасни поступак креирања **Google документа**.
15. Објасни опцију **Подели (Share)** коју користимо на **Google Диску**.



2

ДИГИТАЛНА ПИСМЕНОСТ

2.1. Дигитална писменост и рачунарске мреже

Кључне речи:
дигитална писменост, рачунарске мреже

Попут читања и писања, дигитална писменост представља подразумевану вештину у 21. веку. Означава способност стварања, проналажења и преноса информација у дигиталном облику као и способност процене тачности информације. Дигитална писменост обухвата и развој практичних и функционалних вештина које су у вези са употребом дигиталних уређаја.

Дигитално писмена особа има знања и вештине неопходне за коришћење дигиталних уређаја, као што су рачунари, паметни телефони и сл. Особа која је дигитално писмена свесна је бројних извора информација, предности и мана разних извора информација, ако и вредности информација. Дигитално писмена особа може да разуме различите изворе информација и међусобно их повеже.

Она поседује знања и вештине потребне за коришћење рачунарских мрежа и има развијену способност за укључивање у онлајн заједнице.

Рачунарске мреже

Скуп међусобно повезних или умрежених рачунара ради ефикаснијег и бржег коришћења информација, података и ресурса назива се **рачунарска мрежа** (сл. 2.1).

Слика 2.1. Умрежени рачунари

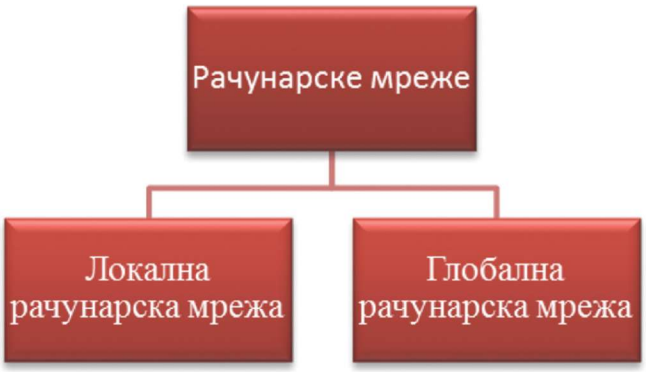
Рачунарске мреже могу се поделити на разне начине, у зависности од тога да ли се посматра: површина коју покрива мрежа, начин повезивања рачунара у мрежи (топологија), начин комуникације рачунара у мрежи (логичка организација) итд.

У зависности од удаљености уређаја с којим се остварује веза преко рачунара, разликују се две врсте мрежа (сл. 2.2),:

– локалне мреже – LAN,
– глобалне мреже – WAN.

Локална рачунарска мрежа обухвата повезане уређаје који се најчешће налазе у једној просторији, згради или у више зграда али у власништву једне компаније. Уређаји могу бити повезани кабловима (**Ethernet**) или бежично (**Wireless**).

Глобална рачунарска мрежа обухвата повезане уређаје на већим удаљеностима. Интернет је најпознатија глобална мрежа.



Слика 2.2. Подела рачунарских мрежа

За повезивање рачунара у мрежу користи се додатна опрема коју чине проводници (телефонски, коаксијални, оптички и и UTP каблови са конекторима), мрежне картице и остали мрежни уређаји.

IP адреса је структура података која служи за јединствено идентификовање рачунара у мрежи.

Ова вредност је јединствени идентификатор тако да два система не могу имати исту IP адресу. Системи који имају више излаза према мрежи, тј. више мрежних картица, могу имати више IP адреса.

IP адресе се записују у форми 4 броја раздвијена тачкама, нпр. 192.168.1.30

Протоколи су скуп правила која се примењују за комуникацију рачунара у мрежи.

Запамти

Дигитална писменост означава способност стварања, проналажења и преноса информација у дигиталном облику као и способност процене тачности информације.

Дигитално писмена особа има знања и вештине неопходне за коришћење дигиталних уређаја, као што су рачунари, паметни телефони и сл .

Скуп међусобно повезних или умрежених рачунара ради ефикаснијег и бржег коришћења информација, података и ресурса назива се **рачунарска мрежа**.

Рачунарске мреже могу се поделити на разне начине, у зависности од тога да ли се посматра: површина коју покрива мрежа, начин повезивања рачунара у мережи (топологија), начин комуникације рачунара у мрежи (логичка организација) итд .

IP адреса је структура података која служи за јединствено идентификовање рачунара у мрежи.

Протоколи су скуп правила која се примењују за комуникацију рачунара у мрежи.

Вежбе и питања за проверу стеченог знања

- ✓ Шта је то дигитална писменост?
- ✓ Које вештине поседује дигитално писмена особа?
- ✓ Шта је рачунарска мрежа?
- ✓ Како можемо поделити рачунарске мреже?
- ✓ Шта је то IP адреса?
- ✓ Шта је протокол?

Користан линк

http://vokainformatika.weebly.com/internet1/

2.2. Интернет

Кључне речи:
Интернет, www, e-mail

Реч **интернет** изведена је од две речи – International network, што значи међународна мрежа. Интернет представља информациону међународну мрежу са неверованом количином података из свих области и елемената савременог живота. Сваки корисник има могућност да користи све информације из система, као и да у систем интернета сачува своје информације. Све је више корисника и базе се шире енормном брзином.

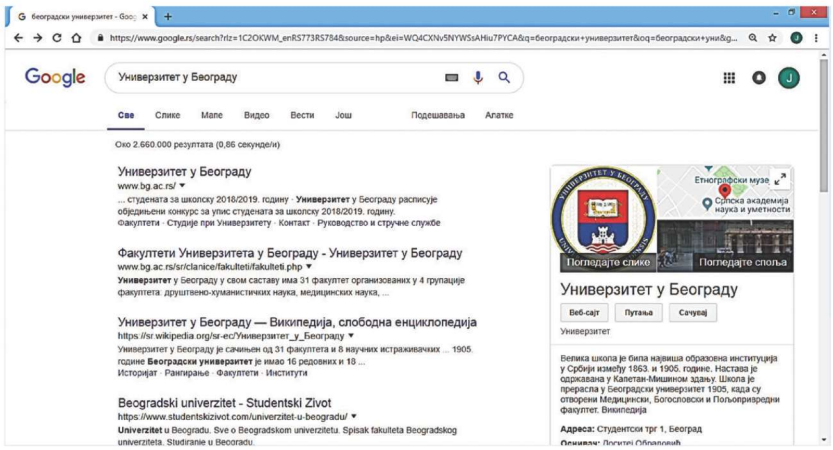
Интернет је највећа глобална мрежа рачунара.

Данас је могуће студирати преко интернета, одржавати скупове и симпозијуме чији учесници седе у својим собама било где на Земаљској кугли, али је тонска и визуелна комуникација преко интернета таква као да су у конференцијској дворани.

Информација је постала најтраженија роба, тако да већина корисника до њих долази помоћу интернета (сл. 2.3).

Главне услуге, тј. сервиси које пружа интернет јесу:
– светска мрежа (**WWW – Word Wide Web**);
– електронска пошта (**E–mail**);
– дискусионе групе (**Usenet Newsgroups**);
– протокол за пренос фајлова (**FTP – File Transfer Protocol**);
– успостављање везе са удаљеним рачунаром (**Telnet**);
– конверзација у реалном времену (**IRC – Internet Relay Chat**).

Најчешће коришћени сервиси на интернету јесу светска мрежа World Wide Web (WWW) и E-mail односно електронска пошта. WWW представља велики број докумената до којих се долази помоћу интернет претраживача, док је електронска пошта најчешћи и најстарији начин коришћења интернета и служи за слање и примање порука између корисника.



Слика 2.3. Претраживање интернета

Запамти

Реч **интернет** изведена је од две речи – International network, што значи међународна мрежа.

Интернет представља информациону међународну мрежу са невероватном количином података из свих области живота.

Најчешће коришћени сервиси на интернету јесу светска мрежа **World Wide Web (WWW)** и **E-mail** односно електронска пошта.

Вежбе и питања за проверу стеченог знања

- ✓ Од којих речи је настала реч интернет?
- ✓ Каква је мрежа интернет?
- ✓ Који су најпознатији интернет сервиси?
- ✓ Чему служи електронска пошта?
- ✓ Пронађи на интернету информације од значаја за место у којем станујеш?

Користан линк

http://vokainformatika.weebly.com/internet1/

2.3. Правила безбедног понашања на интернету

Кључне речи:

интернет, злонамерни програми, антивирусни програми, правила безбедног понашања на интернету

Интернет је моћна база знања, из које човек данас добија много корисних иформација.

Нажалост, у раду са ИКТ уређајима можете се срести и са различитим **злонамерним програмима**.

Ти програми могу да наруше правилан рад уређаја, могу без дозволе да преузмају битне податке, обуставе рад неких програма итд.

Најпознатији злонамерни програми су: **вируси, црви, шпијунски програми, разни рекламни програми** итд.

Да би се спречиле непријатности у раду пожељно је на уређају инсталирати програм који штити од злонамерих програма, тј. неки **антивирусни програм**.

Најпознатији антивирусни програми су: **Avast, AVG, Kaspersky, Eset Nod32 Antivirus, Norton, Microsoft Security Essentials (MSE)** (сл. 2.4).

Слика 2.4. Логотипи најпознатијих антивирусних програма

Поред злонамерних програма на интернету има и злонамерних људи, па је савет да се придржаваш основних правила. Све што није дозвољено у свакодневном животу, није дозвољено ни на интернету.

Центар за безбедни интернет Србије формулисао је десет правила које би требало да усвојиш.

– Немој да дајеш своје личне податке као што су име, презиме, адреса становања, ЈМБГ и број телефона особама које лично не познајеш.
– Одмах се обрати родитељима уколико наиђеш на било шта што те узнемирава или плаши на интернету или мобилном телефону.
– Никада немој да пристанеш на састанак са особом коју си упознао/ла на интернету, а да о томе претходно о томе не обевестиш родитеље. Уколико се родитељи сложе с тим, такав састанак мора да буде на јавном месту и уз присуство твојих родитеља.

– Никада немој да шаљеш своју слику или било шта о себи без договора с родитељима.
– Не одговарај ни на какве поруке које су злонамерне или које те на било који начин узнемиравају. Ниси крив/а што добијаш такве поруке. Ако ти неко пошаље такву поруку, одмах то реци родитељима.
– Разговарај са родитељима и договори се с њима о правилима понашања на интернету и правилима за коришћење мобилног телфона. Заједно одредите време у току дана кад можеш да користиш интернет, колико дуго током једног дана можеш да будеш на нету и сајтове које можеш и треба да посећујеш. Немој да посећујеш друге сајтове без договора са родитељима и поштуј договорена правила.
– Лозинке које имаш немој да откриваш никоме (чак ни најбољим друговима), само својим родитељима.
– Пре него што учиташ нешто са интернета или инсталираш неки софтвер, консултуј се с родитељима о теме јер би могао/ла да угрозиш рад и безбедност свог рачунара или мобилног уређаја или пак да угрозиш приватност и личне податке чланова своје породице.
– Буди примеран корисник интернета и мобилног телефона и не ради ништа чиме би могао да повредиш или угрозиш друге или прекршиш закон.
– Помози својим родитељима да науче како да се забављају и уче на интернету и дели са њима информације о интернету, рачунарима и новим технологијама.

Запамти

Интернет је највећа глобална мрежа рачунара.
Злонамерни програми су: вируси, црви, шпијунски програми, разни рекламни програми итд.
Програми који штите од злонамерних програма називају се антивирусни програми.
Поштуј правила безбедног понашања на интернету.

Користан линк

https://www.avast.com
http://www.pametnoibezbedno.gov.rs/rs-lat
https://kliknibezbedno.wordpress.com/

2.4. Правила лепог понашања на интернету

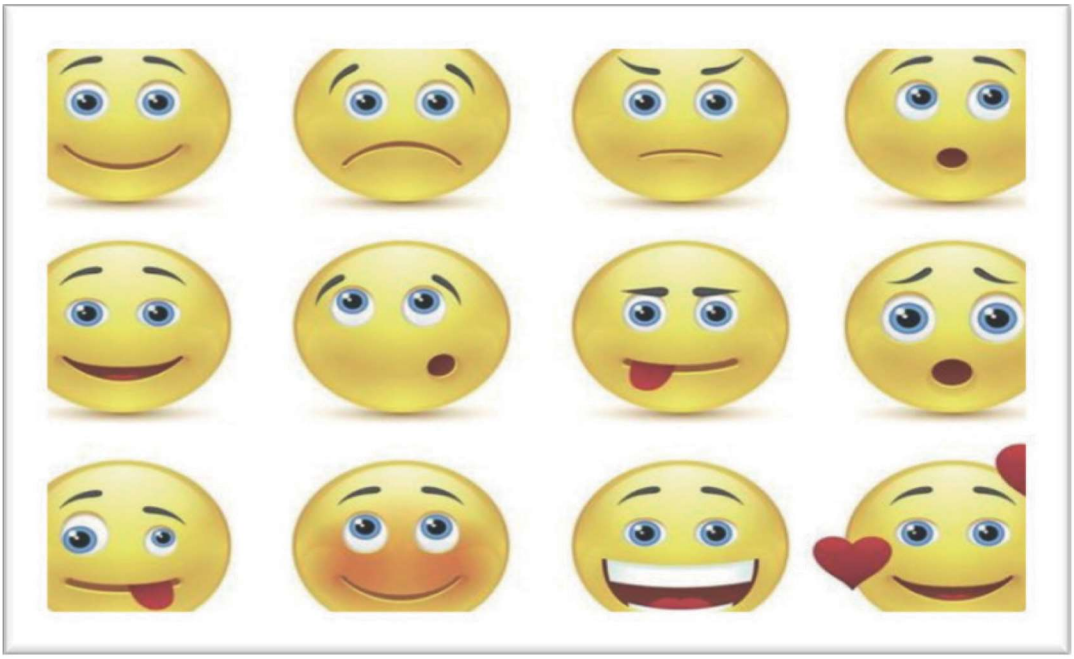
Кључне речи:

нетикеција, интернет бонтон, емотикони

Као што постоје правила лепог понашања у животу, тако постоје и правила лепог понашања на интернету. **Нетикеција** је израз који се користи за **интернет бонтон**.

Навешћемо правила интернет бонтона:

– Забрањено је псовати или вређати особе с којима комуницираш преко чета или Фејсбука.
– Забрањено је представљати се као неко други.
– Слање ланаца среће и порука које обећавају лаку и брзу зараду није пристојно јер се у реалном животу новац не може зарадити на тај начин.
– Поштуј власничка и ауторска права.
– Поруке пиши у пријатељском тону.
– Никад немој било кога да вређаш због националне, расне или верске прдипадности.
– Можеш користити „смајлиће" ради лакшег споразумевања. Смајлићи се још називају **емотикони** (сл. 2.5).
– Не шаљи велике количине података људима који их од тебе нису тражили.
– Писање искључиво великим словима у електронској комуникацији са другим људима сматра се непристојним јер подсећа на викање.



Слика 2.5. Емотикони који се највише користе

Запамти

Нетикеција је израз који се користи за интернет бонтон.

Емотикони су сличице „смајлића“ које можеш користити у комуникацији на интернету.

Вежбе и питања за проверу стеченог знања

✓ Направи постер са десетак најважнијих правила у вези лепог понашања на интернету.

Користан линк

http://internetbonton.blogspot.com/

2.5. Претраживање интернета – одабир и преузимање садржаја

Кључне речи:
читачи, веб-адреса, кључна реч

Претраживање садржаја на интернету често зна да се претвори у мукотрпан посао. У мору информација које су свима доступне важно је знати како доћи до оне која је тачна и права.

Претраживање интернета најчешће се обавља на два начина.

Први начин је да се у неком од програма који се зову **читачи (Browser)** (сл. 2.6) – најпознатији међу њима јесу **Google Chrome**, **Mozilla Firefox**, **Opera**, **Netscape**, **Safari** – унесе тачна адреса веб-сајта у поље за адресирање. Свака веб-адреса изгледа овако: www.ime_web_adrese.domen

Слика 2.6. Најчешће коришћени програми за претраживање интернета

Други начин претраживања интернета користи се када не знамо тачну веб-адресу. У том случају, за проналажење информација користе се „кључне речи" и програми за претраживање од којих је најпознатији **Google**.

Кључна реч је она реч која прецизније описује предмет интернет претраге; знак „+" користи се за везивање кључих речи, а знак „-"за искључивање кључних речи. Сурфовање („једрење" по информацијама) је израз који се често користи као синоном за реч претраживање.

Приступ информацијама више није проблем, али одабир веродостојих у мору нетачних и полутачних јесте. У том смислу врло је важно развити критички став приликом процене вредности одређеног интернет садржаја.

Може се сматрати да је информација тачна ако је актуелна, ако је проверљива, ако су наведени извори одакле је преузета или уколико је аутор особа од интегритета којој се може веровати. Често се на интернету налазе застареле информације које више ничему не служе. Ознаке које нас упућују на информације приказане су латиничним словом и (сл. 2.7).

Слика 2.7. Могуће ознаке за информације

Запамти

Најпознатији **читачи** су **Google Chrome**, **Mozilla Firefox**, **Opera**, **Netscape**, **Safari**.
Претраживање интернета врши се на два начина – помоћу тачне веб-адресе или помоћу кључних речи.
Тачна информација је актуелна ако је проверљива и ако су наведени извори одакле је преузета.

Вежбе и питања за проверу стеченог знања

- ✓ Пронађи на интернету информације о оперативном систему **Windows**. У свесци запиши своја запажања у вези са претрагом.

2.6 Заштита ауторских права

Кључне речи:

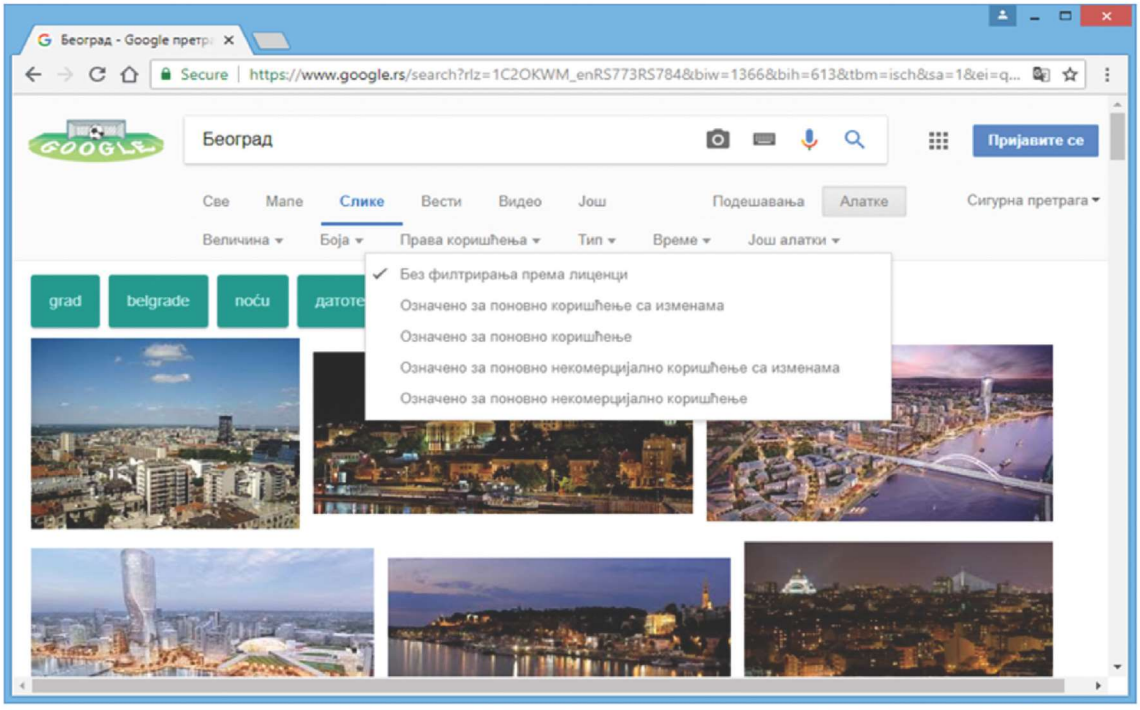
аутор, ауторска права, копирајт, плагијат

Закон о ауторским и сродним правима који је тренутно на снази у Републици Србији дефинише ауторско дело на следећи начин: „Ауторско дело је оригинална духовна творевина аутора, изражена у одређеној форми, без обзира на његову уметничку, научну или другу вредност, његову намену, величину, садржину и начин испољавања, као и допуштеност јавног саопштавања његове садржине.”

Највећи део садржаја на интернету заштићен је ауторским правима и често се на дну интернет странице налази симбол копирајт (сл. 2.8), који јасно показује да се садржај не може копирати и даље користити.

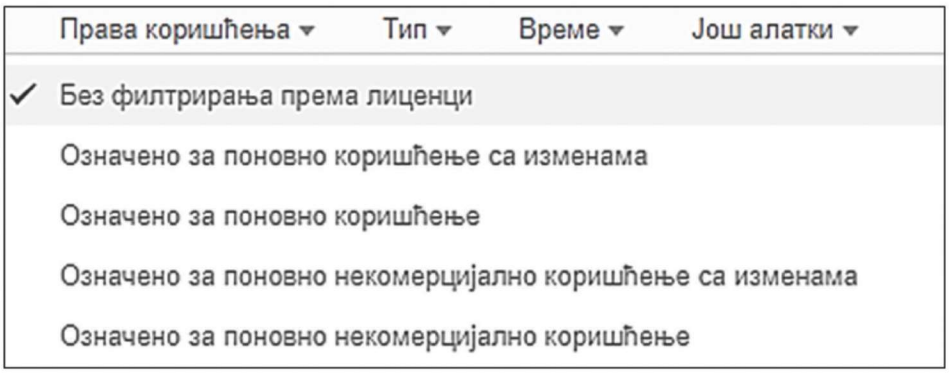
Слика 2.8. Симбол копирајт

На интернету, постоје и садржаји који се могу даље употребљавати без измена или са изменама у комерцијалне или некомерцијалне сврхе (сл. 2.9).



Слика 2.9. Претраживање интернета – поштовање ауторских права

На претходној слици приказана је претрага слика Београда, приликом које је активиран алат за филтрирање слика у односу на степен заштите ауторских права (сл. 2.10).



Слика 2.10. Различити степени заштите ауторских права

Плагијат је копирање и присвајање неког ауторског рада.

Поштовање туђег рада сматра се питањем културе и части и због тога увек треба навести извор преузетих инфомација, слика, текстова, при чему се претходно мора прибавити дозвола за њихово коришћење.

Запамти

Ауторско дело је оригинална духовна творевина аутора.
Симбол копирајт јасно показује да се садржај не може копирати и даље користити.
Плагијат је копирање и присвајање неког ауторског рада.

Вежбе и питања за проверу стеченог знања

✓ Претражи на интернету слике Београда, укључи алат за филтрирање у односу на ауторска права и запиши своја запажања у свеску.

Пројектни задатак

Након савладаних области Информационо-комуникационе технологије и Дигитална писменост, пројектни задатак можете да урадите користећи стечено и надограђено знање о интернету и програмима за обраду текста, слика, презентација итд. Неопходно је одабрати тему пројектног задатка, формирати групе, тј. тимове који ће дефинисати поступке за реализацију задатка, прикупљати информације, припремити текстуални документ, обрадити слике, припремити презентацију и извршити демонстрацију и презентацију пројектног задатка.

Пример задатка

Направите књигу до 15 страна за децу млађег школског узраста на неку од следећих тема:
– Знамените личности Србије (научници, књижевници, уметници, војсковође...);
– Мој крај (представите одређен локалитет, његове карактеристике, специфичности...);
– Дигитално насиље.

Приликом израде задатка консултујте са се наставницима других предмета, како бисте им изложили своју одабрану тему, након чега ће вас упутили у даљи истраживачки рад.

Фазе израде пројектног задатка

Израда пројектног задатка треба да има следеће фазе:
– подела на групе;
– одабир теме задатка;
– дефинисање рока за реализацију пројектног задатка;
– израда плана рада на пројекту и дефинисање поступака потребних за реализацију пројектног задатка;
– проналажење и избор материјала за реализацију пројекта;
– обрада одабраног материјала (текста, слика) ;
– представљање решења и презентације.

Питања за проверу знања

1. Шта је дигитална писменост?
2. Објасни које вештине и знања поседује дигитално писмена особа.
3. Шта је рачунарска мрежа?
4. Шта је интернет?
5. Које услуге пружа интернет?
6. Од чега антивирусни програми штите наше уређаје?
7. Објасни правила којих би требало да се придржаваш на интернету.
8. Шта је нетикеција?
9. Шта представља кључна реч приликом претраживања интернета?
10. Шта је ауторско дело?
11. Шта је плагијат?

3

РАЧУНАРСТВО

3.1. Увод у програмирање

Кључне речи:

програмирање, програм, програмски језик

Програмирањем се баве програмери. Програмери пишу програм програмским језиком којим се рачунару задају инструкције за извршење одређеног задатка.

Шта је програмирање?

Програмирање је поступак решавања неког проблема односно одређеног задатка, применом језика који рачунар разуме како би извршио одређени задатак. Ти језици се називају програмски језици, њих рачунар разуме и на основу њих ради оно што је програмер замислио. Постоји много програмских језика, а ми ћемо у нашем раду користити програмски језик **Ruby**.

Програмери су стручни људи који нам омогућавају да рачунару задајемо задатке и наредбе, тако што су направили, тј. написали рачунарски програм у неком од програмских језика.

Сваки програмер прво треба да дефинише проблем или задатак који жели да реши, потом да одреди редослед радњи које су неопходне за решевање проблема или задатака, тј. да формира алгоритам и након тога, може да приступи писању програма, његовом тестирању, као и изради упутства о његовом коришћењу за даље кориснике.

На основу формираног алгоритма са дефинисаним редоследом корака извршења задатка, програмер може да напише програм у било ком програмском језику.

Најпознатији програмски језици су: **Scratch, Stencyl, Ruby, Swift, Python, AppInventor, Alice, Java, C#, PHP**…(сл. 3.1).

Слика 3.1. Програмски језици

3.2. Радно окружење програмског језика Ruby

Кључне речи:

Ruby, радно окружење, интерактивно окружење

Творац програмског језика **Ruby** је Јукихиро Мацумото (Yukihiro Matsumoto). Тај програмски језик настао је у Јапану средином деведесетих година.

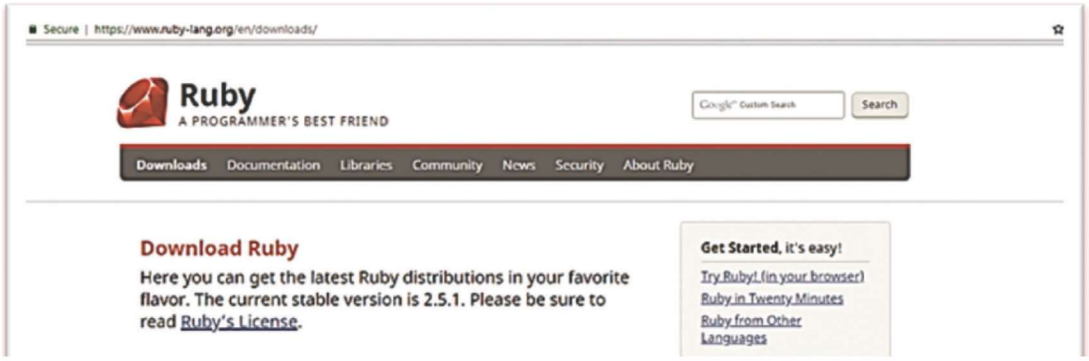
Дизајниран је са идејом да програмирање буде забавно и интересантно за програмера, и да, самим тим, утиче на његову продуктивност и маштовитост (сл. 3.2).

Ruby је намењен како почетницима, тако и искусним програмерима са циљем да на интересантан, једноставан и забаван начин развију технике писања програма у текстуалном програмском језику.

Слика 3.2. Лого програма Ruby

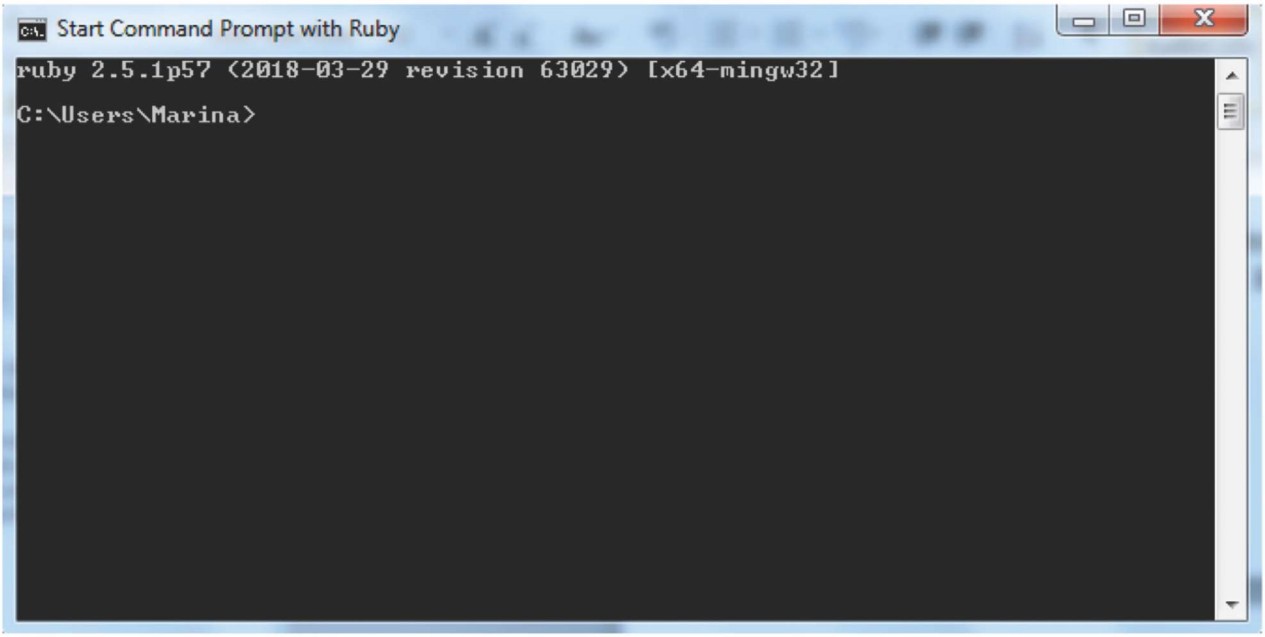
Преузимање и инсталација програмског језика Ruby

Програмски језик је бесплатан и може да се преузме са веб-сајта https://www.ruby-lang.org/en/downloads/. У зависности од постојећег оперативног система, пружа нам се могућност избора одговарајућег програма (сл. 3.3). Преузмимамо га тако што левим тастером миша кликнемо на оперативни систем који користимо. У новоотвореном прозору левим тастером миша кликнемо на дугме Download. Отвориће нам се нова страна на којој ће нам бити понуђено више верзија инсталационог фајла програма. У складу са оперативним системом изабраћемо инсталациони фајл и када на њега кликнемо левим тастером миша отвориће нам се прозор у ком дефинишемо место на рачунару, на коме ћемо га сачувати. Након што изаберемо место на коме ћемо сачувати инсталациони фајл левим тастером миша кликнућемо на дугме Save.



Слика 3.3. Преузимање софтвера

Како би смо га инсталирали, прво морамо приступити месту на рачунару на ком смо га сачували. Потом левим двокликом на инсталациони фајл покрећемо његову инсталацију. Када левим тастером миша кликнемо на дугме Run инсталација ће започети, а можемо и одустати од инсталације програма и левим тастером миша кликнути на дугме Cancel. Након инсталације програма, како би смо отпочели рад и приступили његовом радном окружењу, испратићемо следећу путању (Start Menu\All Programs\Ruby 2.5.1-1-x64 with MSYS2) и левим тастером миша кликнути на извршни фајл Start Command Prommpt with Ruby. Његово радно окружење приказано је на слици 3.4.



Слика 3.4. Радно окружење – Start Command Prompt with Ruby

Коришћење командне линије и интерактивног окружења (irb)

Ово поглавље даће ти основну листу команди које ћеш користити за покретање и тестирање **Ruby** кода. Такође, показаће ти како да користиш интерактивно окружење програма **Ruby** за кодирање – **irb**.

Непознате речи:

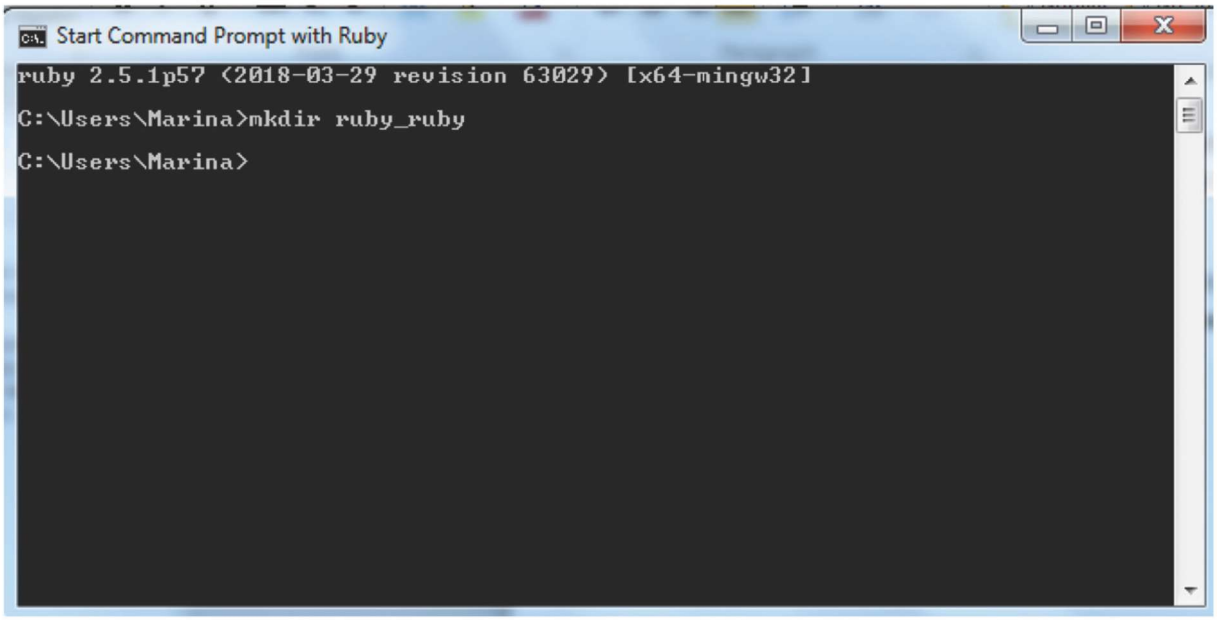
Интерактивно окружење Окружење које омогућава двосмерну интеракцију између корисника и рачунара, таблета и неког другог уређаја.

Командна линија

У петом разреду сазнали смо и како да пронађемо жељену дестинацију, директоријум, фајл на свом рачунару. То знање биће ти потребно и корисно за уношење жељене дестинације, тј. путање фајлова и директоријума.

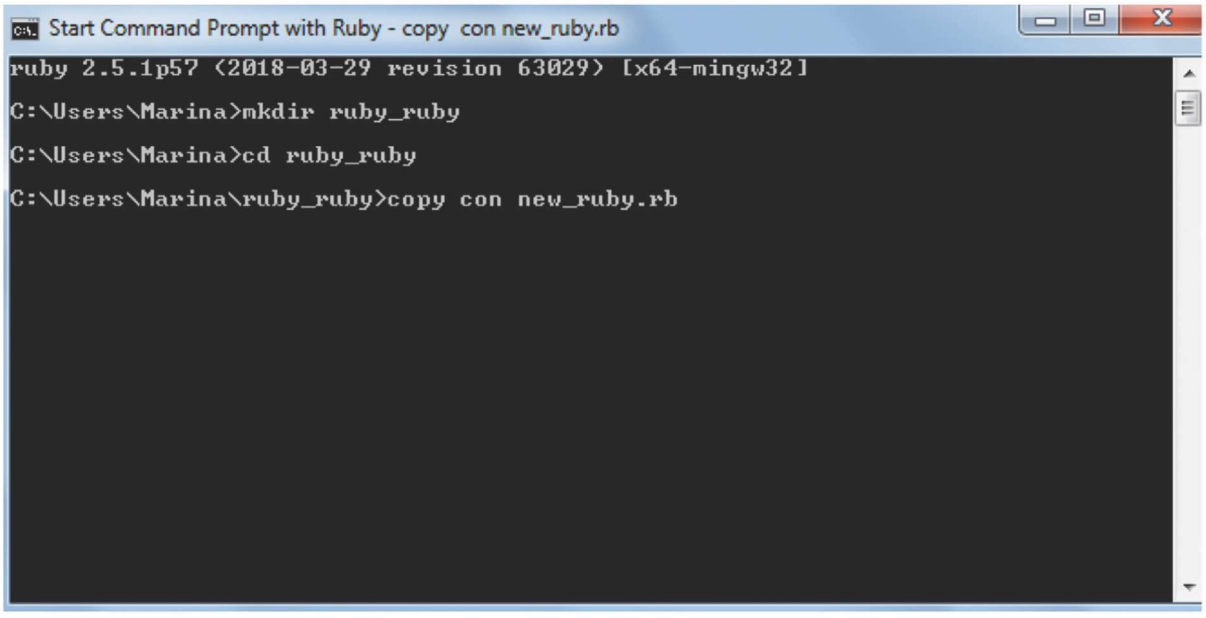
Симбол > у командној линији **Start Command Prompt with Ruby** означава да је потребно унети команду у командну линију.

Да бисмо креирали директоријум, додељујемо му назив **'ruby_ruby'**, и уносимо команду **mkdir ruby_ruby** у командну линију **Start Command Prompt with Ruby** (сл. 3.5). Команда **mkdir** потиче од енглеских речи **Make Directory,** које у преводу значе „направи директоријум".



Слика 3.5. Креирање директоријума

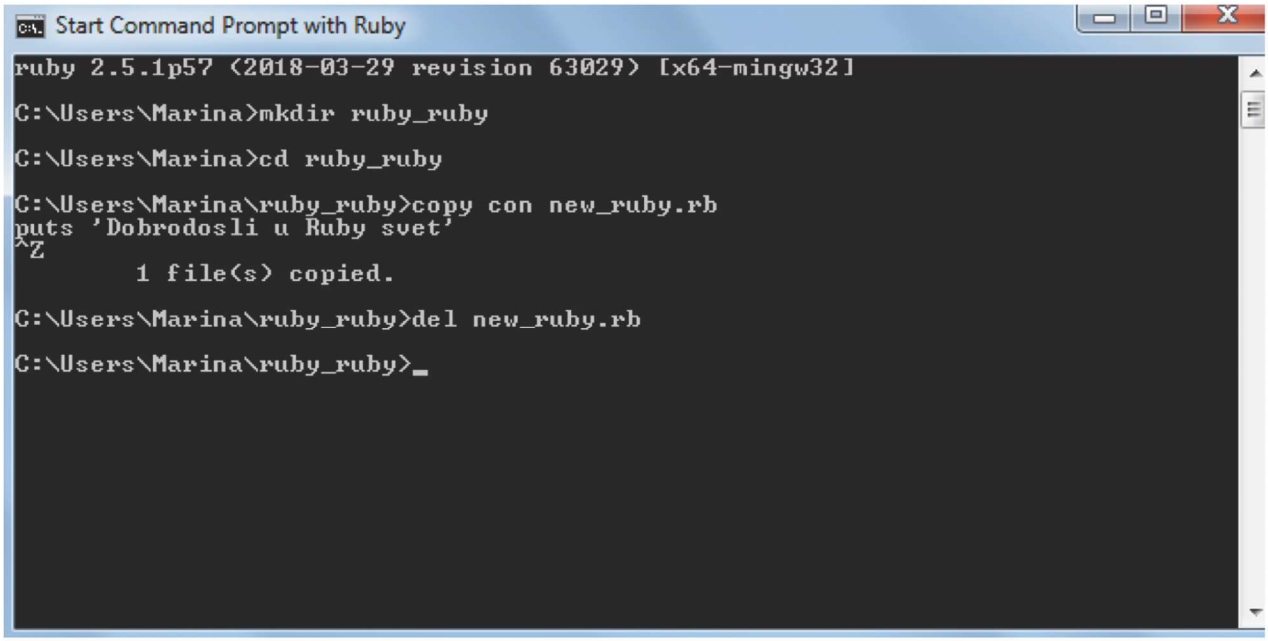
Да бисмо креирали фолдер у директоријуму који смо направили, у командну линију уносимо команду **cd**, а потом назив фолдера '**new_ruby.rb**' и уносимо команду **copy con** у следећу командну линију **Start Command Prompt with Ruby** (сл. 3.6). Команда **cd** потиче од енглеске речи **Change Directory**, која у преводу значи „промени директоријум".



Слика 3.6. Креирање фајла

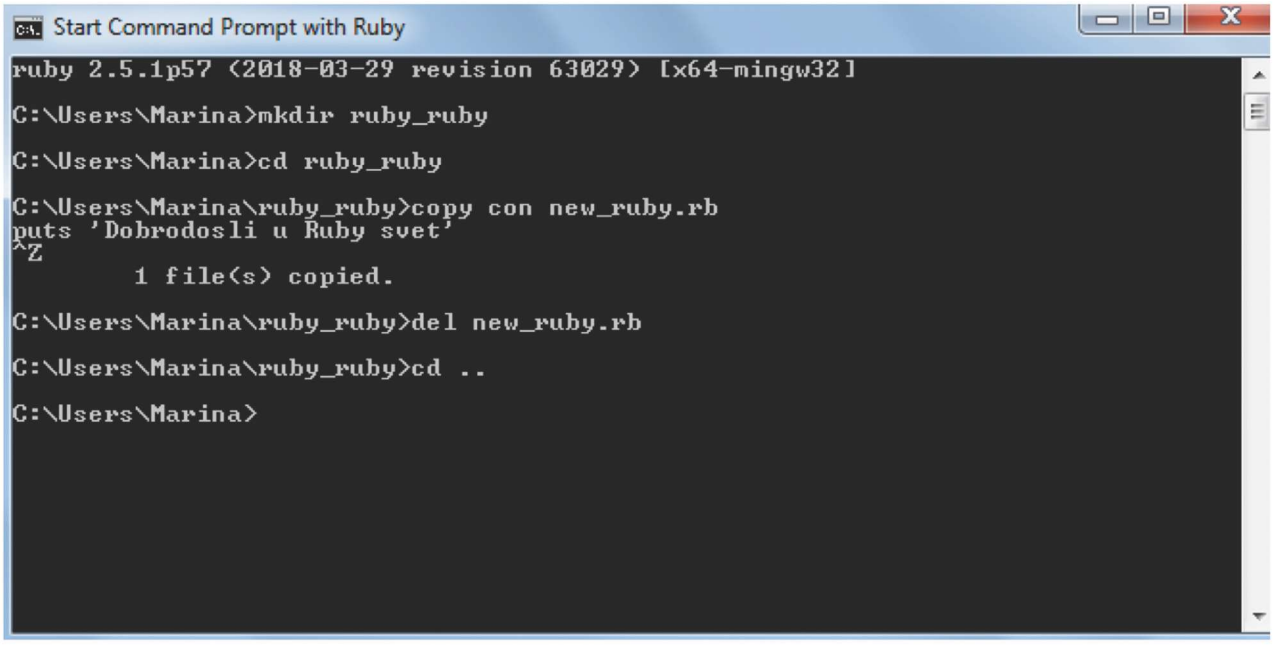
Команда **copy con** омогућава нам креирање фајла. Да бисмо креирали и сачували фајл у нашем директоријуму, у њега уписујемо кôд, а потом притиснемо тастер **Ctrl + Z**, након чега настављамо рад у следећој командној линији **Start Command Prompt with Ruby**.

Да бисмо обрисали фајл који смо управо креирали или било који други фајл, уносимо команду **del** у наредну командну линију, па потом назив фајла који желимо да обришемо из директоријума. У нашем примеру, то је фајл са називом **new_ruby.rb** (сл. 3.7). Команда **del** потиче од енглеске речи **Delete**, која у преводу значи „обрисати".



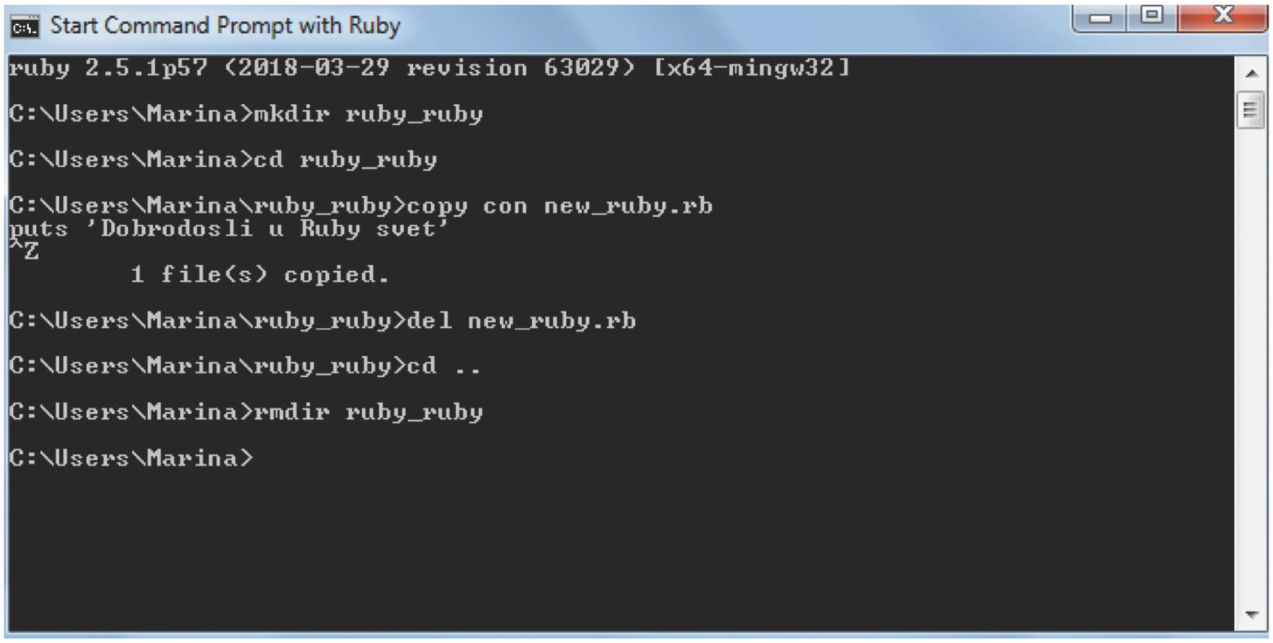
Слика 3.7. Брисање фајла

За излазак из директоријума у ком се налазимо, уносимо команду **cd ..** у командну линију и извршењем команде напуштамо директоријум (сл. 3.8).



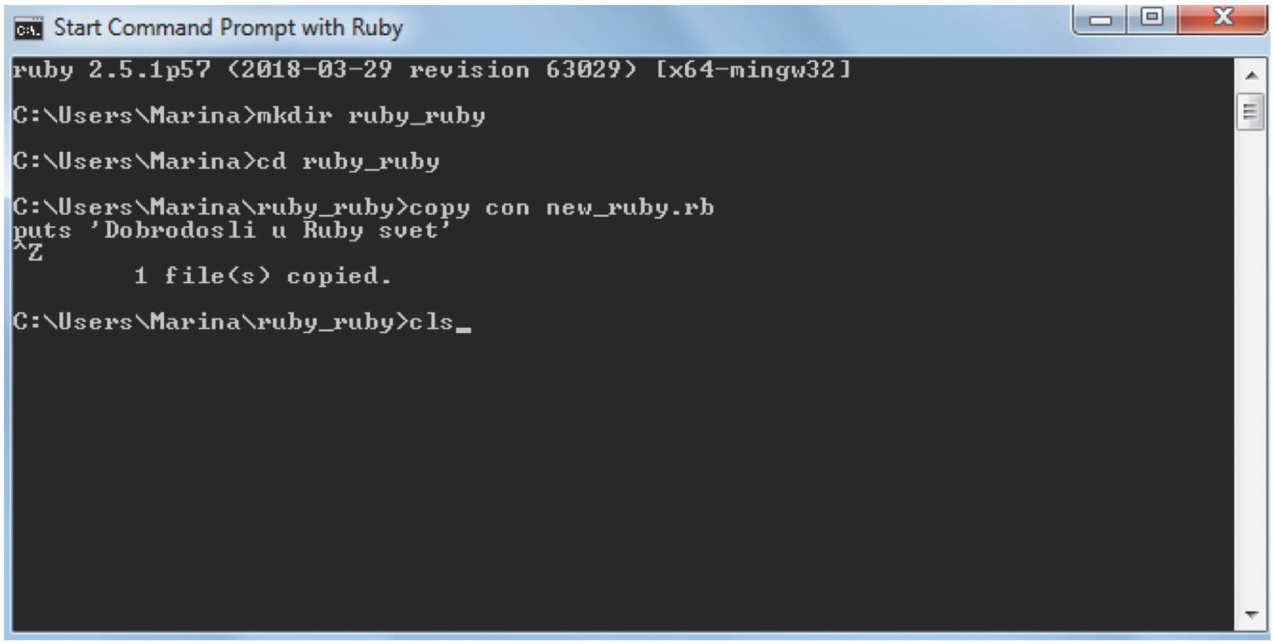
Слика 3.8. Излазак из директоријума у коме се налазимо

Да бисмо обрисали директоријум који смо управо креирали или било који други директоријум, уносимо команду **rmdir** у командну линију, па потом и назив директоријума **ruby_ruby** (сл. 3.9). Команда **rmdir** потиче од енглеске речи **Remove Directory** која у преводу значи „уклони директоријум”.



Слика 3.9. Брисање директоријума

Такође, можемо да обришемо све команде са екрана и започнемо с новим уносима. Да бисмо то урадили, уносимо команду **cls** у командну линију (сл. 3.10). Команда **cls** потиче од енглеске речи **Clean Screen**, која у преводу значи „очисти екран”.



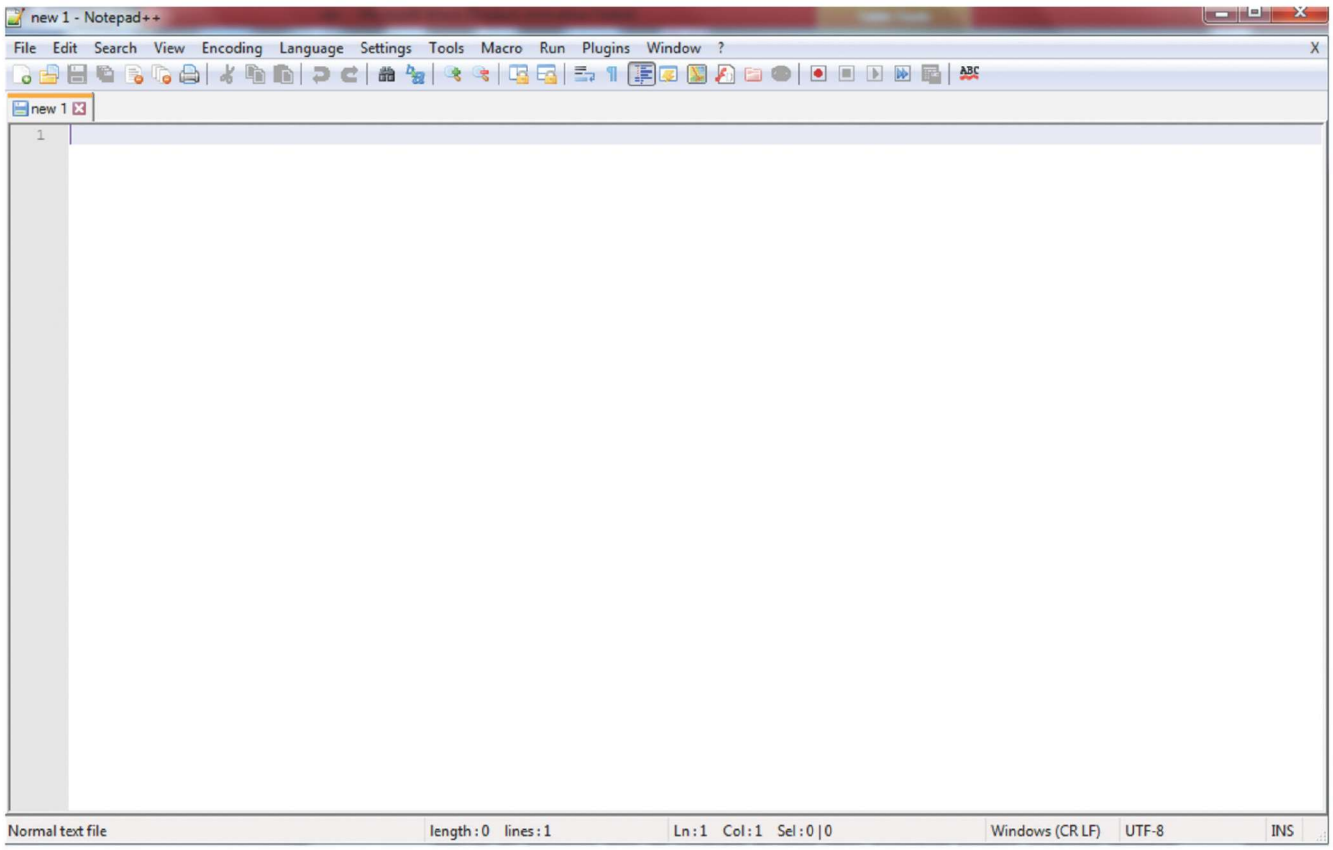
Слика 3.10. Брисање свих унетих команди

Запамти

Команда **mkdir** – креирање директоријума
Команда **copy con** – креирање фајла
Команда **del** – брисање фајла
Команда **cd ..** – излазак из активног директоријума
Команда **rmdir** – брисање директоријума
Команда **cls** – брисање свих унетих команди са екрана

Ово су све команде које ћеш користити приликом формирања фајлова у програмском језику **Ruby**. Претпостављамо да умеш да пронађеш пут до одређених фолдера, да се крећеш до различитих датотека и да користиш основне команде.

Start Command Prompt with Ruby веома је користан приликом креирања директоријума, фајлова, уноса кодова у фајлове итд. У њему, такође, можемо креирати и фајлове са екстензијом **.txt** на исти начин на који смо креирали фајл са **.rb** екстензијом. За рад у њима можемо користити програм **Notepad++**, један од текст едитора, у којем можемо писати програмске кодове. Изглед његовог радног окружење приказан је на слици 3.11.



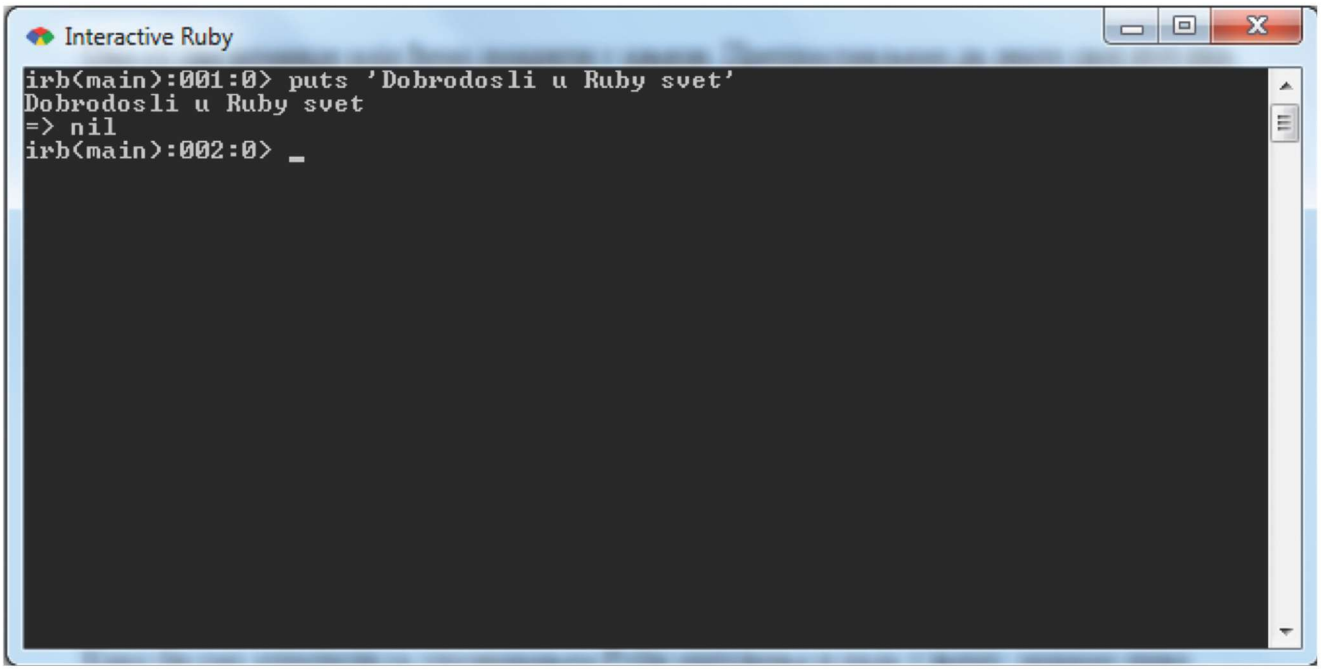
Слика 3.11. Радно окружење текст едитора Notepad++

Програм **Notepad++** можемо сврстати у једноставније текст едиторе, тј. програме за обраду текста, будући да не садржи велики број команди којима можемо вршити форматирање текста.

Управо из тог разлога је погодан за писање кодова, односно рачунарског програма и за то се најчешће користи. Карактерише га прегледност унетог текста, а самим тим је поправљање, тј. кориговање текста, проналажење грешака и пропуста једноставније, што га чини погоднијим за рад.

Интерактивно окружење – *irb*

Ruby има уграђено интерактивно окружење које је названо "**irb**". У њему ћемо писати и тестирати сваки наш **Ruby** кôд, па ћемо се мало боље упознати са њим (сл. 3.12).



Слика 3.12. Интерактивни Ruby с приказаним записом

Након покретања интерактивног окружења "**irb**", у првој линији добили смо испис irb(main):001:0>. Шта нам он говори?

Испис упућује на то да се налазимо у интерактивном окружење "**irb**" програма **Ruby** (irb(main)), као и да се налазимо у првој линији програма (001:0). После симбола (>) уносимо команду.

У програмски језик **Ruby** команду уносимо абецедом јер он не препознаје ћирилична нити латинична слова (ч, ћ, ш, ђ, ж, љ, њ). Да бисмо извршили унос, укуцали смо команду **puts** 'Dobrodosli u Ruby svet', кликнули ентер и добили приказ нашег захтева на екрану – Dobrodosli u Ruby svet, док нам је као вредност нашег захтева узвраћена => **nil** вредност. Након извршења нашег захтева, "**irb**" нам омогућава да унесемо нови захтев у следећој линији програма irb(main):002:0>.

Још један симбол који ће нам олакшати сналажење у програмском коду јесте симбол #. Он нам омогућава унос коментара у било којој линији програмског кода, а нема утицај на извршавање програмског кода.

Запамти

Команда **puts** – омогућава нам унос који ће бити приказан на екрану.
Вредност **nil** – представља недефинисану вредност.
Симбол (**=>**) – након симбола, програм нам враћа вредност нашег захтева.
Симбол (**#**) – унос коментара у програмски кôд без утицаја на његово извршење.

3.3. Основне аритметичке операције

Кључне речи:

симбол, бројеви, аритметичке операције

Како бисмо отпочели са упознавањем **Ruby** окружења и рада у њему, морамо прво да сазнамо шта је то симбол, као и шта су и какви могу бити бројеви. Потом ћемо прећи на основне програмске операције.

Симбол

Ruby симбол креирамо тако што у колони испред укуцане речи или реченице додајемо (:) – две тачке.

Симбол користимо када желимо да дефинишемо нешто, али не желимо да га прикажемо на екрану, нити да га мењамо. Често га називамо неизмењивим стрингом. Са појмом стринга и радом са њим детаљније ћемо се упознати у делу Уџбеника намењеном обради стринга.

Навешћемо пример (# Primer 1):

```
# Primer 1

:ime
:adresa
:" Ova recenica takodje predstavlja simbol "
```

Бројеви

У програмском језику **Ruby**, бројеви се могу представити на много начина.
Можемо их представити целобројним записом, као и децималним бројем.
Основни облик броја назива се целим бројем (приказује се цео број, без децималне тачке).
Навешћемо пример (# Primer 2):

```
# Primer 2  ceo broj
1, 2, 5, 35, 510, 1234567890
```

Децимални број је број који садржи децимално место и децималну тачку.

Навешћемо пример (# Primer 3):

```
# Primer 3  decimalni broj

1.2345, 2345.67, 34.5678
```

Аритметичке операције – сабирање, одузимање, множење и дељење целих бројева

Сабирање

Основне математичке операције у програмском језику **Ruby** прилично су једноставне. За сабирање два броја користи се (+) оператор, као што је приказано у следећим примерима (# Primer 4 и # Primer 5).

Покушај да вежбаш и са другим бројевима.

Непознате речи:

Оператор
Оператор је део израза који нам говори која операција треба бити извршена.

```
# Primer 4

irb(main):001:0 > 1 + 10
=> 11
```

```
# Primer 5

irb(main):001:0 > 1000 + 111
=> 1111
```

Одузимање

Да бисмо извршили операцију одузимања два броја, користићемо (-) оператор, као што је приказано у следећим примерима (# Primer 6 и # Primer 7). Провежбај и са другим бројевима.

```
# Primer 6

irb(main):001:0 >  10 - 1
=> 9
```

```
# Primer 7

irb(main):001:0 > 1000 - 111
=> 889
```

Множење бројева

Да бисмо извршили операцију множења бројева, користићемо (*) оператор, као што је приказано у следећим примерима (# Primer 8 и # Primer 9).

```
# Primer 8

irb(main):001:0 > 10 * 1
=> 10
```

```
# Primer 9

irb(main):001:0 >  1000 * 33
=> 33000
```

Дељење бројева

Да бисмо извршили операцију дељења бројева, користићемо (/) оператор, као што је приказано у следећим примерима (# Primer 10 и # Primer 11).

```
# Primer 10

irb(main):001:0 >  10 / 2
=> 5
```

```
# Primer 11

irb(main):001:0 > 1000 / 500
=> 2
```

Погледај следећи пример (# Primer 12) и уочи неправилност.

```
# Primer 12

irb(main):001:0 > 11 / 2
=> 5
```

Да ли смо добили тачан резултат? Извршићемо проверу и приказати је у следећем примеру (# Primer 13).

```
# Primer 13

irb(main):001:0 >  5 * 2
=> 10
```

Примећујш да смо као резултат провере добили број 10 уместо броја 11. Зашто? То се дешава зато што нам програм приликом дељења целих бројева не приказује остатак броја. Да бисмо приказали и остатак броја, морамо увести децималну тачку, односно децимални број. Покушајмо поново, али овог пута са децималним записом броја (# Primer 14).

```
# Primer 14

irb(main):001:0 >  11.0 / 2
=> 5.5
```

Следи провера (# Primer 15).

```
# Primer 15

irb(main):001:0 > 5.5 * 2
=> 11
```

Добили смо тачан резултат.

Наишли смо на проблем код дељења целим бројем. Шта се дешава са остатком броја приликом дељења целог броја, да ли га некако можемо израчунати и приказати?

Одговор је потврдан.

Користићемо наведени пример дељења целих бројева, где смо као резултат приликом дељења броја 11 бројем 2 добили број 5 и доказали да то није тачан резултат (# Primer 16).

```
# Primer 16

irb(main):001:0 > 11 / 2
=> 5
```

Користи (%) оператор, као што је приказано у следећем примеру, и израчунаћеш остатак броја приликом дељења. Покушај да вежбаш и са другим бројевима.

А сад да пробамо да израчунамо остатак (# Primer 17).

```
# Primer 17

irb(main):001:0 > 11 % 2
=> 1
```

Видимо да смо као резултат приликом дељења бројава 11 и 2, добили остатак 1. Да ли је остатак тачан? Извршићемо проверу, и то тако што ћемо резултат из # Primera 16 помножи-

ти са делиоцем и потом додати остатак добијен у # Primeru 17. Погледај следећи пример (# Primer 18).

```
# Primer 18

irb(main):001:0 > 5 * 2 + 1=> 11
```

Запамти

Када користимо (%) оператор, израчунаћемо само остатак броја приликом целобројног дељења.

Операције и примери које смо приказали једноставни су и можете их извршити и помоћу папира и оловке, а њиховом применом кроз програмски кôд можете извршити врло комплексне рачунске операције (# Primer 19).

```
# Primer 19

irb(main):001:0 >  5.35 * 2.78 * 210.13
=> 3125.26349
```

Помоћу следеће табеле подсетићемо се аритметичких оператора (таб. 3.1).

симбол оператора	опис функције оператора
+	сабира вредности које се налазе с леве и десне стране оператора;
-	одузима вредност која се налази с десне стране оператора од вредности која се налази с леве стране оператора;
*	множи вредности које се налазе с леве и десне стране оператора;
/	дели вредност која се налази с леве стране оператора са вредношћу која се налази с десне стране оператора
%	израчунава остатак након дељења целих бројева.

Табела. 3.1. Листа аритметичких оператора

3.4. Уграђене функције

Кључне речи:

функције, корисничке функције, уграђене функције

Функције су основни део, тј. елемент сваког програма, писаног у било ком програмском језику. Оне су део програма који обавља одређени задатак, кад год их унесемо у наш програм.

Колико пута ћемо их унети у програм зависи искључиво од тога колико нам је пута потребно да програм изврши одређени задатак.

Запамти

Симбол (==) означава једнакост вредности које се налазе са леве и десне стране оператора.
Унос функције у програм == позивање функције у програм

Функције се још називају и методе, процедуре, рутине, итд.
Основна подела функција је на:
1. корисничке функције и
2. уграђене функције.

Корисничке функције јесу функције које прави сам корисник у складу с потребама његовог програма. Оне представљају именовани и самостални део кода који извршава одређени задатак дефинисан од стране корисника. Функција је именована када јој корисник додели властити и јединствени назив, а самостална, јер извршава свој део задатка без обзира на остатак програма.

Уграђене функције јесу функције које су саставни део софтвера за писање нашег програма и њих обезбеђује сам произвођач. Уграђене функције садржи и програмски језик **Ruby**, у ком ћемо радити.

Уграђене функције су такође именоване и самосталне, али за разлику од корисничких функција, њима су називи већ додељени и не можемо их мењати, већ их само можемо применити приликом писања кода нашег програма.

Табеларно ћемо приказати уграђене функције програмског језика **Ruby** (таб. 3.2). Није потребно учити их напамет. Учење напамет листе функција или њихових објашњења при програмирању у било ком програмском језику није добро, а ни корисно. Намена уграђених функција је да нам олакшају обављање одређеног задатка. Можемо позвати односно применити уграђену функцију уколико нам одговара радња коју ће она извршити како би смо обавили задатак.

Објаснићемо намену појединих уграђених функција приказаних у табели: **Abort** – прекида извршавање програма, **Array(obj)** – враћа објекат након што га претвори у низ помоћу оператора **obj.to_a**, **Integer(obj)** – враћа објекат након што га претвори у цео број помоћу

оператора **obj.to_i**, **loop {...}** – понавља блок кодова који се налази унутар витичасте заграде, **gets([rs = $/])** – чита назив датотеке наведене у командној линији, **String(obj)** – враћа објекат након што га претвори у стринг помоћу оператора **obj.to_s**. Кроз примере које ћемо обрађивати у овом поглављу Уџбеника научићеш практично да примениш и користиш поједине уграђене функције.

Уколико те интересује да сазнаш више и упознаш се са преосталим функцијама можеш истраживати и ширити своје знање уз помоћ следећег линка https://ruby-doc.org/docs/ruby-doc-bundle/Manual/man-1.4/function.html.

Листа уграђених функција у програмском језику **Ruby**			
abort	exit([result = 0])	open(path[, mode = “r”])	sleep([sec])
Array(obj)	exit!([result = 0])	p(obj)	split([sep[, max]])
at_exit {...}	fail(...)	print([arg...])	sprintf(fmt[, arg...])
autoload(classname, file)	Float(obj)	printf(fmt[, arg...])	srand([seed])
binding	fork fork {...}	proc {| x|...} proc	String(obj)
block_given?	format(fmt[, arg...])	putc(c)	syscall(sys[, arg...])
callcc {| c|...}	gets([rs = $/])	puts([str])	system(cmd[, arg...])
caller([n])	global_variables	raise(...)	sub(x, y)
catch(tag) {...}	gsub(x, y) gsub(x) {...}	rand([max = 0])	sub!(x, y) sub!(x) {...}
chomp([rs = $/])	gsub!(x, y)	readline([rs = $/])	test(test, f1[, f2])
chomp!([rs = $/])	Integer(obj)	readlines([rs = $/])	throw(tag[, value = nil])
chop	lambda {| x|...} proc {| x|...}	require(lib)	trace_var(var, cmd)
chop!	load(file[, private = false])	scan(re) scan(re) {|x|...}	trap(sig, cmd) trap(sig) {...}
exec(cmd[, arg...])	loop {...}	set_trace_func(proc)	untrace_var(var[, cmd])

Табела 3.2. Листа уграђених функција у програмском језику **Ruby**

3.5. Стрингови (низови)

Кључне речи:

стринг, повезивање, конверзија

Стринг, тј. низ представља листу карактера који се налазе у одређеном низу. Стрингови су окружени једноструким знацима навода (‘Добар дан’) или двоструким знацима навода (“Добар дан”). На оба начина можемо стварати низ.

Карактер је основна словна информација у програму, он се односи на слово, број или знак интерпункције.

Уколико желимо да се у нашем стрингу прикаже појединачни цитат, то можемо урадити на следећа два начина приказана у примерима (# Primer 20 и # Primer 21).

```
# Primer 20: dvostruki znaci navoda

irb(main):001:0 >  "Nastavnica je rekla, Dobar dan!'"
```

```
# Primer 21: jednostruki znaci navoda

irb(main):001:0 >  'Nastavnica je rekla, \'Dobar dan!\"
```

У примеру 21 појављује се знак (\). Он говори рачунару да знаци навода који се налазе унутар стринга укључују у њега наведене цитиране знакове. Двоструки знаци навода дозвољавају нам да урадимо повезивање **Ruby** кода са стрнговима. Основна синтакса је: # {**Ruby** кôд иде овде}, а враћени израз биће спојен са стрингом. Приказаћемо повезивање **Ruby** кода са стрнговима у следећем примеру (# Primer 22).

```
# Primer 22:  interpolacija stringa

irb(main):001:0 > a = ‘informatika’
=> “informatika”

irb(main):002:0 > “Moj omiljeni predmet je #{a}!”
=> “Moj omiljeni predmet je informatika!”
```

Запамти

Повезивање **Ruby** кода са стрнговима функционише само када се налази између двоструких наводника.

Поређење једнакости

Често се дешава да је потребно да проверимо да ли су две вредности једнаке. Програмски језик **Ruby** дозвољава нам да то и урадимо на једноставан начин. Да бисмо тестирали једнакост две ствари, можемо користити (==) оператор. Применом тог оператора упоредићемо објекте с леве стране оператора са објектом на десној страни оператора. Потом ћемо добити информације да ли је поређење истинито или лажно, добићемо приказ (**true**) за истинитост, односно једнакост вредности или приказ (**false**), уколико поређење није једнако.

Испробаћемо то на следећем примеру (# Primer 23).

```
# Primer 23:  poredjenje jednakosti

irb(main):001:0 >  7 == 7
=> true

irb(main):002:0 >  7 == 2
=> false
```

Примећујемо да смо, поредећи број 7 са бројем 7, добили као одговор **true**, што значи да су објекти с леве и са десне стране једнаки, док смо приликом поређења броја 7 са бројем 2 као одговор добили **false**, што значи да објекти нису једнаки. Такође, овакав начин поређења можемо применити и код поређења стрингова, што ћемо испробати на следећем примеру (# Primer 24).

```
# Primer 24:  poredjenje jednakosti

irb(main):001:0 >  'dobar' == 'dobar'
=> true

irb(main):002:0 >  'dobar' == 'dobra'
=> false
```

До сада смо видели да можемо поредити бројеве као и стрингове, а да ли можемо извршити поређење броја и стринга, покушаћемо да сазнамо из следећег примера.

Испробаћемо то на следећем примеру (# Primer 25).

```
# Primer 25:  poredjenje jednakosti

irb(main):001:0 >  7 == 7
=> true

irb(main):002:0 >  '7' == '7'
=> true
```

При поређењу два иста броја, као у нашем примеру, једнакост леве и десне стране је тачна, тј. истинита, као и код поређења два стринга. Међутим, увиђамо да када поредимо стринг с леве стране и број са десне стране, једнакост није тачна, тј. објекти с леве и са десне стране нису једнаки.

Повезивање стрингова

При раду са стринговима у програмском језику **Ruby** можемо извршити повезивање стрингова. Како ћемо то урадити и шта ћемо добити као резултат, закључићемо из следећих примера (# Primer 26).

```
# Primer 26:  povezivanje stringova

irb(main):001:0 >   'uradi' + 'ti'
=> "uraditi"

irb(main):002:0 >   'dobro' + 'nameran'
=> "dobronameran"
```

Погледај следећи пример (# Primer 27). Шта мислиш, шта ћемо добити као резултат?

```
# Primer 27:  povezivanje stringova

irb(main):001:0 >  '2' + '3'
```

Логично је да помислиш да ћемо као резултат добити број 5, јер ако саберемо бројеве 2 и 3 добијамо број 5, али то није тачан одговор. Бројеви су приказани као стрингови и не можемо их сабрати. Овим поступком вршимо повезивање стрингова и добијамо као резултат стринг '23', приказан у следећем примеру (# Primer 28).

```
# Primer 28:  povezivanje stringova

irb(main):001:0 >  '2' + '3'
=> '23'
```

Шта мислиш да ли можемо да саберемо стринг и број? Хајде да то испробамо у следећем примеру (# Primer 29).

```
# Primer 29:  povezivanje stringa i celog broja

irb(main):001:0 >  '2' + 3
=> TypeError: no implicit conversion of Fixnum into String
   from (irb):2:in `+'
   from (irb):3
   from /usr/local/rvm/rubies/ruby-2.0.0-rc2/bin/irb:16:in `<main>'
```

Као одговор, добили смо грешку у постављању захтева, зато што не можемо да сабирамо, тј. број не можемо да додајемо стрингу, као ни стринг броју и немамо резултат. На који начин можемо да коригујемо наш захтев како би смо добили жељени резултат?

Морамо да извршимо конверзију стринга у број или обрнуто конверзију броја у стринг, што ћемо показати у следећем поглављу.

Конверзија стринга

Непознате речи:

Конверзија стринга
Конверзија стринга је поступак претварања стринга у други тип података.

Уколико желимо да саберемо стринг и број, користићемо исти захтев. Напоменули смо да морамо извршити конверзију стринга како бисмо их повезали.

Наш захтев (# Primer 30).

```
# Primer 30:  konverzija stringa

irb(main):001:0 >  '2' + 3
```

Конверзију стринга ћемо извршити на следећи начин – стринг у нашем примеру је ‘2’. Како бисмо га сабрали са бројем 3, морамо извршити његову конверзију, и то тако што ћемо приликом уношења захтева стрингу додати ознаку која га конвертује у број (**to_i**). То ћемо приказати у следећем примеру (# Primer 31).

```
# Primer 31:  konverzija stringa

irb(main):001:0 >   '2'.to_i
=> 2
```

Када смо савладали конверзију стринга, наш захтев можемо поставити у правилном облику, као што је приказано у следећем примеру и добићемо жељени резултат (# Primer 32).

```
# Primer 32:  konverzija stringa

irb(main):001:0 >  '2'.to_i + 3
=> 5
```

Најчешћи оператори приликом вршења конверзија су:

to_i – конвертује стринг у цео број;
to_f – конвертује стринг у децимални број;
to_s – конвертује број у стринг.

Испробај ове операторе конверзије код куће или у школи, па продискутуј о добијеним резултатима на часу.

3.6. Променљиве

Кључне речи:
променљиве,
вредност променљиве

Променљиве се користе ради обезбеђивања места у меморији рачунара за складиштење података. Оне имају назив и тренутну вредност. Вредност може бити број, слово или текст.

На променљиве можемо гледати као на контејнер који садржи информације и податке. Њихова једина сврха је означавање, складиштење и чување података у меморији.

Додела вредности променљивима

Један од најтежих задатака у програмирању јесте додела назива променљивој. Када променљивој додељујемо назив, морамо се трудити да он буде прецизан и разумљив другом програмеру.

Када додељујемо вредност променљивој, користићемо симбол (=). Име променљиве уносимо с леве стране симбола, а вредност коју желимо да доделимо променљивој и да је сачувамо, уносимо са десне стране симбола, као у следећем примеру (# Primer 33).

```
Primer 33:  dodela vrednosti promenjivoj

irb(main):001:0 >  moje_ime = 'Marina'
=> "Marina"

# pozvacemo promenjivu kako bi smo proverili da li smo pravilno uneli njenu
vrednost

irb(main):002:0 >  moje_ime
=> "Marina"
```

У претходном примеру, за променљиву **moje_ime** сачували смо у меморији и њену вредност приказану стрингом **'Marina'**.

Пре него што прочиташ објашњење, приказаћемо пример у ком смо променљивој **x** придружили вредност 3, променљивој **y** вредност **x**, а потом смо променљивој **x** променили вредност у 5. Шта мислиш, колика је вредност променљиве **y** након промене вредности променљиве **x**? Погледај следећи пример и продискутуј о томе на часу. (# Primer 34).

Primer 34: dodela vrednosti promenjivoj

```
Primer 34:  dodela vrednosti promenjivoj

irb(main):001:0 >  x = 3
=> 3

irb(main):002:0 >  y = x
=> 3

irb(main):003:0 >  x = 5
=> 5

irb(main):004:0 >  y
=> 3
```

Видимо да је вредност која је меморисана за променљиву **x**=3, као и да је вредност за променљиву **y**, која је једнака променљивој **x**, такође, меморисана вредност 3. Потом промењивој **x** додељујемо нову вредност 5, која се меморише. Када бисмо од нашег програма затражили информацију колика је вредност променљиве **y**, добили бисмо да је њена вредност и даље 3, без обзира на промену вредности променљиве **x**.

Зашто је то тако?

Променљиве нам дозвољавају да мењамо њихове меморисане вредности, али у нашем случају, морали бисмо поново да доделимо вредност промењивој **y**, како би се она променила. Уколико то не урадимо, без обзира на промене вредности променљиве **x**, вредност променљиве **y** увек ће бити једнака 3, јер је тако уписана у меморију.

Погледајте следећи пример и тестирајте га кроз програм како бисте увидели шта се догађа с вредностима променљивих **x** и **y** (# Primer 35).

```
Primer 35:  dodela vrednosti promenjivoj

irb(main):001:0 >  x = 3
=> 3

irb(main):002:0 >  y = x
=> 3

irb(main):003:0 >  x = 5
=> 5

irb(main):004:0 >   y
=> 3
```

```
irb(main):004:0 >   y
=> 3

irb(main):005:0 >  x
=> 5

irb(main):006:0 >  y = x
=> 5

irb(main):007:0 >  y
=> 5
```

Није неопходно да вредности променљивима доделимо ми, већ можемо написати кôд тако да вредност променљивој додели корисник. То ћемо урадити користећи методу **get.chomp**. Она кориснику даје могућност да унесе тражену информацију и да притиском на тастер Enter добије приказани одговор на постављени захтев.

Запамти

Метода **get.chomp** – омогућава кориснику да унесе и додели вредност за тражену променљиву.

Приказаћемо поступак у следећем примеру (# Primer 36).

```
Primer 36:  dodela vrednosti promenjivoj od strane korisnika

irb(main):001:0 >  ime = gets.chomp
# ocekujemo da korisnik unese ime i pritisne taster enter
Jovan
# na ekranu se ispisuje sledeci zapis
=> “Jovan”

irb(main):002:0 >  prezime = gets.chomp
Jovic
=> “Jovic”
```

Након уноса вредности променљивих од стране корисника, с променљивима можемо да радимо на креативан начин. Уколико променљивој **ime**, за коју је вредност унео корисник (Jovan), желимо да придодамо следећи испис **‘je najbolji djak!‘**, програм ће нам вратити испис на екрану као у следећем примеру (# Primer 37).

```
Primer 37:

irb(main):001:0 > ime
=> “Jovan”

irb(main):002:0 > ime + ‘ je najbolji djak! ‘
=> “Jovan je najbolji djak!”
```

3.7. Методе које користи програмски језик Ruby

Кључне речи:

методе, дефинисање методе, услови, поређења, комбиновани изрази

Програмери се често сусрећу с проблемом да одређени део кода мора да се изврши много пута у току програма. Како би се избегао тај проблем, у већини програмских језика постоји опција која се назива процедура. Она омогућава да се заједнички кôд постави на једно место.

У програмском језику **Ruby**, процедуру називамо методом. Како бисмо користили методу, морамо употребити за њу резервисану реч **def**. Након дефинисања методе додељујемо јој назив. После тога морамо ставити резервисану реч **end**, како бисмо означили завршетак њеног дефинисања.

Запамти

Дефинисање методе вршимо помоћу резервисане речи **def.**
Завршетак дефинисања методе вршимо помоћу резервисане речи **end.**

Употреба методе приказана је на следећем примеру (# Primer 38).

```
Primer 38:  definisanje metode

irb(main):001:0 >  def metod

irb(main):002:0 >  end
```

Сазнали смо како се дефинишу и именују методе. У следећим примерима дефинисаћемо методе сабирања и одузимања. Следи пример дефинисања методе сабирања (# Primer 39).

```
Primer 39:  definisanje metode sabiranja

irb(main):001:0 >  def sabiranje (a, b)

irb(main):002:0 >  a + b

irb(main):003:0 >  end

# pozvacemo definisanu metodu i izvrsiti unos vrednosti za koje nam je potreban rezultat
irb(main):004:0 >  sabiranje(10, 13)
 => 23
```

Након дефинисања методе сабирања, у било ком тренутку писања програма можемо позвати методу сабирања и добити, као резултат, тражени збир. Исто правило важи и за следећи пример дефинисања методе одузимања (# Primer 40).

```
Primer 40:  definisanje metode oduzimanja

irb(main):001:0 >  def oduzimanje (a, b)

irb(main):002:0 >  a - b

irb(main):003:0 >  end

# pozvacemo definisanu metodu i izvrsiti unos vrednosti za koje nam je potreban rezultat
irb(main):004:0 >  oduzimanje(13, 10)
 => 3

# Izracunavamo i negativne brojeve
irb(main):005:0 >  oduzimanje(10, 13)
 => -3
```

У следећим примерима (# Primer 41 и # Primer 42) дефинисаћемо и методу множења и дељења.

```
Primer 41:  definisanje metode mnozenja

irb(main):001:0 >  def mnozenje (a, b)

irb(main):002:0 >  a * b

irb(main):003:0 >  end

# pozvacemo definisanu metodu i izvrsiti unos vrednosti za koje nam je potreban rezultat
irb(main):004:0 >  mnozenje (3, 7)
 => 21
```

```
Primer 42:  definisanje metode deljenja

irb(main):001:0 >  def deljenje (a, b)

irb(main):002:0 >  a / b

irb(main):003:0 >  end

# pozvacemo definisanu metodu i izvrsiti unos vrednosti za koje nam je potreban rezultat
irb(main):004:0 >  deljenje (21, 7)
 => 3
```

Покушајмо још мало да се поиграмо методама. Можемо дефинисати методу која ће помножити или поделити већ дефинисане методе сабирања и одузимања. Приказаћемо пример множења наведених дефинисаних метода, а ти код куће или у школи, покушај да самостално урадиш пример дељења већ дефинисаних метода сабирања и одузимања (# Primer 43).

```
# Primer 43:  rad sa definisanim metodama

irb(main):001:0 >  def mnozenje_metoda (metoda_sabiranja,
metoda_oduzimanja)

irb(main):002:0 >  metoda_sabiranja *  metoda_oduzimanja

irb(main):003:0 >  end

# pozvacemo definisanu metodu i izvrsiti unos vrednosti za koje nam je potreban rezultat
irb(main):004:0 >  mnozenje_metoda (sabiranje (10,13), oduzimanje (13,10))
 => 69
```

Услови, поређења, комбиновани изрази, тачно и нетачно

Програмирање у програмском језику **Ruby**, као и у осталим програмским језицима, захтева креирање и праћење тока захтева како би смо дошли до његовог решења. У току програмирања, морамо постављати услове, поредити вредности, комбиновати изразе, добијати тачне или нетачне тврдње итд. Како бисмо све то постигли, морамо да се упознамо са симболима операција које ћемо користити. Набројаћемо све које смо до сада упознали, а симболе и њихова значења приказаћемо табеларно (табела. 3.3).

1. Операције поређења <=, <, >, > =
2. Операције једнакости ==, ! =
3. Логичка операција &&
4. Логичка операција ||

Табеларни приказ симбола операција и њихова значења (табела 3.3)

Симбол операције	значење симбола
<=	мање или једнако, вредност која се налази с леве стране симбола мања је или једнака вредности која се налази са десне стране симбола;
<	мање, вредност која се налази с леве стране симбола мања је од вредности која се налази са десне стране симбола;
>	веће, вредност која се налази с леве стране симбола већа је од вредности која се налази са десне стране симбола;
> =	веће или једнако, вредност која се налази с леве стране симбола већа је или једнака вредности која се налази са десне стране симбола;
==	једнако, вредност која се налази с леве стране симбола једнака је вредности која се налази са десне стране симбола;
! =	није једнако – различито, вредност која се налази с леве стране симбола није једнака вредности која се налази са десне стране симбола;
&&	и, вредност која се налази са леве стране симбола и вредност која се налази са десне стране симбола испуњавају постављени услов;
||	или, вредност која се налази с леве стране симбола или вредност која се налази са десне стране симбола испуњава постављени услов.

Табела. 3.3. Листа симбола операција и њихова значења

Да бисмо савладали примену симбола операција, урадићемо следеће задатке.

Задатак бр. 1

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност исказа да је број 4 мањи од броја 6.

```
Resenje zadatka br. 1

irb(main):001:0 > 4 < 6
=> true   # tacno
```

Задатак бр. 2

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност исказа да је број 2 већи од броја 5.

```
Resenje zadatka br. 2

irb(main):001:0 > 2 > 5

=> false   # netacno
```

Задатак бр. 3

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност исказа да је број 34 мањи или једнак броју 35.

```
Resenje zadatka br. 3

irb(main):001:0 > 34 <= 35

=> true   # tacno
```

Задатак бр. 4

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност исказа да је број 77 већи или једнак броју 77.

```
Resenje zadatka br. 4

irb(main):001:0 > 77 >= 77
=> true   # tacno
```

Задатак бр. 5

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност исказа да је број 32 једнак броју 32.

```
Resenje zadatka br. 5

irb(main):001:0 > 32 == 32

=> true   # tacno
```

Задатак бр. 6

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност исказа да је број 45 једнак броју 46.

```
Resenje zadatka br. 6

irb(main):001:0 > 45 == 46

=> false   # netacno
```

Задатак бр. 7

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност исказа да је број 45 различит од броја 46.

```
Resenje zadatka br. 7

irb(main):001:0 > 45 != 46

=> true   # tacno
```

Задатак бр. 8

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност исказа да је број 45 различит од броја 45.

```
Resenje zadatka br. 8

irb(main):001:0 > 45 != 45
=> false   # netacno
```

Задатак бр. 9

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност исказа да је број 45 различит од броја 46, као и да је број 73 једнак броју 73.

```
Resenje zadatka br. 9

irb(main):001:0 > (45 != 46) && (73 == 73)

=> true   # tacno
```

Задатак бр. 10

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност исказа да је број 45 различит од броја 45, као и да је број 73 једнак броју 73.

```
Resenje zadatka br. 10

irb(main):001:0 > (45 != 45) && (73 == 73)

=> false   # netacno
```

Задатак бр. 11

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност једног од исказа: први исказ гласи да је број 45 различит од броја 23, а други исказ да је број 73 једнак броју 72.

```
Resenje zadatka br. 11

irb(main):001:0 > (45 != 23) || (73 == 72)

=> true   # tacno
```

Задатак бр. 12

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити тачност једног од исказа: први исказ гласи да је број 23 једнак броју 22, а други исказ да је број 72 једнак броју 73.

```
Resenje zadatka br. 12

irb(main):001:0 > (23 == 22) || (72 == 73)

=> false   # netacno
```

Задатак бр. 13

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити да ли први и други исказ имају једнаке вредности. За први исказ, унеси променљиве **a**, **b** и **c**, додели им произвољне вредности, а њихов збир представљаће вредност промењиве **d**. За други исказ, разлика вредности промењивих **c** и **a** представљаће вредност промењиве **e**.

```
Resenje zadatka br. 13

irb(main):001:0 > a = 3
=> 3

irb(main):002:0 > b = 5
=> 5

irb(main):003:0 > c = 7
=> 7

irb(main):004:0 > d = a + b + c
=> 15

irb(main):005:0 > e = c – a
=> 4

irb(main):006:0 > d == e
=> false
# netacno, vrednost prvog iskaza iznosi 15, vrednost drugog iskaza iznosi 4, iz cega zakljucujemo da nemaju jednake vrednosti.
```

Задатак бр. 14

Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити да ли први и други исказ имају једнаке вредности. За први исказ унеси променљиве **a** и **b**, а за други исказ унеси променљиве **c** и **d**; додели им произвољне вредности.

Збир промењивих **a** и **b** представљаће вредност променљиве **e**, док ће разлика промењивих **c** и **d** представљати вредност променљиве **f**.

```
Resenje zadatka br. 14

irb(main):001:0 > a = 6
=> 3

irb(main):002:0 > b = 5
=> 5
```

```
irb(main):003:0 >  c = 7
=> 7

irb(main):004:0 >  d = 3
=> 3

irb(main):005:0 >  e = a + b
=> 11

irb(main):006:0 >  f = c - d
=> 4

irb(main):007:0 >  e == f
=> false
# netacno, vrednost prvog iskaza iznosi 11, vrednost drugog iskaza iznosi 4, iz cega zakljucujemo da nemaju jednake vrednosti.
```

3.8. Гранање

Кључне речи:
гранање, изјава случаја

Гранање у програмирању најчешће везујемо за разгранате алгоритме и разгранату програмску структуру.

Гранање је дефинисано испуњавањем одређеног услова постављеног у задатку програма. У програмском језику **Ruby**, у коме до решења задатка долазимо гранањем, задатак можемо написати користећи основне логичке структуре које имају своје резервисане речи. Помоћу тих резервисаних речи **case** (случај), **when** (кад), **else** (иначе) и **end** (крај), као и помоћу резервисаних речи **if** (ако), **elsif** (ако иначе) и **else** (иначе), писаћемо потребне кодове за задатке који следе.

У случају примене резервисаних речи **case**, **when**, **else** и **end**, наш кôд исписујемо као изјаву случаја који смо поставили, док применом резервисаних речи **if**, **elsif** и **else** тражимо потврдан или негативан одговор на наш захтев. Приказаћемо и тестирати задатке за оба случаја (Задатак 15, Задатак 16, Задатак 17 и Задатак 18).

Задатак 15.

Напиши програмски кôд у интерактивном **irb** окружењу применом резервисаних речи изјаве случаја. Неопходно је кориснику омогућити да унесе било који број до броја 10. У случају да корисник унесе број 5, програм ће вратити испис “tvoj broj je 5”, док у случају да корисник унесе број 2, програм ће вратити испис “tvoj broj je 2”. Иначе, испис који ће приказати програм уколико корисник није унео ни број 5, ни број 2, гласи “broj koji si izabrao nije ni 2 a ni 5”.

```
Resenje zadatka br. 15

irb(main):001:0 >  puts “Unesi broj do 10”

irb(main):002:0 >  a = gets.chomp.to_i

# korisnik unosi broj

irb(main):003:0 >  case a

irb(main):004:0 >  when  5

irb(main):005:0 >  puts “ tvoj broj je 5”

irb(main):006:0 >  when  2

irb(main):007:0 >  puts “ tvoj broj je 2”
```

Задатак 16.

Напиши програмски кôд у интерактивном **irb** окружењу применом резервисаних речи изјаве случаја. Неопходно је омогућити кориснику да унесе било који број до броја 10, који ће представљати вредност промењиве **a**. У случају да корисник унесе број 5, програм ће вратити испис "tvoj broj je 5", док у случају да корисник унесе број 2, програм ће вратити испис "tvoj broj je 2". Иначе, испис који ће програм приказати уколико корисник није унео ни број 5, ни број 2, гласи "broj koji si izabrao nije ni 2 a ni 5". На крају задатка прикажи вредност променљиве коју је унео корисник.

```
Resenje zadatka br. 16

irb(main):001:0 >  puts "Unesi broj do 10"

irb(main):002:0 >  a = gets.chomp.to_i

# korisnik unosi broj

irb(main):003:0 >  answer  =  case a

irb(main):004:0 >  when  5

irb(main):005:0 >  " tvoj broj je 5"

irb(main):006:0 >  when  2

irb(main):007:0 >  " tvoj broj je 2"

irb(main):008:0 >  else

irb(main):009:0 >  " broj koji si izabrao nije ni 2 a ni 5"

irb(main):010:0 >  end

irb(main):011:0 >  puts answer
```

Задатак 17.

Напиши програмски кôд у интерактивном **irb** окружењу применом резервисаних речи помоћу којих тражимо потврдан или негативан одговор на наш захтев. Неопходно је омогућити кориснику да унесе било који број до броја 10. Ако је корисник унео број 5, програм ће вратити испис "tvoj broj je 5", а ако је корисник унео број 2, програм ће вратити испис "tvoj broj je 2". У супротном, испис који ће приказати програм уколико корисник није унео ни број 5, ни број 2, гласи "broj koji si izabrao nije ni 2 a ni 5".

```
Resenje zadatka br. 17
irb(main):001:0 >  puts "Unesi broj do 10"
irb(main):002:0 >  a = gets.chomp.to_i
# korisnik unosi broj
irb(main):003:0 >  if  a == 5
irb(main):004:0 >  puts " tvoj broj je 5"
irb(main):005:0 >  elsif  a == 2
irb(main):006:0 >  puts " tvoj broj je 2"
irb(main):007:0 >  else
irb(main):008:0 >  puts " broj koji si izabrao nije ni 2 a ni 5"
irb(main):009:0 >  end
```

Задатак 18.

Напиши програмски кôд у интерактивном **irb** окружењу применом резервисаних речи помоћу којих тражимо потврдан или негативан одговор на наш захтев. Неопходно је унети промењиве **a**, **b**, и **c**. Промењива **a** садржаће вредност 7, промењива **b** вредност 9, а промењива **c** вредност 2. Ако је испуњен услов да је **((a+b)/c)** мање или једнако броју 8, програм ће вратити испис "tvoj racun je tacan". Иначе, ако је задовољен услов да је **(a – b + c)** једнако броју 1, програм ће вратити испис "tvoj racun je netacan". Ако нису задовољени ни један, ни други услов, програм ће вратити испис "nisam siguran koji je rezultat".

Resenje zadatka br. 18

```
irb(main):001:0 >  a =7

irb(main):002:0 >  b = 9

irb(main):003:0 >  c =  2

irb(main):004:0 >  if ( (a + b)/c) <= 8

irb(main):005:0 >  puts " tvoj racun je tacan "

irb(main):006:0 >  elsif ( a – b + c) == 1

irb(main):007:0 >  puts " tvoj racun je netacan "

irb(main):008:0 >  else

irb(main):009:0 >  puts " nisam siguran koji je rezultat "

irb(main):010:0 >  end
```

3.9. Понављање

Кључне речи:

петља, WHILE петља, DO/WHILE петља, FOR петља

Понављање дела кода у програмирању називамо петљом и најчешће га везујемо уз цикличне алгоритме и цикличу програмску структуру.

Петља је понављање извршења дела кода за одређени број понављања или док не добијемо као резултат оно што задовољава наш постављени услов. У овом поглављу упознаћемо се с простим петљама, са њиховим извршењем, а потом и са **WHILE** петљом, **DO/WHILE** петљом и **FOR** петљом.

Најједноставнији начин креирања петље подразумева коришћење методе **loop**. Метода **loop** формира блок који се означава витачастом заградом {...} или блок **do ... end**.

Петља ће извршити кôд који се налази унутар блока, што значи само онај кôд који се налази унутар витичасте заграде, тј. између симбола {...} или унутар блока **do ... end**, и то све док ручно не прекинемо његово извршавање притиском тастера **Ctrl + c** или уметањем команде **break** у блок.

Запамти

Метода **loop** – омогућава понављање дела кода унутар програма све док не остваримо постављени захтев.

Следећим примером (# Primer 44) приказаћемо једноставну петљу и тестираћемо је.

```
# Primer 44:  jednostavna petlja

irb(main):001:0 >  loop do

irb(main):002:0 >  puts " Ponavljacu ovu recenicu, sve dok ne pritisnes Ctrl + c"

irb(main):003:0 >  end
```

Тестирањем ове једноставне петље програм је реченицу "Ponavljacu ovu recenicu, sve dok ne pritisnes Ctrl + c" исписивао на екрану небројано много пута и не заустављајући се све док нисмо притиснули тастер **Ctrl + c**.

WHILE петља

WHILE петљом извршавамо наредбе у програмском коду уз претходну проверу постављеног услова који се процењује као тачан или нетачан (**true** или **false**).

Извршавањем WHILE петље добијамо параметар који се назива **boolean**, он нам говори да ли је нешто тачно или лажно (нетачно). Када **boolean** израз постане лажан, WHILE петља се не извршава поново, а програм се наставља и након WHILE петље.

Запамти

Параметар **boolean –** даје нам информацију да ли је нешто тачно или лажно (**true** или **false**).

У следећем примеру креираћемо кôд који врши одбројавање од броја који унесе корисник, па до броја 0, а на крају извршења задатка приказује се испис сваког понављања на екрану (# Primer 45).

```
# Primer 45:  WHILE petlja

irb(main):001:0 >  x = gets.chomp.to_i

# korisnik unosi broj

irb(main):002:0 >  while x >=0

irb(main):003:0 >  puts x

irb(main):004:0 >  x = x - 1

irb(main):005:0 >  end

irb(main):006:0 >  puts "WHILE petlja"
```

DO/WHILE петља

DO/WHILE петља функционише на сличан начин као и **WHILE** петља, једина битна разлика је у томе што се након извршења кода унутар петље врши провера услова како би се видело да ли код мора поново да се изврши или не мора. У **DO/WHILE** петљи услов се налази на крају петље.

Креираћемо пример који поставља питање кориснику и у зависности од његовог одговора, петља ће се или извршити поново, или ће њено поновно извршење бити прекинуто (# Primer 46).

```
# Primer 46:  DO/WHILE petlja

irb(main):001:0 >  loop do

irb(main):002:0 >  puts “Da li zelite da ponovite procedure”

irb(main):003:0 >  answer = gets.chomp

irb(main):004:0 >  if answer = ‘NE’

irb(main):005:0 >  break

irb(main):006:0 >  end

irb(main):006:0 >  end
```

Видимо да смо применом команде **break** прекинули понављање петље, уколико је испуњен услов, односно уколико је корисник као одговор на постављено питање унео одговор **NE**.

FOR петља

FOR петља се користи код понављања над колекцијом елемената. За разлику од претходних петљи, FOR петља не може да буде бесконачна, будући да је њено понављање дефинисамо бројем елемената. Она мора започети резервисаном речи **for** коју мора да прати променљива, као и резервисаном речи **in**, након које се уносе елементи.

У следећем примеру (# Primer 47) тестираћемо FOR петљу.

```
# Primer 47:  FOR  petlja

irb(main):001:0 >  x = [ 1, 2, 3, 4, 5 ]

irb(main):002:0 >  for i in x do

irb(main):003:0 >  puts i

irb(main):004:0 >  end
```

3.10. Основне структуре података

Кључне речи:
структура података, низ, хаш табеле

Структуре података и алгоритми међусобно су тесно повезани и на основу њих се помоћу програмског језика пише потребан рачунарски програм.

У петом разреду било је речи о томе да програмске структуре могу бити линијске, цикличне и разгранате, у складу са основним алгоритмима. У шестом разреду покушаћемо да савладамо основне структуре података кроз примере и наш рад.

Најосновнију структуру података чине низови и хаш табеле, а на основу њих и од њих граде се сложеније структуре података којима се нећемо бавити, а то су стекови, редови за чекање итд.

Низови

Низови у процесу програмирања представљају сложени податак, будући да су сачињени од других података који могу бити различитог типа. Сваки податак који се налази у низу чини елемент низа, сваки елемент има одређени објекат преко ког можемо да приђемо само том елементу.

Основна подела низова је на:
– једнодимензионалне низове;
– дводимензионалне низове и
– вишедимензионалне низове.

Једнодимензионални низови представљају колекцију података односно елемената низа код којих број елемената представља дужину низа.

Дводимензионалне низове називамо матрицама или табелама, код којих податке односно елементе распоређујемо по врстама и колонама. Први индекс одређује ред док други индекс одређује колону у оквиру тог реда.

Вишедимензионалмни низови представљају скуп више дводимензионалних низова.

У програмском језику **Ruby**, као и у већини других програмских језика, низ дефинишемо тако што податке уносимо у угласту заграду **[ ]**. То ћемо приказати у следећем примеру (# Primer 48).

```
# Primer 48:  Definisanje niza sa razlicitim tipovima podataka

irb(main):001:0 >  [ 1, 2, 'ime', 4, 5.234, 'prezime' ]
```

У раду са низовима желимо да прикажемо одређени елемент низа, а то ћемо урадити позивањем његовог броја индекса, што значи да је свака позиција у низу нумерисана. Сваки елемент у низу има своју одређену позицију, као и свој јединствени редни број односно нумерацију. Његов јединствени редни број представља његов индекс. Из предходног примера желимо да прикажемо елемент чији је број индекса 4. Подели са одељењем очекивани елемент, па урадимо следећи пример (# Primer 49).

```
# Primer 49:  Prikazivanje elementa niza

irb(main):001:0 >  niz = [ 1, 2, 'ime', 4, 5.234, 'prezime' ]

irb(main):002:0 >  niz [ 4 ]
=> 5.234
```

Уместо очекиваног елемента 4, као резултат добили смо елемент децималног типа податка 5.234. Зашто?

То је зато што у низовима бројеви индекса елемената отпочињу са 0. Први елемент низа представљен је бројем индекса 0, други елемент бројем индекса 1, и тако редом, до краја низа.

У раду са низовима желимо да избришемо поједини елемент или да додамо нови; помоћу следећих примера савладаћемо поступке брисања и додавања елемената низу.

Додавање новог елемента вршићемо методом **push** (# Primer 50).

```
# Primer 50:  Dodavanje elementa nizu

irb(main):001:0 >  niz = [ 1, 2, 'ime', 4, 5.234, 'prezime' ]

irb(main):002:0 >  niz.push ('odeljenje')
=> [ 1, 2, 'ime', 4, 5.234, 'prezime', 'odeljenje' ]
```

Уколико желимо да избришемо последњи елемент у низу, користићемо методу **pop** (# Primer 51).

```
# Primer 51:  Brisanje poslednjeg elementa niza

irb(main):001:0 >  niz = [ 1, 2, 'ime', 4, 5.234, 'prezime' ]

irb(main):002:0 >  niz.pop
=> 'prezime'

irb(main):003:0 >  niz
=> [ 1, 2, 'ime', 4, 5.234 ]
```

Уколико желимо да избришемо неки од елемената низа, користићемо методу **delete_at** и методу **delete** (# Primer 52 и # Primer 53).

```
# Primer 52:  Brisanje odredjenog elementa niza pomocu broja indeksa elementa

irb(main):001:0 >  niz = [ 1, 2, 'ime', 4, 5.234, 'prezime' ]

irb(main):002:0 >  niz.delete_at ( 3 )
=> 4

irb(main):003:0 >  niz
=> [ 1, 2, 'ime', 5.234, 'prezime' ]
```

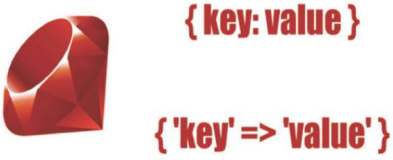
```
# Primer 53:  Brisanje odredjenog elementa niza pomocu vrednosti elementa

irb(main):001:0 >  niz = [ 1, 2, 'ime', 4, 5.234, 'prezime' ]

irb(main):002:0 >  niz.delete ( 4 )
=> 4

irb(main):003:0 >  niz
=> [ 1, 2, 'ime', 5.234, 'prezime' ]
```

Запамти

Метода **push** – омогућава додавање елемената у низ.
Метода **pop** – омогућава уклањање, тј. брисање последњег елемента у низу.
Метода **delete_at** – омогућава брисање одређеног елемента низа на основу броја индекса елемента.
Метода **push** – омогућава брисање одређеног елемента низа на основу вредности елемента.

Хаш табеле

Хаш табеле представљају структуру података која чува и препознаје елементе на основу придружених кључева. Хаш, у суштини представља колекцију парова **key-value** (кључ-вредност) (сл. 3.13). Како бисмо лакше разумели хаш табелу, навешћемо неколико примера:



Слика 3.13. Колекција парова key-value

Пример колекције парова **key-value**
“ucenik” = > “ocena”
“zaposleni” = > “plata”
“doktor” = > “pacijent”

Унос кључева у хаш табелу најчешће се врши коришћењем симбола, док се за вредност хаша може користити било који тип података. Хаш табелу формирамо тако што кључ-вредност мора да се постави унутар витичасте заграде { }, иза симбола (**:**); кључ од његове вредности раздвајамо симболом (= >). Да бисмо лакше разумели и применили унос кључева у хаш табелу, извршићемо приказ и проверу следећим примером (# Primer 54).

```
# Primer 54:  Formiranje Hash-a

irb(main):001:0 >  hash = { : ucenik = > ‘ ocena ‘}
=> { : ucenik = > ‘ ocena ‘ }
```

Хаш можемо формирати на још један начин, који ћемо приказати следећим примером. Када извршиш тестирање оба начина, изабери онај који ти највише одговара (# Primer 55).

```
# Primer 55:  Formiranje Hash-a

irb(main):001:0 >  hash = { ucenik : ‘ ocena ‘ }
=> { : ucenik = > ‘ ocena ‘ }
```

Хаш структура података намењена је обради велике количине података, па ћемо унос за више података приказати у следећем примеру (# Primer 56).

```
# Primer 56:  Unos vise podataka u Hash

irb(main):001:0 >  ucenik  = { ime : ‘ Jovan ‘ , prezime : ‘ Jovic ‘ }
=> { : ime = > ‘ Jovan ‘,  : prezime = > ‘ Jovic ‘ }
```

Из претходног примера видимо да можемо уносити велики број парова уз раздвајање симболом зареза (,). Додаћемо пар у формирани хаш и уклонити пар из њега. Поступке ћемо приказати у следећим примерима (# Primer 57 и # Primer 58).

```
# Primer 57:  Dodavanje elemenata  Hash-u

irb(main):001:0 >  ucenik [ : visina] = ‘ 155 cm ‘
=> “ 155 cm “

irb(main):002:0 >  ucenik [ : tezina] = ‘ 50 kg ‘
=> “ 50 kg “

# izvrsili smo dodatni unos za visinu i tezinu naseg ucenika, i kada pozovemo hash
dobijamo sledece podatke

irb(main):003:0 >  ucenik
=> { : ime = > ‘ Jovan ‘,  : prezime = > ‘ Jovic ‘,  : visina = > ‘ 155 cm ‘,
: tezina = > ‘ 50 kg ‘ }
```

```
# Primer 58:  Uklanjanje elemenata  Hash-a

irb(main):001:0 >  ucenik.delete ( : visina )
=> “ 155 cm “

# izvrsili smo uklanjanje elemenata za visinu naseg ucenika, i kada pozovemo hash
dobijamo sledece podatke

irb(main):002:0 >  ucenik
=> { : ime = > ‘ Jovan ‘,  : prezime = > ‘ Jovic ‘,  : tezina = > ‘ 50 kg ‘ }
```

Научили смо да хаш представља колекцију парова **key-value** (кључ-вредност). На који начин можемо приказати вредност за поједини кључ сазнаћемо из следећег примера (# Primer 59).

```
# Primer 59:  Prikazivanje vrednosti za trazeni kljuc

irb(main):001:0 >  ucenik [ : prezime]
=> “ Jovic “
```

Навели смо да хаш углавном садржи велики број елемената, а биће неопходно да кроз рад прикажемо све елементе или применимо одређене методе на њима. Како бисмо то могли да урадимо, употребићемо **each** методу и приказати поступак у следећем примеру (# Primer 60).

```
# Primer 60:  Primena metode each

irb(main):001:0 >  ucenik  = { : ime = > ‘ Jovan ‘,  : prezime = > ‘ Jovic ‘,
: visina = > ‘ 155 cm ‘, : tezina = > ‘ 50 kg ‘ }

# primenjujemo metodu each

irb(main):002:0 >  ucenik.each do | key, value |

irb(main):003:0 >  puts “ Jovan ’ ovo  #{ key]  je  #{ value }”

irb(main):004:0 >  end

# dobijamo sledeci ispis elemenata

Jovanovo ime je Jovan
Jovanovo prezime je Jovic
Jovanovo visina je 155 cm
Jovanovo tezina je 50 kg
```

При раду са хаш табелама биће нам потребно да проверимо да ли се у хашу налази одређени кључ који нас интересује. То можемо проверити применом методе **has_key** на врло једноставан начин приказан у следећем примеру (# Primer 61).

```
# Primer 61:  Pretraga kljuca

irb(main):001:0 >  ucenici  = { “ Jovan ” = > 15,  “ Marko” = > 10,
“ Zoran “ = > 17}
= > { “ Jovan ” = > 15,  “ Marko” = > 10,  “ Zoran “ = > 17}

irb(main):002:0 >  ucenici.has_key? ( Marko )
= > true

rb(main):003:0 >  ucenici.has_key? ( Djordje )
= > false
```

Да ли можемо да прикажемо само кључеве из одређеног хаша или само вредности које се у њему налазе?

Можемо, али хајде да то испробамо кроз следеће примере (# Primer 62 и # Primer 63).

```
# Primer 62:  Prikaz svih kljuceva hash-a

irb(main):001:0 >  ucenici  = { " Jovan " = > 15,  " Marko" = > 10,
" Zoran " = > 17}
= > { " Jovan " = > 15,  " Marko" = > 10,  " Zoran " = > 17}

irb(main):002:0 >  ucenici.keys
= > [ " Jovan ", " Marko ", " Zoran " ]
```

```
Primer 63:  Prikaz svih vrednosti hash-a

irb(main):001:0 >  ucenici  = { " Jovan " = > 15,  " Marko" = > 10,
" Zoran " = > 17}
= > { " Jovan " = > 15,  " Marko" = > 10,  " Zoran " = > 17}

irb(main):002:0 >  ucenici.values
= > [ 15, 10, 17 ]
```

Запамти

Низ представља структуру података која чува и препознаје елементе на основу њиховог броја индекса.

Хаш табеле представљају структуру података која чува и препознаје елементе на основу придружених кључева.

3.11. Основни алгоритми

Кључне речи:
алгоритми, линијски, циклични, разгранати

Подсетићемо се алгоритама и њихове поделе јер они представљају основу за сваког програмера, а такође, на основу једног алгоритма можемо написати програм у било ком програмском језику.

Алгоритам представља кораке којима решавамо одређени проблем; у већини случајева приказује се графички и има свој дијаграм тока. Алгоритми могу бити линијски, циклични и разгранати.

Подсетићемо се ознака које се најчешће користе приликом писања графичке шеме алгоритма (сл. 3.14), као и графичких шема линијских, цикличних и разгранатих алгоритама, уз приказ одговарајуће програмске структуре кроз **irb** окружење.

3.14. Ознаке за писање алгоритма

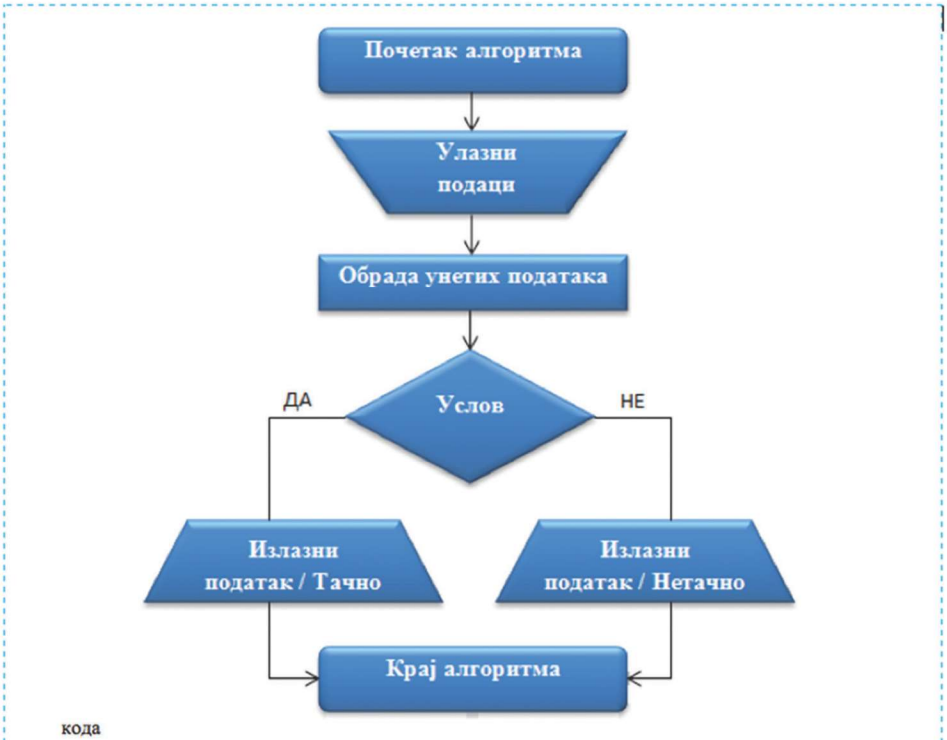
1. Линијски алгоритам је алгоритам у ком се сваки корак извршава само једанпут. Линијски алгоритам приказаћемо графичком шемом алгоритма и линијском програмском структуром у програмском коду, у **irb** окружењу (сл. 3.15).

Слика 3.15. Приказ графичке шеме линијског алгоритам и линијског програмског кода

2. Циклични агоритам је алгоритам у ком се одређени корак понавља док неки услов алгоритма не постане задовољен, или одређени број пута. Приказаћемо циклични алгоритам графичком шемом алгоритма и цикличном програмском структуром у програмском коду, у **irb** окружењу (сл. 3.16).

Слика 3.16. Приказ графичке шеме цикличног алгоритам и цикличног програмског кода

3. Разгранати алгоритам је алгоритам који садржи корак у ком се проверава неки постављени услов. У зависности од тога да ли је услов испуњен или није, извршавају се наредни кораци алгоритма или се пак поједини кораци поново извршавају. Приказаћемо разгранати алгоритам графичком шемом алгоритма и разгранатом програмском структуром у програмском коду, у **irb** окружењу (сл. 3.17).



кода

```
Primer 66:
irb(main):001:0 >  a =2
irb(main):002:0 >  b = 4
irb(main):003:0 >  c =  6
irb(main):004:0 >  if ( (a + b) * c) <= 36
irb(main):005:0 > puts “ tacno”
irb(main):006:0 >  elsif ( a + b - c) == 1
irb(main):007:0 >  puts “netacno”
irb(main):008:0 >  else
irb(main):009:0 >  puts “ nisam siguran koji je rezultat”
irb(main):010:0 >  end
```

Слика 3.17. Приказ графичке шеме разгранатог алгоритма и разгранатог програмског кода

Пројектни задатак

Пројектни задатак на крају другог полугодишта можете урадити користећи програмски језик **Ruby**. Као и у претходном пројектном задатку, неопходно је одабрати тему пројектног задатка, формирати групе, тј. тимове који ће дефинисати поступке за реализацију задатка, прикупљати информације, графички представити алгоритам пројектног задатка, у интерактивном окружењу "**irb**" написати и тестирати **Ruby** кôд задатка и извршити демонстрацију и презентацију пројектног задатка.

Пример задатка

У програмском језику **Ruby** написати кôд помоћу ког сваки ученик одељења може да унесе своје оцене за предмете у шестом разреду и добије просечну оцену за сваки предмет, као и за крајњи успех.

Приликом писања кода потребно је користити аритметичке операције – сабирање, одузимање, множење и дељење целих бројева, као и променљиве којима треба доделити вредности. Приликом писања кода, дефинишу се методе које су нам неопходне за извршење пројектног задатка. У зависности од креативности чланова тима који израђују графички алгоритам задатка, користи се разграната програмска структура или циклична програмска структура. Такође, у зависности од плана израде задатка, користе се потребне структуре података.

Фазе израде пројектног задатка

Пројектни задатак има следеће фазе:

- подела на групе;
- избор теме задатка;
- дефинисање рока за реализацију пројектног задатка;
- дефинисање поступака потребних за реализацију пројектног задатка;
- прикупљање информација;
- израда графичког алгоритма за реализацију пројектног здатка;
- писање и тестирање кода у интерактивном окружењу "**irb**";
- демонстрација и презентација пројектног задатка.

Увидели сте да је у фазама израде пројектног задатка прво наведена подела на групе, па потом избор теме. Наш предлог задатка вам можда није претерано занимљив, па зато можете да се организујете да свака група предложи по једну тему, а потом да цело одељење разматра идеје, дискутује о њима и одабере једну тему која ће бити предмет пројектног задатка. Уколико немате или не знате како да искажете своју идеју, посаветујте се с предметним наставницима, јер се овај пројектни задатак израђује путем програмског језика на часовима Информатике и рачунарства, али његова тема може бити везана за било који други предмет или област која вам је интересантна и занимљива.

Питања за проверу знања

1. Шта је програмски језик?
2. Наброј програмске језике које знаш.
3. Шта креирамо применом команде **mkdir**?
4. Чему служи команда **cls**?
5. Објасни чему нам служи симбол (#).
6. Објасни дељење целог броја.
7. Када се приказује остатак приликом дељења целог броја?
8. Објасни аритметичке операторе.
9. Шта су корисничке функције?
10. Шта су уграђене функције?
11. Шта је стринг?
12. Објасни конверзију стринга.
13. Које операторе најчешће користимо приликом вршења конверзије.
14. Објасни на који начин додељујемо вредности променљивима.
15. Како вршимо дефинисање методе?
16. Објасни симболе операције поређења <=, <, >, > =
17. Објасни резервисане речи **case** (случај), **when** (кад), **else** (иначе) и **end** (крај).
18. Шта омогућава метода **loop**?
19. Коју информацију нам пружа параметар **Boolean**?
20. Шта представља низ у процесу програмирања?
21. Шта нам омогућва примена методе **push**?
22. Објасни разлику између цикличног и разгранатог алгоритма.
23. Напиши програмски кôд у интерактивном **irb** окружењу којим ћеш поделити број 15 са бројем 2, тако да добијеш тачан резултат без остатка.
24. Напиши програмски кôд у интерактивном **irb** окружењу којим ћеш поделити број 17 са бројем 5, тако да добијеш тачан резултат без остатка.
25. Напиши програмски кôд у интерактивном **irb** окружењу којим ћеш проверити тачност исказа да је број 45 мањи од броја 63.
26. Напиши програмски кôд у интерактивном **irb** окружењу којим ћеш проверити тачност исказа да је број 31 мањи или једнак броју 33.

27. Напиши програмски кôд у интерактивном **irb** окружењу којим ћеш проверити тачност исказа да је број 25 различит од броја 25.
28. Напиши програмски кôд у интерактивном **irb** окружењу којим ћеш проверити тачност исказа да је број 145 различит од броја 246, као и да је број 23 једнак броју 23.
29. Напиши програмски кôд у интерактивном **irb** окружењу којим ћеш проверити тачност једног од исказа: први исказ гласи да је број 145 различит од броја 223, а други да је број 173 једнак броју 172.
30. Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити да ли први и други исказ имају једнаке вредности. За први исказ унеси промењиве **a** и **b**, додели им произвољне вредности, њихов збир ће представљати вредност промењиве **d**. За други исказ разлика вредности промењивих **b** и **a** представљаће вредност промењиве **e**.
31. Напиши програмски кôд у интерактивном **irb** окружењу, којим ћеш проверити да ли први и други исказ имају једнаке вредности. За први исказ унеси промењиве **a** и **b**, а за други исказ унеси промењиве **c** и **d** и додели им произвољне вредности. Разлика промењивих **a** и **b** представљаће вредност промењиве **e**, док ће збир промењивих **c** и **d** представљати вредност промењиве **f**.
32. Напиши програмски кôд у интерактивном **irb** окружењу применом резервисаних речи изјаве случаја. Неопходно је омогућити кориснику да унесе било који број до броја 7. У случају да корисник унесе број 3, програм ће вратити испис “tvoj broj je 3”, док у случају да корисник унесе број 4, програм ће вратити испис “tvoj broj je 4”. Иначе, испис који ће програм приказати уколико корисник није унео ни број 3, ни број 4, гласи “broj koji si izabrao nije ni 3 a ni 4”.
33. Прикажи графички алгоритам претходног задатка.
34. Напиши програмски кôд у интерактивном **irb** окружењу применом резервисаних речи помоћу којих тражимо потврдан или негативан одговор на наш захтев. Неопходно је омогућити кориснику да унесе било који број до броја 7. Ако је корисник унео број 3, програм ће вратити испис “tvoj broj je 3”, а ако је корисник унео број 4, програм ће вратити испис “tvoj broj je 4”. У супротном, испис који ће програм приказати уколико корисник није унео ни број 3, ни број 4, гласи “broj koji si izabrao nije ni 3 a ni 4”.
35. Прикажи графички алгоритам претходног задатка.
36. Напиши програмски кôд у интерактивном **irb** окружењу применом резервисаних речи помоћу којих тражимо потврдан или негативан одговор на наш захтев. Неопходно је унети промењиве **a**, **b**, **c** и **d**. Променљива **a** ће садржати вредност 4, променљива **b** вредност 7, променљива **c** вредност 12, а променљива **d** садржаће вредност збира промењивих **a** и **b**. Ако је испуњен услов да је **((a+b-c)/d)** мање или једнако броју 3, програм ће вратити испис “tvoj racun je tacan”. Иначе, ако је задовољен услов да је **(a-b+c-d)** једнако броју 2, програм ће вратити испис “tvoj racun je netacan”. Ако нису задовољени ни један, ни други услов, програм ће вратити испис “nisam siguran koji je rezultat”.

37. Прикажи графички алгоритам претходног задатка.

38. Напиши програмски кôд у интерактивном **irb** окружењу применом WHILE петље. Тај кôд врши одбројавање од броја који корисник унесе, па до броја 0. Кориснику треба омогућити да унесе било који број до броја 17. На крају извршења задатка потребно је приказати испис сваког понављања на екрану.

39. Прикажи графички алгоритам претходног задатка.

40. Напиши програмски кôд у интерактивном **irb** окружењу применом DO/WHILE петље који поставља питање кориснику. Питање формулиши самостално. У зависности од твог интересовања, као и од корисниковог одговора, петља ће се извршити поново или се неће извршити.

41. Прикажи графички алгоритам претходног задатка.

42. Напиши програмски кôд у интерактивном **irb** окружењу; у њему ћеш дефинисати низ са различитим типовима података који те описују. Прикажи елемент низа чији је број индекса елемента 3.

43. Напиши програмски кôд у интерактивном **irb** окружењу; у њему ћеш дефинисати низ са различитим типовима података који описују твој град и потом додати нови елемент низу.

44. Напиши програмски кôд у интерактивном **irb** окружењу; у њему ћеш дефинисати низ са различитим типовима података који описују твоју школу, а потом обриши елемент низа чији је број индекса елемента 2.

45. Напиши програмски кôд у интерактивном **irb** окружењу; у њему ћеш дефинисати низ са различитим типовима података који описују твој успех у школи, а потом обриши елемент низа по твом избору помоћу методе брисања елемента низа на основу вредности елемента.

46. Напиши програмски кôд у интерактивном **irb** окружењу; у њему ћеш формирати хаш табелу са елементима који садрже различите типове података који те описују.

47. Напиши програмски кôд у интерактивном **irb** окружењу у коме ћеш приказати све кључеве хаш табеле користећи податке из задатка број 46.

48. Напиши програмски кôд у интерактивном **irb** окружењу у коме ћеш приказати све вредности хаш табеле користећи податке из задатка број 46.

Речник

А

Алгоритам – графички приказани кораци којима решавамо одређени проблем

Апликативни софтвер –програм који извршава конкретне активности за корисничке потребе

Ауторско дело – оригинална творевина аутора

Б

Бекап – копија података

В

Веб-претраживач – компјутерски програм чији је основни задатак да омогући преглед веб-презентација

Вируси – злонамерни програми

Г

Графичка картица – компонента рачунара намењена за обраду и приказ визуелних података на одговарајућим излазним уређајима, нпр. на монитору

Д

Датотека (Фајл) – именовани, структурирани скуп података садржајно везаних, смештених на медију за меморисање

Дигитална писменост – Означава способност стварања, проналажења и преноса информација у дигиталном облику као и способност процене тачности информације као и развој практичних и функционалних вештина које су у вези са употребом дигиталних уређаја

Е

Едитовање – уређивање текста

Екстензија – представља додатак на име датотеке и служи да укаже на врсту њеног садржаја

З

Звучна картица – електронски уређај који обрађује звучне сигнале

И

Иконе – сличице које представљају програме или докумената која се чувају на рачунару

Информационо-комуникационе технологије (ИКТ) – обухватају разноврсне технолошке алате и ресурсе који се користе за пренос, складиштење, стварање, дељење и размену информација

Интерактивно окружење – окружење које омогућава двосмерну интеракцију између корисника и рачунара, таблета и неког другог уређаја

Интернет – највећа глобална мрежа рачунара

Информација – обрађени податак

К

Конверзија стринга – поступак претварања стринга у други тип података

Копирајт – симбол који означава да је садржај заштићен ауторским правима

М

Матична плоча – електронска штампана плоча која повезује све делове рачунара

Меморија – физички уређај који се користе за складиштење програма или података привремено или трајно

Метода – процедура у програмском језику

Монитор – електрични излазни уређај који служи за приказивање слике послате са другог уређаја, обично графичке карте у склопу рачунара

Мултимедија – комбинација различитих облика

Мултимедијална презентација – презентација која садржи текст, табеле, графиконе, слике, звук, видео-записе

Н

Нетикеција – правила лепог понашања на интернету

О

Оперативни систем – скуп програма и рутина одговорних за контролу и управљање уређајима и рачунарским компонентама

Оператор – део израза који нам говори која операција треба бити извршена

П

Плагијат – копирање и присвајање неког ауторског рада

Податак – једноставна изолована, необрађена чињеница која има неко значење

Провајдер – пружалац услуга интернета

Програмирање – поступак решавања неког проблема односно одређеног задатка, применом језика који рачунар разуме

Програм – целина која је састављена од кодова којима се решава неки проблем или задатак

Програмски језик – језик којим се пише код за рачунар

Променљива – обезбеђује место у меморији рачунара за складиштење података

Процедура – метода у програмском језику Ruby

Р

Радно окружење – радна површина у којој реализујемо свој задатак

Рачунарска графика – област визуелног рачунарства која омогућава креирање слика и њихову обраду

Рачунарски систем – јединство хардвера и софтвера

Резолуција – број пиксела по хоризонтали и вертикали

С

Селектовање – означавање

Скенер – уређај намењен преношењу садржаја с папира у дигитални облик

Слајд – основни елемент презентације

Софтвер – програми и подаци на рачунару

Стринг – представља листу карактера који се налазе у одређеном низу

Структура података – начин представљања података у рачунару

Т

Тастатура – периферни уређај рачунарског система који служи за комуникацију са рачунаром

Ћ

Ћелија – представља поље пресека реда и колоне у који се уноси податак

Ф

Фајл (Датотека) – именовани, структурирани скуп података садржајно везаних, смештених на медију за меморисање

Флеш меморија – рачунарски медијум за складиштење података

Фолдер – фасцикла која се користи за бољу организацију података на рачунару

Фонт – облик слова

Функција – део програма који обавља одређени задатак

Х

Хакер – особа која ствара изван стандардних техничких лимита, нелегално улази у туђе сигурносне и личне оперативне системе и користи туђе информације

Хардвер – делови рачунара који се могу додирнути

Ц

Црви – врста злонамерних програма

Ш

Штампач – уређај помоћу којег се подаци исписују с рачунара на папир

Литература

Драгољуб Васић, Драган Маринчић, Миодраг Стојановић: ИНФОРМАТИКА И РАЧУНАРСТВО за пети разред основне школе, Завод за уџбенике, Београд, 2012.

Драгољуб Васић, Драган Маринчић, Миодраг Стојановић: ИНФОРМАТИКА И РАЧУНАРСТВО за шести разред основне школе, Завод за уџбенике, Београд, 2011.

Драгољуб Васић, Драган Маринчић, Миодраг Стојановић: ИНФОРМАТИКА И РАЧУНАРСТВО за седми разред основне школе, Завод за уџбенике, Београд, 2011 .

Зоран Урошевић, Рачунарске мреже и комуникације, Завод за уџбенике, Београд 2012.

https://groups.csail.mit.edu/ana/Publications/PubPDFs/An-Insiders-Guide-to-the-Internet.pdf

https://support.office.com/sr-latn-rs/word

https://www.ruby-lang.org/en/documentation/ruby-from-other-languages/

Марина Лакчевић
Јасмина Алексић

ИНФОРМАТИКА И РАЧУНАРСТВО
За 6. разред основне школе
Прво издање, 2019. година

Издавач
Завод за уџбенике
Београд, Обилићев венац 5
www.zavod.co.yu

Ликовни уредник
Аида Спасић

Дизајн и компјутерска обрада
Горан Скакић

Корице
Аида Спасић

Графички уредник
Борис Поповић

Коректура

Завод за уџбенике
Обим: 15½ штампарских табака
Формат: 20,5 × 26,5 cm