UNIVERZA V MARIBORU

FAKULTETA ZA NARAVOSLOVJE IN MATEMATIKO

Oddelek za matematiko in računalništvo

Krista Rizman Žalik

Uporabniška programska oprema in programiranje

Maribor, 2021

Predgovor

V tem gradivu je predstavljena uporabniška programska oprema, primeri njene uporabe in dograditve s programiranjem. Gradivo je namenjeno predvsem študentom prve bolonjske stopnje študijskega programa izobraževalno računalništvo na Fakulteti za naravoslovje in matematiko pri predmetu Uporabniška programska oprema v izobraževanju. Ker so se predavanja v študijskem letu 2021 izvajala na daljavo, je bil primarni cilj pri pripravi tega gradiva študentom zagotoviti vir za izgradnjo razumevanja in znanja uporabe, učinkovite dograditve in prilagoditve uporabniške programske opreme, uporabne v izobraževalnem računalništvu. Zato upam, da bo delo študentom omogočalo tudi kvaliteten študij.
Cilj pripravljenega gradiva je predstaviti koncepte uporabniške programske opreme in predvsem učinkovito uporabljati možnosti dograditve uporabniške programske opreme pri izobraževalnem računalništvu. V vsakem poglavju so k opisu uporabniške programske opreme dodani tudi primeri in dograditve. V pomoč pri zbranem gradivu so tudi obstoječi učbeniki, zbirke vaj in spletni viri.

Kazalo vsebine

Kazalo slik

1 Uvod

Programska oprema (angl. software) je zbirka navodil, ki računalniku povedo, kako naj deluje [1]. Strojna oprema (angl. hardware) izvaja navodila programske opreme in zajema opremo, iz katere je sistem zgrajen.

Programska oprema se deli na sistemsko in uporabniško programsko opremo, programsko opremo za programiranje ter zlonamerno programsko opremo [1].
Sistemska programska oprema upravlja strojno opremo in zagotovi osnovne funkcionalnosti, ki jih za pravilno delovanje potrebujejo uporabniki ali uporabniška programska oprema. Sistemsko programsko opremo sestavljajo operacijski sistem, skupaj s krmilniki naprav in servisnimi programi, ki upravlja računalniške vire [1].
Programska oprema za programiranje vsebuje nabor orodij za pomoč razvijalcem pri pisanju programov. Tovrstna orodja so urejevalniki besedil, prevajalniki, povezovalniki, nalagalniki, razhroščevalniki in tolmači [1]. Razvojna okolja omogočajo celoten razvoj programske opreme. Programska oprema za programiranje razvijalcem omogoča ustvarjanje, odpravljanje napak in vzdrževanje aplikacij.
Zlonamerna programska oprema je razvita z namenom, da bi izvedla eno ali več škodljivih aktivnosti: poškodovala računalnike, ukradla gesla ali podatke, poškodovala ali motila drugo programsko opremo [1].

Uporabniško programsko opremo sestavlja skupek programov, ki nam omogoča napredno uporabo računalnika za različne naloge. V izobraževalnem računalništvu potrebujemo urejevalnike besedil, uporabniško programsko opremo za delo s števili, s preglednicami, uporabniško programsko opremo za izdelavo risb in slik in za delo z zbirkami podatkov. Potrebujemo programe za učenje programiranja kot so Alice, Blockly, Scratch in druge. Spletni brskalniki kot so Opera, Google Chrome in drugi omogočajo uporabo spletnih programov.

Računalnikarji želimo uporabniško programsko opremo prilagoditi in dopolniti ter izkoristiti možnosti za avtomatizacijo pogostih opravil. Tako v Wordu ali Excelu pišemo makroje. Za besedila z enačbami velikokrat uporabimo logične oblikovalnike besedil kot je na primer LaTeX. Spletne vsebine opišemo z uporabo jezika HTML in z opisom stilov v CSS ali pa uporabimo enega izmed množice vizualnih urejevalnikov, ki nam generirajo kodo v HTML in CSS na podlagi vizualnega opisa vsebine.

Uporabniška programska oprema se nenehno posodablja. Prihajajo vedno nove verzije z novimi funkcionalnostmi in včasih tudi novim izgledom. To gradivo ne opisuje zgodovine razvoja uporabniške programske opreme, čeprav je največkrat zelo zanimiva, predvsem njeni začetki.
Cilj tega gradi tudi ni natančno spoznavanje posamezne verzije uporabniške programske opreme ampak spoznavanje bistvenih značilnosti. Opisane so značilnosti posamezne vrste programske opreme, dodani so najpogostejši ukazi in izdelani preprosti zgledi in programčki z rezultati, ki povečajo razumevanje.

1.1 Struktura gradiva

Gradivo je sestavljeno iz šestih poglavij. Vsako poglavje opiše eno vrsto programske opreme. Gradiva ni potrebno brati poglavje za poglavjem. Torej si lahko sami določimo vrstni red poglavij ali pa preberemo samo tisto, ki nas trenutno zanima.

Uvodnemu poglavju sledi poglavje, ki opisuje pravno zaščito programske opreme in vrste ter značilnosti programske opreme glede na licenco.
Tretjo poglavje opiše osnovne koncepte oblikovalnikov besedil. Dodan je primer makra v Wordu. Namen makrov je avtomatizacija pogostih opravil. V nadaljevanju poglavja je opisan logični oblikovalnik LaTeX in izdelan primer dokumenta v LaTeXu, ki vključuje pogoste ukaze uporabljenje za oblikovanje besedil s slikami, tabelami in kazali. Opisano je tudi oblikovanje besedil na spletu s HTML, CSS in JavaSript.
V četrtem poglavju spoznamo programe za delo s števili. Opisani so koncepti in princip izračunavanja preglednic. Opisan je spletni program Wolfram Alpha in nekaj ukazov jezika Wolfram Language, v katerega se prevedejo matematična vprašanja in jih potem izračuna Wolfram Mathematica . Wolfram Alpha razume angleški jezik, s katerim lahko izrazimo vsa vprašanja, med drugimi tudi matematična vprašanja za simbolično in numerično računanje. Predstavljen je program GNU Octave, ki spada med prosto programje za numerično računanje s funkcionalnostmi in programiranjem, kompatibilnim z lastniškim programom MATLAB.
V petem poglavju opišemo delo s podatki v relacijskih podatkovnih bazah, predstavimo pa tudi NoSQL podatkovno bazo MongoDB.
V zadnjem. šestem poglavju spoznamo značilnosti programov za učenje programiranja Blockly in Alice.

2 Pravna zaščita programske opreme

Osnovna delitev programske opreme je delitev na licenco, ki opiše način uporabe, spreminjanja in razširjanja programske opreme. Programsko opremo lahko razdelimo na lastniško, prosto in odprtokodno programje [1].

Prosto programje (angl. free software) so računalniški programi, ki jih lahko uporabljamo za katerikoli namen, razmnožujemo, razširjamo, spreminjamo in izboljšujemo [2]. Prosto ali svobodno programje omogoča naslednje štiri stopnje svobode: svobodna uporaba za katerikoli namen, svobodna preučitev, svobodno razširjanje in svobodno razširjanje spremenjene verzije programa. Za svobodno preučitev in nadgradnjo programa je potrebna izvorna koda. Svobodno programje daje svobodo uporabnikom in možnost za nadaljnji razvoj. . Najbolj znana licenca, ki označuje prosto programje, je GNU GPL (angl. General Public Licence) [3]. V okviru GNU GPL licence lahko programsko kodo spreminjamo in jo nadalje distribuiramo, vendar mora spremenjena koda ohraniti licenco GNU GPL. To pomeni, da dopolnjene kode ne smemo prodajati ali kako drugače omejevati njene uporabe, lastne spremembe pa moramo tudi javno objaviti. Še vedno se lahko prodajajo mediji, na katerih je prosto programje ali storitve, povezane s tem. Primeri programske opreme zaščitene z licenco GNU GPL so Blender, GIMP, Inkscape, Linux, Open Office in drugi.

Odprtokodna programska oprema (angl. open source software) (OSS) je programska oprema, za katero je izvorna koda dostopna pod zaščitno licenco [4]. Cilj je drugačen kot pri prostem programju, gre za razvojno metodologijo: dati ljudem možnost, da pregledajo in popravijo ter dopolnijo programe, kar ustvari kvalitetnejšo programsko kodo. To je iniciativa poslovnega sveta. Odprta koda se lahko povezuje z lastniško kodo. Odprokodno programsko opremo lahko dopolnimo in prodajamo. Torej ne zahteva distribucije identičnih ali spremenjenih kopij pod isto licenco, kot je določeno pri prostem programju. Tako je možno vgraditi kodo odprtega programja v program prostega programja, obratno ni možno. Licence programske opreme, ki jih je potrdila organizacija Open Source Initiative so: Apache License 2.0 (Apache-2.0), Eclipse Public License 2.0 (EPL-2.0), Mozilla Public License 2.0 (MPL-2.0) in druge [5].

Lastniško programje omejuje uporabo, spreminjanje in razširjanje programov. Lahko je zastonj ali plačljivo in je zaščiteno s komercialno licenco. Ta določa namen in obseg uporabe: število uporabnikov in računalnikov na licenco, cena nove licence ali nadgradnje obstoječe ter morebitne izjeme za namene izobraževanja.

Brezplačno programsko opremo (angl. freeware) [6] lahko prenesemo z interneta in jo uporabimo brezplačno. Vendar spreminjanje tovrstnih programov ni dovoljeno. Prav tako ne smemo zaračunavati pristojbine za distribucijo brezplačne programske opreme. Med najbolj uporabljano brezplačno aplikacijsko programsko opremo sodijo Adobe Acrobat Reader, Google Chrome in Skype.
Tabela 1 podaja primerjavo med prostim, odprtim in brezplačnim programjem.
Poskusna različica (angl. trial version) je na voljo za brezplačno namestitev na računalnik za določeno časovno obdobje, največkrat en mesec. Poskusne različice

(angl. shareware) so programska oprema za skupno rabo, ki jo je potrebno po določenem času kupiti. Takšna programska oprema je na primer WinZip.

Sankcije za nelegalno uporabo programske opreme so podrobneje definirane v Kazenskem zakoniku RS in v Zakonu o avtorski in sorodnih pravicah.

Vrsta programske opreme	PROSTO PROGRAJE	ODPRTO PROGRAMJE	BREZPLAČNO PROGRAMJE
angl.	Free software	Open-source software	Freeware
Opis	Prosto v smislu svobode uporabe, dopolnjevanja, razširjanja in ne v smislu cene, ni vedno zastonj.	Prost dostop do izvorne kode, dovoljeno razširjanje in dopolnjevanje.	Zastonj programska oprema brez izvorne kode.
Licence	GNU General Public License	Apache License 2.0 (Apache-2.0), Eclipse z Eclipse Public License 2.0 (EPL-2.0), Mozilla Public License 2.0 (MPL-2.0)	Lastniška npr: Google Chrome Copyright 2021 Google LLC. All rights reserved.
Ozadje	Socialno gibanje	Razvojna metodologija	Tržni cilji
Pravila	4 stopnje svobode, https://www.gnu.org/philosophy/free-sw.html	Iniciativa odprtega programja https://opensource.org/OSD	Zastonj in plačljive so bolj funkcionalne verzije
Primeri	Blender GIMP Inkscape Linux Open Office	Apache HTTP strežnik Eclipse Mozilla Firefox z	Adobe Acrobat Reader Google Chrome Skype

Tabela 1: Različne lastnosti proste, odprte in brezplačne programske opreme.

3 Oblikovalniki besedil

Urejevalniki besedil omogočajo ustvarjanje, popravljanje in tiskanje besedil. Najbolj razširjen urejevalnik besedil je Notepad ++ [9], ki sodi v prosto programje z licenco GNU GPL. Urejevalniki besedil omogočajo tvorbo novega dokumenta, odpiranje obstoječega dokumenta, shranjevanje dokumenta v različnih formatih in tiskanje.

Oblikovalniki besedil pa poleg zgornjih aktivnosti omogočajo tudi oblikovanje znakov, določanje razmika med znaki in vnos besedilnih učinkov, oblikovanje odstavkov, iskanje in zamenjavo delov besedila, vstavljanje in urejanje tabel, vstavljanje slik ter vstavljanje in oblikovanje enačb. Najpogostejši oblikovalnik besedil je Microsoft Word [7], ki je del paketa Microsoft Office. Po funkcionalnosti mu je podoben Open Office Writer, ki je del prostega programja Open Office z licenco GNU GPL [8].

Programi za namizno založništvo omogočajo naprednejše grafično oblikovanje besedil z veliko grafike in tudi kvalitetnejše barvno tiskanje. Najpogosteje uporabljen je Adobe InDesign [10]. Med drugim je namenjen za ustvarjanje plakatov, brošur, časopisov in knjig.

Programsko opremo za oblikovanje besedil delimo na:

1. Vizualne oblikovalnike, ki sproti prikazujejo rezultate oblikovanja besedil. Delujejo po principu »kar vidiš, to dobiš« (angl. What You See Is What You Get - WYSIWYG).
2. Logične oblikovalnike, ki rezultatov oblikovanja besedil ne prikazujejo sproti. Obliko besedila definiramo z ukazi ali značkami. Primer logičnih oblikovalnikov so urejevalniki za LaTeX.
3. Spletne oblikovalnike, ki omogočajo tudi skupinsko oblikovanje. Najpogosteje je uporabljen spletni vizualni oblikovalnik Google Dokumenti (Google Docs).

3.1 Makroji v Wordu

V programu Microsoft Word lahko avtomatiziramo pogosta opravila tako, da poženemo ustvarjene makroje z ukazi v programskem jeziku Visual Basic for Aplications [11]. Z izbiro zavihka Pogled (angl. View) najdemo izbiro makroji, kjer lahko izberemo možnosti posnemi ali uredi makro. V okolju urejevalnika makrojev najdemo v menuju pomoč tudi pomoč za Visual Basic for Aplications.

Makroji so računalniški programi, ki avtomatizirajo določeno opravilo v Wordu. Računalniški program je zaporedje stavkov, ki napravi pove, kaj narediti.

3.1.1 Visual Basic for Aplications

V programih, ki sodijo v paket Microsoft Office, lahko programiramo s programskim jezikom Visual Basic for Aplications.

Nekaj najpogostejših ukazov:

```
' komentar,

Selection.Font.Color = wdColorRed       'rdeča barva črk
Selection.Font.Color = RGB(255,0,0)     'rdeča barva črk
Selection.Font.Bold = wdToggle          'krepko
Selection.Font.Name = "Aharoni"         'družina pisav
Selection.Font.Size = 12                'velikost pisave

Selection.ParagraphFormat.Alignment = wdAlignParagraphJustify
'nov odstavek

Selection.InsertAfter("besedilo")               'pisanje v dokument
Selection.InsertAfter(spremenljivka)

' pisanje v dokument besedilo in vrednosti spremenljivke
Selection.TypeText ("besedilo" & spremenljivka)

'deklariranje spremenljivke
dim spremenljivka as string
spremenljivka = "besedilo"
a=5

'deklaracija  celoštevilčne spremenljivke  leto:
Dim letoo As Integer


' izpiše besedilo v Word dokument:
 Selection.TypeText Text:= stevilo  & " * "

  ' vrinemo prazno vrstico
   Selection.TypeParagraph

' '& pretvori število v string pri izpisu!!!!:
        Selection.TypeText ("Maribor," & Day(d) & "." & Month(d) & "." &
Year(d))

' ČITANJE spremenljivk: niza znakov ime  in stevila stevilo
ime = InputBox("Zagnali ste program WORD! Kdo ste?")
stevilo = InputBox(”Starost?”)

'IZPIS spremenljivk  MsgBox      ; znak   +    združi 2 niza znakov:
MsgBox (“Pozdravljen”+ ime)
```

```
'&  zdruzi nize znakov in števila  in pretvori  število v  niz znakov:
MsgBox ("Pozdravljen "& ime & " in " & stevilo  & "let!")

'Izbira:
If  stevilo1 > stevilo2 Then
     MsgBox ("Stevilo1 je večje od stevila2");
ElseIf   stevilo1 = stevilo2 Then
    MsgBox ("Stevilo2  je večje ali enako kot stevilo1");
End if;

'Operatorji primerjanja: > , < , >=,  <=,  <>, =
'Logični operatorji: Not,  And,  Or,   Xor

'Ponavljanje:
For stevilo = 1 To 3
  MsgBox ("Stevilo:" & stevilo);
Next stevilo;

While  pogoj
    stavki
Wend
```

3.1.2 Makro za izpis naključnih števil

Napišimo makro, ki generira in izpiše želeno število naključnih števil med 1 in 100.

```
Sub NaključnoŠtevilo()
'Makro NaključnoŠtevilo prebere število naključnih števil, ki jih želimo
'generirati.
'Makro generira zahtevano število naključnih števil med 1 in 100 in jih
'izpiše.

ŠteviloNaključnihŠtevil = InputBox("Koliko naključnih števil (med 1 in 100)
želite generirati: ")
Selection.TypeText Text:=ŠteviloNaključnihŠtevil + " naključnih števil "

For st = 1 To ŠteviloNaključnihŠtevil
    Selection.TypeParagraph
    Selection.Font.Size = 12
    Dim NaključnoŠtevilo
    Randomize ' Inicializira naključni generator
    NaključnoŠtevilo = Int((100 * Rnd) + 1)
    Selection.TypeText Text:=" Naključno število = " & NaključnoŠtevilo
Next st

End Sub
```

3.1.3 Makro za izpis datuma

Makro, ki na mesto svetlobne značke vpiše trenutni datum in izpiše pozdrav ter število delovnih dni do konca tedna:

```
Sub Datum()
'
' Makro Datum izpiše informacije o trenutnem dnevu v okno za sporočila.
' V Word dokument izpiše datum.
Dim dnevi As Variant
dnevi  =  Array("nedelja",  "ponedeljek",  "torek",  "sreda",  "četrtek",
"petek", "sobota")

Dim današnjiDatum As Date
današnjiDatum = Now 'trenutni datom

ime = InputBox("Zagnali ste makro. Vnesite ime:")
sporočilo = " Lep pozdrav " + ime + " ! "
sporočilo1 = " Veliko uspeha pri delu s programom Word. "

danVTednu = Weekday(današnjiDatum)
sporočilo = sporočilo + " Danes je " + dnevi(danVTednu - 1) + ". "

If danVTednu = 6 Then
sporočilo = sporočilo + " Danes je petek in je zadnji delovni dan ! "
ElseIf (danVTednu = 1 Or danVTednu = 7) Then
sporočilo = sporočilo + " Danes ni delovni dan! "
Else
sporočilo = sporočilo + " Je še" + CStr(7 - danVTednu - 1) + "delovnih
dni!" 'CStr pretvori vrednost integer v string
End If

MsgBox (sporočilo & vbCrLf & sporočilo1) 'vbCrL naredi novo vrstico
```

```
' izpis datuma v dokument
Selection.TypeText    ("Maribor,"    &    Day(današnjiDatum)    &    "."    &
Month(današnjiDatum) & "." & Year(današnjiDatum))
End Sub
```

3.2 Oblikovanje besedil z LaTeX

TeX je jezik za pisanje dokumentov, ki ga je ustvaril ameriški programer D. E. Knuth [12]. LaTeX je en izmed dialektov jezika TeX, ki je namenjen za pisanje člankov in knjig. Nudi podporo za povezovanje in samodejno številčenje odstavkov, poglavij in enačb. LaTeX je program, ki prečita vhodno datoteko in pripravi datoteko z oblikovanim besedilom v formatu DVI, ki ga lahko pretvorimo v PostScript ali format PDF. Vhodna datoteka vsebuje ukaze LaTeXa.

Pri obdelavi vhodne datoteke mora LaTeX vedeti, kateri standard postavitve (angl. layout) uporabiti. Prvi ukaz v LaTeX dokumentu je ukaz
\documentclass[opcijski parametri]{razred dokumenta}.
Na koncu ukaza napišemo razred dokumenta, ki določa vrsto dokumenta, ki je lahko članek, knjiga ali pismo. Opcijski parametri omogočajo prilagoditev razreda dokumenta in so ločeni z vejicami.
Primer prve vrstice LaTeX dokumenta, ki določa, da bo dokument članek z osnovno velikostjo pisave 12 točk in postavitvijo primerno za obojestransko tiskanje na papir A4:
\documentclass[11pt, a4paper, twoside]{article}.
Za tem ukazom in drugimi opcijskimi ukazi, sledi ukaz: \begin{document}. Območje med \documentclass{...} in \begin{document} se imenuje preambula in vsebuje ukaze, ki vplivajo na celoten dokument. V preamoli navedemo pakete, ki jih bomo uporabili z ukazom \usepackage, ki mu sledi ime paketa.
Na koncu vhodne datoteke moramo dokument zaključiti z ukazom \end{document}.

Najpogostejše ukaze LaTeXa pojasnjuje naslednji preprost primer dokumenta v LaTeXu, ki ga zapišemo v datoteko primerLatex.tex.

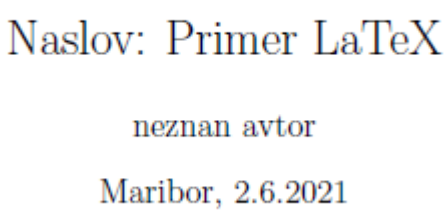
```
\documentclass [12pt] {article}              % nastavimo poljubno velikost
%fonta črk in tip dokumenta članek

\usepackage [slovene] {babel}                % podpora za slovenščino
\usepackage [cp1250] {inputenc}              % podpora za šumnike
\pagenumbering {arabic}                      % številčenje strani
\usepackage [pdftex] {graphicx}              % vključimo paket za grafiko

\renewcommand{\figurename}{Slika}            % preddefiniramo ukaz

\newtheorem{dokaz}{Dokaz}[section]          % Nov ukaz dokaz, ki bo opisan
% kot Dokaz, ki mu sledi zaporedna številka znotraj poglavja

\author{neznan avtor}
\title{Naslov: Primer LaTeX}
\date{Maribor, 2.6.2021}

\begin{document}
\maketitle                                          % izpiše  naslov

\newpage                                        %nova stran

\section{PRVO POGLAVJE}           % prvo poglavje
Besedilo
\begin{dokaz}                    % začetek leme v razdelku ena
To je prvi dokaz v 1 poglavju.
```

```
\end{dokaz}                        %konec leme v razdelku ena

\section{DRUGO POGLAVJE}      %drugo poglavje
\begin{dokaz}                   %začetek leme v razdelku dva
To je prvi dokaz v drugem poglavju.
\end{dokaz}                     % konec leme v razdelku dva
\begin{dokaz}                    % začetek druge leme v razdelku dva
To je drug dokaz v drugem poglavju.
\end{dokaz}                      %konec druge leme v razdelku dva

\section{TRETJE POGLAVJE}            % tretje poglavje
To \textbf{je} \textbf{besedilo} .
Vir:[\cite{Novak}]   %ukaz za citiranje vira
Besedilo .... [\cite{Novak2}]

Enačbe v besedilu pišemo z dolarjem $ y=\sum_{I=1}^{10}
\frac{\frac{1}{3*i}}{x_{i}}    $in  jih  lahko  oblikujemo  sz  spletnimi
urejevalniki, kot je na spletnem naslovu:
 http://latex.codecogs.com/

\begin{equation}
       y=\sum_{I=1}^{10}
\frac{\frac{1}{3*i}}{x_{i}}
\end{equation}

\subsection{Vsebina}      %podpoglavje
Besedilo....
Spodnja slika  \ref{fig-slika} opisuje .....

\begin{figure}[h]
       \centering
\includegraphics[scale=0.2]{slika.jpg}
       \caption{Slika opisuje}
       \label{fig-slika}
\end{figure}

Sledi tabela. LaTeX kodo za tabelo lahko generiramo s pomočjo spletnega
urejevalnika: https://www.tablesgenerator.com/
Vsebino opisujeta tabeli  \ref{prva-tabela} in \ref{druga-tabela}.
\newpage
\begin{table} [htbp]
\centering
\begin{tabular}{|clr|}\hline
ena & dva & tri \\ \cline{2-2}
1& 2 & 3 \\ \hline
\end{tabular}
\caption{Prva tabela}
\label{prva-tabela}
\end{table}

\begin{table}[htbp]
\centering
\begin{tabular} {cc}
a & 175 \\
b & 60 \\
cr & 110 \\
\end{tabular}
\caption {Druga tabela}
\label{druga-tabela}
\end{table}

\tableofcontents   %izpiše kazalo
\listoffigures  %izpis kazala slik
\listoftables    % kazalo tabel

\begin{thebibliography}{99}   %literatura, reference
\bibitem{Novak}
```

```
Novak, Naslov, pp102-300 , 2014.
\bibitem{Novak2}
 Novak, Naslov,  2018.

\end{thebibliography}

\end{document}
```

Ustvarjen document je potrebno prevesti v DVI, PDF ali PS format.

Rezultat zgornjega LaTeX dokumenta je datotekaprimerLatex.pdf s tremi stranmi prikazani na slikah 1, 2 in 3.



Slika 1: Prva stran dokumenta primerLatex.pdf, ki smo ga izdelali v prejšnjem primeru.

1 PRVO POGLAVJE

Besedilo

Dokaz 1.1 *To je prvi dokaz v 1 poglavju.*

2 DRUGO POGLAVJE

Dokaz 2.1 *To je prvi dokaz v drugem poglavju.*

Dokaz 2.2 *To je drug dokaz v drugem poglavju.*

3 TRETJE POGLAVJE

To **je besedilo** . Vir:[[1]] Besedilo .... [[2]]

Enačbe v besedilu pišemo z dolarjem $y = \sum_{I=1}^{10} \frac{\frac{1}{3*i}}{x_i}$in jih lahko oblikujemo sz spletnimi urejevalniki, kot je na spletnem naslovu: http://latex.codecogs.com/

$$y = \sum_{I=1}^{10} \frac{\frac{1}{3*i}}{x_i} \tag{1}$$

3.1 Vsebina

Besedilo.... Spodnja slika 1 opisuje .....

Slika 1: Slika opisuje

Sledi tabela. Za tabelo laho uporabi spletni urejevalnik:https://www.tablesgenerator.com/ Vsebino opisujet tabeli 1 in 2.

2

Slika 2: Druga stran dokumenta primerLatex.pdf, ki smo ga izdelali v prejšnjem primeru.

ena	dva	tri
1	2	3

Tabela 1: Prva tabela

a	175
b	60
cr	110

Tabela 2: Druga tabela

Kazalo

Slike

Tabele

Literatura

[1] Novak, Naslov, pp102-300 , 2014.

[2] Novak, Naslov, 2018.

3

Slika 3: Tretja stran izdelanega dokumenta v Latex (primerLatex.pdf).

3.3 HTML

Timothy John Berners-Lee je leta 1989, zaposlen v jedrskem pospeševalniku CERN, zasnoval svetovni splet, izumil protokol HTTP (angl. HyperText Transfer Protocol) in postavil prvi spletni strežnik [13]. S pomočjo sodelavcev je vzpostavil prvo HTTP povezavo med odjemalcem in strežnikom preko interneta, zaradi česar velja za izumitelja svetovnega spleta. Prva HTML stran iz avgusta 1991 je še vedno dosegljiva na naslovu:
http://info.cern.ch/hypertext/WWW/TheProject.html.

Za oblikovanje besedil na spletu uporabimo jezik za označevanje nadbesedila HTML (angl. HyperText Markup Language) [14]. Poleg HTML, ki opisuje oblikovne in vsebinske značilnosti, pri izdelavi spletnih strani praviloma uporabljamo še CSS za opis izgleda in JavaScript za opis funkcionalnosti.
Za opis strukture in izgleda besedila v HTML uporabljamo značke. Značke praviloma označujejo začetek in konec izgleda ali strukture. Kot primer znački <header> in </header> označujeta začetek in konec glave dokumenta.
Pogosto uporabljene značke so: <title>, <body>, <header>, <footer>, <article>, <section>, <p>, , , <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>, <ol>, <li> in druge.

Spletna stran je dokument, ki je lahko prikazan v oknu spletnega brskalnika ali kot besedilo v urejevalniku besedil (npr. Notepad ++) ali vizualnem urejevalniku HTML. Obstaja veliko prosto dostopnih urejevalnikov za vizualno urejanje HTML dokumentov. Primer spletnega vizualnega urejevalnika je na naslovu: https://html-online.com/editor/.

Poglejmo si primer HTML dokumenta in njegov prikaz na sliki 4.

```
<head>             <!-- začetek glave z meta podatki o spletni strani -->
<title>Page Title</title>                  <!-- naslov spletni strani -->
</head>             <!-- konec glave in začetek telesa spletne strani -->
<body>

<h1>Naslov</h1>     <!-- Naslovi -->
<h2>Podnaslov</h2>
<h3>Podnaslov</h3>
<p>Odstavek.</p>
<p>Odstavek.</p>

<!-- Vključitev slike. Oznaka <img> se uporablja za vključitev slike v
     spletno  stran HTML.
     src - Določa pot do slike in ime slike
     alt - Določa nadomestno besedilo za sliko, če slike iz nekega
     razloga ni mogoče prikazati -->
<img src="slika.jpg" alt="Ni slike ! ">

<!-- hiperpovezava -->
<a href="https://www.w3schools.com">Hiperpovezava na tečaj</a>

<h4>Neurejen seznam</h4>
<ul>                                <!-- Neoštevilčen seznam -->
  <li>a</li>
  <li>b</li>
  <li>c</li>     </ul>
```

```
<h4>Urejen seznam</h4>
<ol>                                    <!-- Oštevilčen seznam -->
  <li>a</li>
  <li>b</li>
  <li>c</li>
</ol>

Tabela:
<table>
  <tr>
    <th>ime kolone1</th>
    <th>ime kolone2</th>
  </tr>
  <tr>
    <td>a</td>
    <td>b</td>
  </tr>
  <tr>
    <td>c</td>
    <td>d</td>
  </tr>
</table>

</body>
</html>
```

Naslov

Podnaslov

Podnaslov

Odstavek.

Odstavek.

Ni slike ! Hiperpovezava na tečaj

Neurejen seznam

- a
- b
- c

Urejen seznam

1. a
2. b
3. c

Tabela:

ime kolone1	ime kolone2
a	b
c	d

Slika 4: Prikaz HTML dokumenta. Ker spletni pregledovalnik ni našel datoteke slika.jpg, je izpisal besedilo, ki je navedeno v alt delu značke <img>: »Ni slike !«.

3.4 CSS

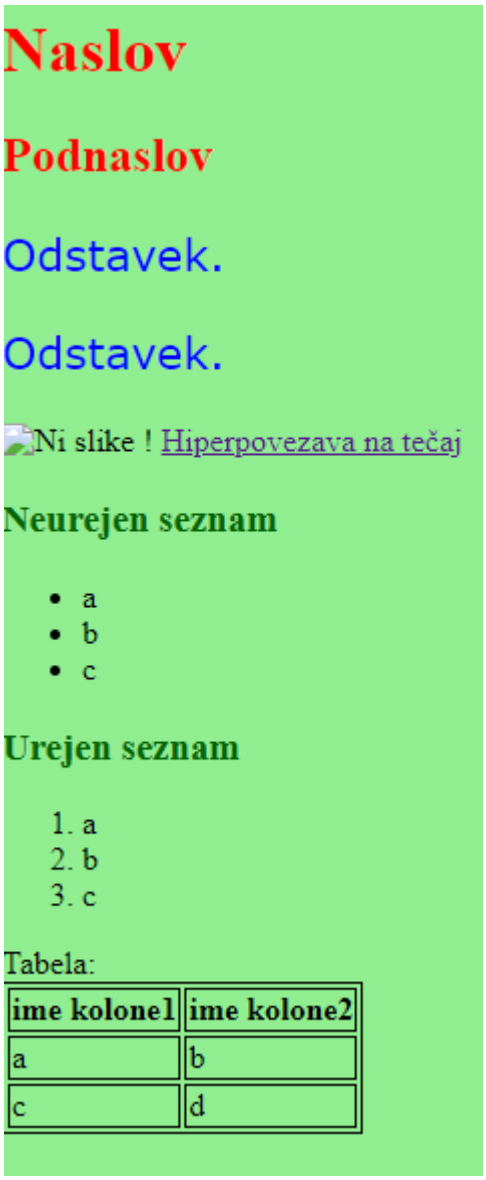
Kaskadne stilske podloge (angl. Cascading Style Sheets) ali krajše CSS [15] opisujejo izgled HTML elementov. CSS omogoča opis izgleda spletnih strani v obliki preprostega slogovnega jezika. Določamo lahko barve, velikosti, odmike, poravnave, obrobe, lego na zaslonu, prikaz ob izbiri z miško in vrsto drugih atributov. Stile opišemo v obliki:
selektor {lastnost: vrednost}

Selektorji izberejo elemente, ki jih želimo oblikovati. Primeri selektorjev so body, h1, h2, h3, p.
CSS lahko dodamo v glavo HTML dokumenta ali v svojo datoteko, ki jo vključimo z dodajanjem vrstice: <link rel="stylesheet" href="mojiStili.css">.

Primer stilov v datoteki mojiStili.css:

```
<style>
body {
  background-color: lightgreen;}
h1,h2{
  color:red;
  text-align: left;}
p {
  font-family: verdana;
  color: blue;
  font-size: 22px;}
 h3 {
  color:darkgreen;}
table, th, td {
  border: 1px solid black;}
</style>
```

Uporaba stilov za HTML dokument, izdelan v prejšnjem poglavju, da prikaz na sliki 5.



Slika 5: Izgled preprostega besedila, oblikovanega s HTML (v prejšnjem poglavju in prikazanega na sliki 4) in CSS datoteko mojiStili.css. Slika manjka namenoma.

3.5 JavaScript

JavaScript je objektni skriptni programski jezik [16]. Vsi današnji spletni pregledovalniki vsebujejo namenski stroj (interpreter) za izvajanje kode JavaScript. Jezik JavaScript je nastal leta 1995 v podjetju Netscape. Številne lastnosti in strukture so podobne programskemu jeziku Java, čeprav je bil razvit neodvisno od nje.

JavaScript je programski jezik, ki naredi spletne strani dinamične in interaktivne. Spletni pregledovalniki berejo, intepretirajo in izvedejo skripte v JavaScriptu. Z Node.js mnogi izvajajo JavaScript na strežniku. Od nastanka JavaScripta v letu 1995 je veliko spletne kode na v JavaScriptu.

Datoteko z JavaScript kodo vključimo v HTML dokument z ukazom:

```
<script type = "text/javascript" src = "imeDatoteke.js"></script>
```

Skripte, za katere ne želimo, da so enostavno vidne znotraj izvorne koda datoteke HTML, damo v svojo datoteko in jo vključimo v HTML.
Lahko pa JavaScript programe pišemo kar znotraj HTML dokumenta. Funkcije, zapisane v JavaScriptu, ki jih kličemo iz več mest, damo v glavo HTML dokumenta. JavaScript kodo s klici funkcij pa imamo v telesu dokumenta HTML.

Funkcije lahko kličemo iz elementov HTML, kot na primer:

```
<a href="somewhere.html" onclick="mojaFunkcija()">Click me</a>.

<button onclick= "mojaFunkcija()" >Računaj</button>
```

Knjižnice in ogrodja JavaScripta vsebujejo nabore vnaprej napisane kode, pripravljene za uporabo. Nekatera priljubljena ogrodja (angl. framework) JavaScripta so Angular, React, Vue in Node.js. Največkrat uporabljena knjižnica je jQuery. Ogrodje zagotavlja temelje in strukturo, medtem ko knjižnica omogoča dodajanje že pripravljenih komponent. Tudi ogrodja vsebujejo knjižnice. Knjižnice so manjše od ogrodij in se običajno uporabljajo za bolj specifične namene.

Za spremenljivke in operatorje velja podobno kot v programskih jezikih C++ in Java. JavaScript ima kontrolne stavke if, if...else in switch ter zanke for in while enake oblike kot C++ in Java. Sledijo najpogostejši JavaSript stavki.

```
var priimek = "Janez"      // (spremenljivka z besedilom tipa string
var ure = 37;              // numerična vrednost - integer
var opravil = false;       // logična vrednost - Boolean

if (pogoj)
{
   stavek;
   stavek;
}
else
{
   stavek;
```

```
   stavek;
}

for (začetnaVrednost; končnaVrednost; inkrement)
{
   stavki;
}

while (pogoj)
{
   stavki;
}

function ime(parameter_1, parameter_2)
{
   stavki;
   return(spremenljivka);
}
```

3.5.1 Izračun faktoriele

Izdelajmo JavaScript program za izračun faktoriele.

```
<html>
<head>
<script type="text/javascript">

function faktoriela(obrazec)
{
   var n = obrazec.število.value;
   // iz obrazca prečita vrednost polju z imenom število

   var rezultat = 1;
   for (var i = 1; i<= n; i++)
   {
      rezultat = rezultat * i;
   }
   prikaz.innerHTML= n +"!"  +  " = " + rezultat
   // Prikaz vsebuje izpis faktoriele števila

   return false
}
</script>

</head>
<body>

<h1> Faktoriela</h1>

<form method="post" onsubmit="return faktoriela(this)" >
<p><strong>Vnesite </strong> število za izračun faktoriele:

<input type="text" name="število" value="1" size="2" />
</p>
<p>   </p>
<input type="submit" value="Izračunaj" />
<input type="reset" value="Inicializiraj" />
</form>

<p id="prikaz"> </p>
</body>
</html>
```

3.5.2 Java Script in DOM

V zadnjih nekaj letih se je objektna struktura spletnih brskalnikov standardizirala, kar olajša programiranje. DOM (angl. Document Object Model) je W3C (World Wide Web Consortium) [17] standard, ki opisuje hierarhijo objektov, kar programom in skriptam omogoča dinamičen dostop in posodobitev vsebine, strukture, prikaza in sloga dokumentov.

W3C DOM standard obravnava tri dele:
- Core DOM standard je model za vse tipe dokumentov.
- XML DOM standard je model za XML dokumente.
- HTML DOM standard je za HTML dokumente z drevesno strukturo vozlišč.

Objektni model dokumenta (DOM) je predstavitev podatkov objektov, ki sestavljajo strukturo in vsebino spletnega dokumenta. Takoj po besedi <script> lahko začnemo uporabljati API (Application Programming Interface) za dokumente ali okenske predmete za manipulacijo samega dokumenta ali katerega koli od različnih elementov na spletni strani (potomci dokumenta) [18].

DOM je objektni model za delo s strukturo in slogi strani HTML. Predstavlja stran kot jo vidi brskalnik in omogoča razvijalcu, da jo spremeni z JavaScriptom [19]. Objekt document in njegove metode so natančno opisane [20,21]. Uporabimo metodo getElementById objekta document za doseganje elementov po IDju za zamenjavo besedila Test s pozdravom.

```
<p id="demo">Test<\p>
document.getElementById("demo").innerHTML = "Pozdravljeni!";
```

3.5.3.1 Primer dodajanja novega HTML elementa

Spodnji primer spremeni vsebino naslova, ustvari nov naslov in ga doda v telo HTML strani.

```
<html>
  <head>
    <script>
       //  Tvorimo nov HTML element, ki ga ustvarimo z
       //  document.createElement.
       //  Nato ga dodamo v telo HTML dokumenta:

       function dodajNaslov()
       {
         // Tvori naslov in ga doda na konec strani.
         Const novh1 = document.createElement(»h1«);
         const naslov = document.createTextNode(»Nov naslov«);
         novh1.appendChild(naslov);
         document.body.appendChild(novh1);
      }

    </script>
  </head>

<body>

<!--    Za    spreminjanje    prikazanih    HTML    elementov    uporabimo
document.getElementById("id_elementa").innerHTML. -->
```

```
<h1 id="naslov">Naslov:  JavaScript lahko spremeni vsebino.</h1>
<button type="button"
onclick='document.getElementById("naslov").innerHTML =
 " Nov naslov!"'>Spremeni naslov!</button>

<!--  Dokumentu lahko dodamo nov HTML element, ki ga  ustvarimo z
document.createElement in nato ga dodamo v telo HTML dokumenta. -->

<h1 id="naslov">Naslov:  JavaScript doda HTML element v spletno
stran.</h1>
<button type="button" onclick='dodajNaslov()'>Dodaj naslov!</button>

  </body>
</html>
```

3.5.3 JavaScript in objekti

JavaScript je objektni jezik. Poljuben objekt ustvarimo s konstruktorsko funkcijo, ki objekt inicializira.

V resničnem življenju imajo vsi študenti podobne lastnosti. Za vsakega študenta lahko določimo ime, priimek, naslov, e-mail, leto vpisa in trenutni letnik vpisa. V programu vse naštete lastnosti združimo v objekt. Poleg lastnosti lahko objektu dodamo še metode, ki posodobijo vrednost kakšne izmed spremenljivk objekta ali vrnejo vrednost na podlagi vrednosti spremenljivk. Za vsakega študenta lahko naredimo svoj objekt. Vrednosti lastnosti se med študenti razlikujejo, metode pa imajo vsi študenti iste.
Vrednosti lastnosti so zapisane v obliki ime: vrednost in so ločene z vejicami.

Sledi primer programa, kjer ustvarimo objekt študent z lastnostmi ime, priimek, naslov, letoVpisa in letnik, ter metodo celotnoIme, ki vrne ime in priimek študenta. Celotno ime študenta prikažemo na spletni strani.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript objekti</h2>

<p id="podatek"></p>
<p id="podatek1"></p>

<script>

// Ustvari objekt:
 const študent = {
  ime: "Janezek",
  priimek: "Novak",
  naslov : "Glavni trg 1 Maribor" ,
  letoVpisa: 2020,
  letnik: 2,
  celotnoIme: function() {
    return this.ime + " " + this.priimek;
  }};


// Izpišimo nekaj podatkov objekta. Izpišemo vrednost lastnosti:
document.getElementById("podatek").innerHTML =
```

```
študent.ime + " je " + študent.priimek + "ov.";


// Izpišimo vrednost, ki nam jo vrne metoda celotnoIme() objekta študent.
document.getElementById("podatek1").innerHTML =
študent.celotnoIme()+".";
</script>

</body>
</html>
```

4 Delo s števili

Preglednica je podatkovna datoteka, sestavljena iz vrstic in stolpcev. Uporablja se za razvrščanje podatkov in omogoča uporabniku preprosto in intuitivno upravljanje predvsem številčnih podatkov. Popularna programa za ustvarjanje in urejanje preglednic sta Excel [22], ki je del Microsoft Office, in Calc [23], ki je del Open Office.

Vsaka celica ima svoj naslov, ki je sestavljen iz naslova stolpca (ena ali več velikih črk: A, B, …, AA, AB, …) in naslova vrstice (število). Naslov celice v prvem stolpcu in prvi vrstici je tako A1. V celice lahko vnašamo različne tipe podatkov: števila, besedila, datume, itd. Poleg tega lahko vrednost celic izračunamo z uporabo matematičnih formul ali funkcij, ki so vgrajene v program za urejanje preglednic. Vsebina celic, v katerih se vrednost izračuna, se začne z enačajem. V funkcije ali matematične formule lahko vstavljamo podatke iz drugih celic, to storimo z vključitvijo naslova celice, na katero se sklicujemo. Če torej želimo, da je vrednost celice B1 enaka dvakratniku vrednosti celice A1, v celico B1 vpišemo »=A1*2«.

Za preglednice je značilno podatkovno vodeno računanje. Ob spremembi podatka v eni celici, se posodobijo vrednosti vseh celic, ki se izračunajo iz te celice. Kot primer, če spremenimo celico A1 se bo spremenila tudi celica B1, ki vsebuje enačbo »=A1*2«.

Privzeto se celice v preglednici naslavljajo relativno. To pomeni, da če zgornjo enačbo iz celice B1 premaknemo za en stolpec desno, bo celica C1 vsebovala spremenjeno enačbo »=B1*2«. Podobno se zgodi, če enačbo iz celice B1 premaknemo za vrsto navzdol. V tem primeru bo celica B2 vsebovala »=A2*2«.

Poznamo tudi absolutno naslavljanje, ki ob premiku enačbe ohrani isti stolpec, če je v enačbi pred naslovom stolpca znak $ (»=$A1*2«) in isto vrsto, če je v enačbi znak $ pred naslovom vrstice (»=A$1*2«). Absolutno naslavljanje celice lahko hkrati velja za vrstico in stolpec (»=A1*2«).

4.1 Excel macro

Podobno kot v Wordu, lahko tudi v programu Microsoft Office Excel napišemo makro. Excel nudi številne funkcije, ki jih lahko uporabimo za izračun posamezne celice. Poleg vgrajenih, lahko zapišemo tudi svoje funkcije z uporabo jezika Visual Basic for Application, ki smo ga uporabili tudi za makroje v Wordu.

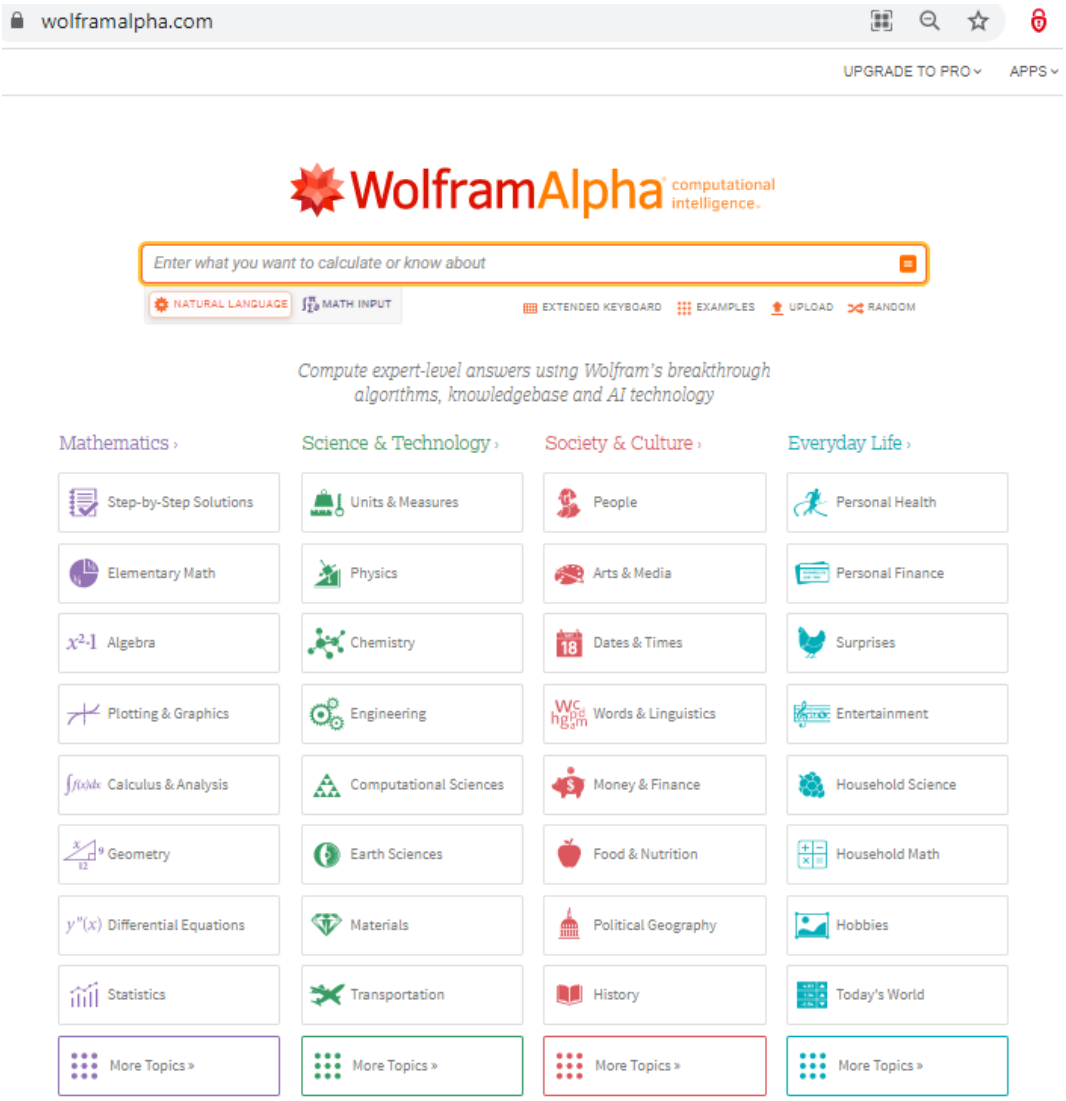
Primer funkcije v Excelu, ki v besedilu s številom vrne samo besedilo ali samo število.

```
' VBA koda tvori funkcijo VrniDel, ki vrne del besedila v Celici.
' Če je VrniŠtevilo true, funkcija vrne iz samo število, sicer pa vrne samo
besedilo

Function VrniDel(Celica As String, VrniŠtevilo As Boolean)

Dim DolžinaStringa As Integer
DolžinaStringa = Len(Celica)
'Funkcija Len vrne dolžino besedila, ki je v Celici

' Pregledamo znak za znakom besedila v Celici
' Mid vrne določeno število znakov iz besedilnega niza z začetkom pri
' navedenem položaju in na osnovi navedenega števila znakov.

For številkaZnaka = 1 To DolžinaStringa
  If IsNumeric(Mid(Celica, številkaZnaka, 1)) Then
    število = število & Mid(Celica, številkaZnaka, 1)
  Else
    Besedilo = Besedilo & Mid(Celica, številkaZnaka, 1)
  End If
Next številkaZnaka

If VrniŠtevilo Then
  VrniDel = število
Else
  VrniDel = Besedilo
End If

End Function

Funkcijo pokličemo, da v celico vpišemo na primer:
= VrniDel(E4;true)    vrne število v besedilu

= VrniDel(E4;false)   vrne besedilo brez števila
```

4.2 Wolfram Alpha

Wolfram Alpha [24] je brezplačen spletni servis, ki ga je leta 2009 razvilo podjetje Wolfram Research, ki je znano tudi po orodju za simbolično in numerično računanje Mathematica. Poleg brezplačne, ima Wolfram Alpha tudi plačljivo različica PRO, ki nudi tudi razlago rezultatov korak za korakom.

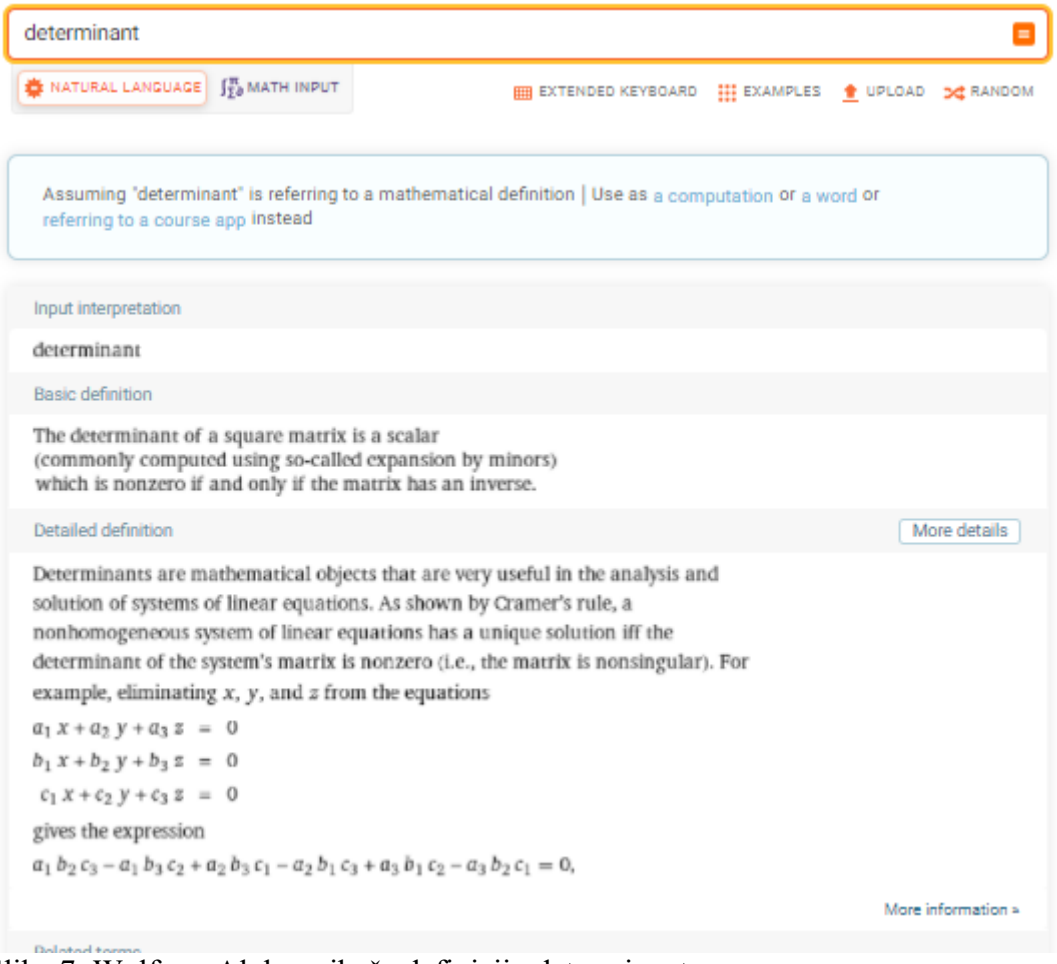
Wolfram Alpha obravnava ukaze ali vprašanja v obliki ključnih besed, besednih zvez, matematičnih enačb ali stavkov v angleškem jeziku. Na vprašanja poda odgovore z uporabo Wolframovih algoritmov, baze znanja, Mathematice za matematična vprašanja in tehnologije umetne inteligence. Zna uporabiti shranjeno znanje za različna področja, ki so prikazana na sliki 6. Wolfram Alpha omogoča simbolno in numerično računanje.



Slika 6: Prva stran spletnega servisa Wolfram Alpha.

Wolfram Alpha sodi med inteligentne aplikacije Spleta 3.0. Splet je kot svetovni informacijski prostor od leta 1969 dosegel velik napredek. Vedno bolj uporablja algoritme umetne inteligence in postaja masiven splet inteligentnih interakcij.
Začetke spleta so zaznamovali HTTP protokol in jezik HTML. Proti koncu obdobja Spleta 1.0, ki je trajalo od leta 1990 do 2000 so se razvili JavaScript, Java in PHP. Sledilo je desetletje spleta 2.0 do leta 2010, ko so se razvijale spletne aplikacij za razširjeno uporabo, izmenjavo informacij in sodelovanje na svetovnem spletu. Uporabniki sodelujejo pri ustvarjanju vsebin.
Splet 3.0, ki je trajal od leta 2010 do 2020, je semantični splet, imenovan tudi pametni splet. Podatki so distribuirani, razdrobljeni, vseprisotni na spletu. To je splet za ljudi in tudi pametne stroje (umetno inteligenco), ki znajo te podatke analizirati in razumeti.
Prihaja obdobje Spleta 4.0, ki mora nuditi inteligentne aplikacije in predvideti upravljanje in inteligentno uporabo vsega razpoložljivega znanja v omrežju z medijsko konvergenco in bolj intuitivnimi iskalnimi mehanizmi [25].

Wolfram Alpha je spletna inteligentna aplikacija, ki posreduje odgovore na vprašanja zastavljena v naravnem jeziku (angleškem) ali z enačbami. Lahko reši linearne enačbe, poišče lastne vrednosti matrike, pa tudi odvaja in integrira, nariše funkcije in še veliko več. Vsebuje definicije (slika 7) in zna razložiti rešitev korak za korakom (slika 8).



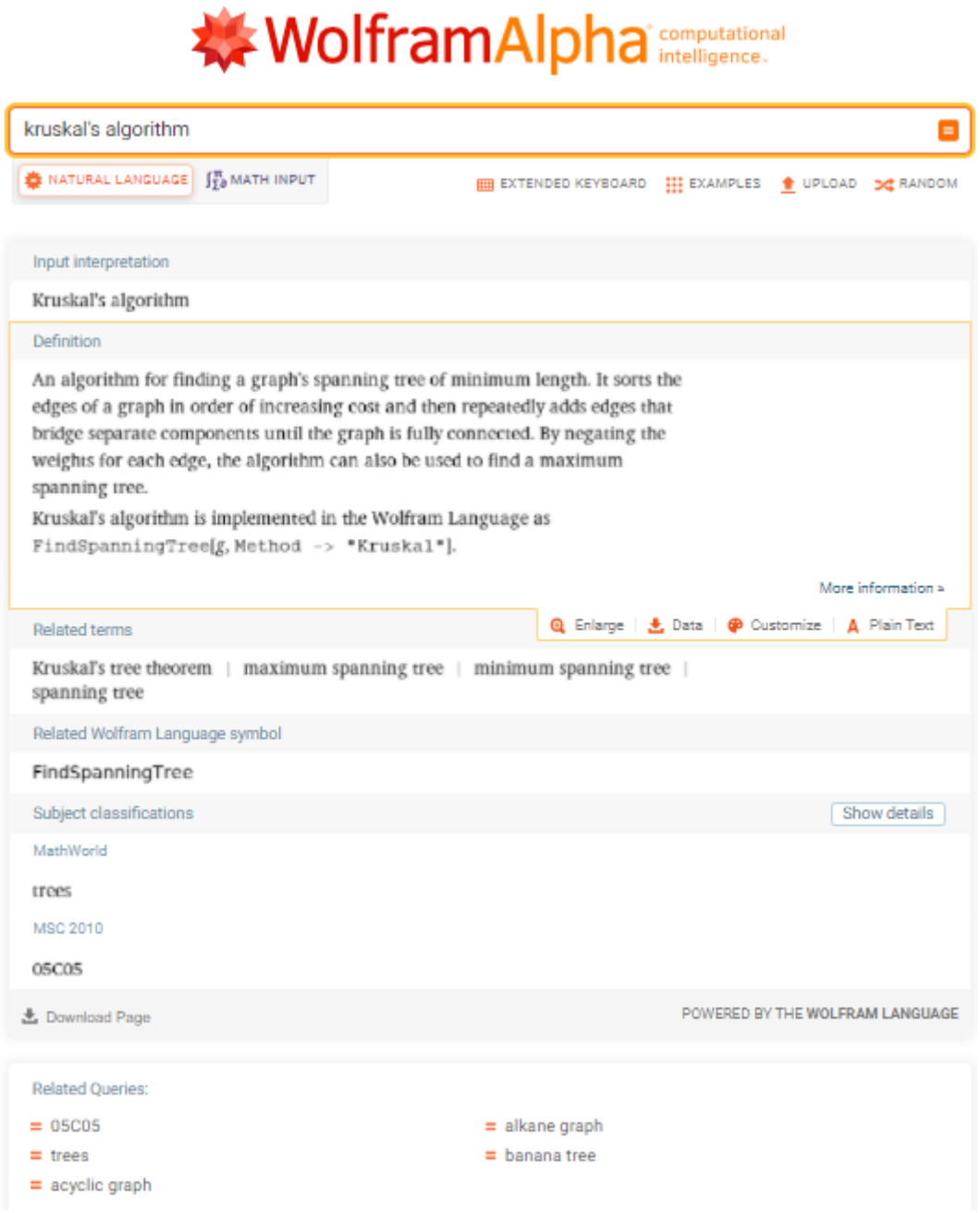
Slika 7: Wolfram Alpha prikaže definicijo determinante.

Slika 8: Razlaga rešitev korak za korakom v Wolfram Alpha.

Nekaj primerov povpraševanja v naravnem jeziku:

- Highest mountain Slovenia.
- Prime numbers.
- How many basketballs fit in Boing 747?
- Kruskal's algorithm (slika 9).

Ob vsaki poizvedbi nam ponudi še povezana vprašanja. V desnem kotu okna je vedno prikazano, da servis izvede Wolfram Language.



Slika 9: Odgovor servisa Wolfram Alpha na vprašanje o Kruskalovem algoritmu.

4.3 Jezik Wolfram

Jezik Wolfram (angl. Wolfram Language) [26] je jezik programskega paketa Mathematica in je v ozadju Wolfram Alphe. Pri poizvedbah se v Wolfram Alphi izpišejo tudi ukazi ali funkcije jezika Wolfram. Tudi na izpisu zgoraj, na sliki 9, je navedena funkcija za vpeto drevo v jeziku Wolfram Language: FindSpanningTree. Jezik Wolfram poudarja simbolno računanje, funkcionalno programiranje in programiranje, ki temelji na pravilih in lahko uporablja poljubno strukturo in podatke.
Sledi nekaj ukazov v jeziku Wolfram. Argumenti funkcij so označeni z oglatimi oklepaji.

```
Sin[Pi]                          funkcija, ki izračuna sinus(360).
Integrate[f,{x,xmin,xmax}]       vrne določen integral.

For zanka:
For[i = 1; t = x, i^2 < 10, i++, t = t^2 + i; Print[t]]

If stavek vrne t, če je pogoj pravilen, sicer vrne f.
Ifpogoj,t,f]

Ta if stavek vrne t, če je pogoj izpolnjen, f, če ni izpolnjen in u če
pogoj ni pravilen in ni nepravilen:
If[pogoj,t,f,u]

Funkcija:
abs[x_] := If[x < 0, -x, x]

Ujemanje vzorcev uporablja najpogostejši konstrukt presledek _ ali polje
presledkov__ , ki se ujema s katerikoli izrazom.

Funkcija MatchQ preveri, če izraz ustreza vzorcu.
Vsi spodnja preverjanja se ujemajo:
MatchQ[a,_]
MatchQ[a[1],_]
MatchQ[{ 10,20},_List]

Ujemanje vzorcev lahko uporabimo v povpraševanju:
Cases[S, izlet[_, _, _, "Paris"] ]
Zgornje povpraševanje lahko vrne več rezultatov:
izlet[12, oktober, letalo, Paris], izlet[13, november, avtobus, Paris]


Algoritmi v obliki pravil:
Operator /; pomeni pogojno izvajanje, tako da se pravilo izvede za y>z
Operator ponavljaj //. Služi za ponavljajoče uporabo pravila, dokler ni
več nobenih sprememb.

praviloUredi := {s1___,s2_,s3_,s4___} /; s2>s3 -> {s1,s3,s2,s4}

 { 8, 9, 1, 5, 3} //. praviloUredi
```

4.4 GNU Octave

GNU Octave [27] je programski jezik, ki je v veliki meri združljiv z jezikom MATLAB. Octave nudi zmogljivo matematično sintakso, skladno z vgrajenimi orodji za risanje in vizualizacijo v 2D in 3D. Octave omogoča numerično računanje, delo z matrikami, nudi funkcije za linearno algebro, delo s polinomi, risanje, lahko pa tudi sprogramiramo svoje funkcije

Octave je prosta programska oprema z licenco GNU GPL. Octave deluje v različnih operacijskih sistemih: Microsoft Windows, GNU/Linux, macOS in BSD, obstaja pa tudi več spletnih servisov Octave [28].

Poglejmo si nekaj pogostejših ukazov.

```
Linearna algebra vektorji:

b = [1; 1; 1]   # vektor
A = [ 1 1 1;    # matrika
      1 3 1;
      1 1 1]
x = A \ b       # Reši sistem Ax = b

Polinom je predstavljen z vektorjem koeficientov.

P=[1 0  -3]   predstavlja polinom x^2 - 3

Vrednost polinoma v točki T (x0,y0)

y0=polyval(P,x0)

Enostavno je najti polinom druge stopnje, ki se ujema s točkami vektorja
X in Y.

X=[1 0 2] in Y=[0 3  4]

P2=polyfit(X,Y,2);

Izris sinusne funkcije:
t = 0:0.01:2*pi;
y = sin (t);
plot (t, y);
title ("en val sinusne funkcije");
```

Ukaze lahko izvajamo neposredno v komandni vrstici okolja Octave ali pa jih zapišemo v tekstovno datoteko, shranimo v obliki ime_programa.m ter v okolju Octave datoteko poženemo kot skripto.

4.4.1 Primer: izračun faktoriele

Rekurzivna funkcija za izračun faktoriele:

```
function f = faktoriela(n)
%Rekurzivna definicija:
%n!  =  n * (n-1) * (n-2) * … * 3 * 2 * 1= n * (n-1)!
if (n>1)
   f = n * faktoriela(n-1);
else
```

```
   f = 1;
endif
endfunction
```

Nerekurzivna funkcija za izračun faktoriele:

```
function f = faktoriela1(n)
%Nerekurzivna definicija:
%n! = n * (n-1) * (n-2) * … * 3 * 2 * 1
f=1;
for stevec=2:n
   f= f*stevec
end
endfunction

%Funkcijo lahko pokličemo večkrat
faktoriela(3)
faktoriela(6)
faktoriela(9)
```

5 Zbirke podatkov

Podatkovne baze [29] so danes nepogrešljive povsod, kjer je potrebno:
- na urejen način shranjevati podatke,
- zagotoviti celovitost, varnost, konsistentnost, in podatkovno neodvisnost podatkov,
- omogočiti učinkovit dostop do podatkov z nadzorom nad sočasnim dostopom do podatkov in
- zagotoviti učinkovito administracijo in enostavno obnavljanje po nesrečah.

Podatkovna baza je po priporočilih ANSI/SPARC sestavljena iz treh nivojev:
1. zunanji (uporabniški) nivo – predstavlja uporabniški pogled na podatkovno bazo,
2. konceptualni (logični) nivo – opiše strukturo shranjenih podatkov, atribute podatkov in povezave med podatki,
3. notranji (fizični) nivo – določi kje in kako so podatki shranjeni v podatkovni bazi in kakšen je dostop do podatkov.

Podatkovne baze delimo glede na podatkovnem modelu na katerem temeljijo na:
- relacijske, ki so dandanes najpogosteje uporabljane,
- objektne, ki se niso komercialno uveljavile in
- NoSQL (angl. Not only SQL) podatkovne baze.

5.1 Relacijske podatkovne baze

Relacijske podatkovne baze temeljijo na relacijskem podatkovnem modelu, ki ga je predlagal E. F. Codd leta 1970 [30]. Relacijske podatkovne baze shranjujejo podatke v tabelah s stolpci in vrsticami.

Zagotavljajo lastnosti ACID (angl. Atomicity, Consistency, Isolation; Durability):
- *A* označuje atomarnost, kar pomeni, da se izvede samo celotna transakcija ali pa se transakcija ne izvede.
- *C* označuje konsistentnost pred in po transakciji.
- *I* označuje izolacijo; Več transakcij se izvede istočasno brez medsebojnega vplivanja.
- *D* označuje vzdržljivost oz. trajnost hranjenja sprememb uspešnih transakcij.

Na spletnem portalu DB-Engines so leta 2021 poročali o najpogosteje uporabljenih relacijskih bazah [31]:
1. Oracle Database
2. MySQL
3. Microsoft SQL Server
4. PostgreSQL (prosta programska oprema)
5. IBM Db2
6. SQLite (prosta programska oprema)
7. Microsoft Access
8. MariaDB (prosta programska oprema)
9. Hive (prosta programska oprema, specializirana za podatkovna skladišča)

10. Teradata
11. Microsoft Azure SQL Database

Standardni jezik za relacijske podatkovne baze je jezik SQL. SQL je strukturirani povpraševalni jezik (angl. Standard Query Language). Določen je z ANSI (angl. American National Standards Institut) standardom iz leta 1986 in ISO (International Organization for Standardization) standardom iz leta 1987.

SQL [32,33] se uporablja za vstavljanje, iskanje, spreminjanje in brisanje zapisov v podatkovni bazi. Osnovni stavki so:
INSERT INTO - vrine nove podatke v bazo.
SELECT - izpiše podatke iz baze,
UPDATE - spremeni podatke v podatkovni bazi,
DELETE - izbriše podatke iz baze.

Orodja, kot je MySQL Workbench [34], nudijo zaslone za tvorjenje tabel, vrivanje ažuriranje in brisanje podatkov.

Stavke SQL opremimo s komentarji:

```
SELECT * FROM profesorji;   --enovrstični komentar

/* Večvrstični kmentar: Naslednja poizvedba  izbere vse kolone
in vse vrstice iz tabele študenti : */

SELECT  * FROM študenti ;
```

Poglejmo si sintakso stavkov INSERT in SELECT.

```
INSERT INTO ime_tabele (kolona1, kolona2, kolona3, ...)
VALUES (vrednost1, vrednost2, vrednost3, ...);

Primer:
INSERT INTO profesorji VALUES (12345678910, 'Novak Miha','FERI');

SELECT [distinct] seznam atributov
FROM ime tabel
WHERE pogoji
ORDER BY atributi;

FROM     -- opiše tabele.
WHERE    -- izloči vrstice, ki ne zadostujejo pogoju.
DISTINCT -- izloči ponavljajoče vrstice.
ORDER BY -- uredi po atributih naraščajoče, če sledi ASC ali pa nič in
             - padajoče če atributom sledi besedica DESC.
```

Če poizvedba zahteva podatke iz več tabel, moramo zapisati pogoje združevanja podatkov med tabelami. Izpišimo vse podatke iz dveh tabel: fakultete (id, naziv) in profesorji (emso, ime, id_fakultete)

```
SELECT  *
FROM fakultete, profesorji
WHERE profesorji.id_fakultete=fakultete.id;
```

5.2 NoSQL podatkovne baze

NoSQL podatkovne baze, za razliko od relacijskih podatkovnih baz, ne potrebujejo tabelarične sheme. Z NoSQL označujemo nerelacijske, porazdeljene, odprtokodne in horizontalno razširljive podatkovne baze [35]. NoSQL podatkovne baze ne podpirajo poizvedovalnega jezika SQL. Večina NoSQL podatkovnih baz ne uporablja stika (angl. join) več tabel. NoSQL podatkovne bazei prav tako ne zagotavljajo ACID transakcijskih lastnosti. V primerjavi z relacijskimi podatkovnimi bazami so te baze bolj prilagodljive in zagotavljajo boljšo zmogljivost.

Podatkovne baze NoSQL lahko obdelujejo tako strukturirane kot nestrukturirane podatke, učinkovite so tudi pri obdelavi tudi nestrukturiranih masovnih podatkih (angl. Big Data). Zaradi tega organizacije, kot so Facebook, Twitter, LinkedIn in Google, ki so se prve soočile z omejitvami relacijskih zbirk podatkov pri hranjenju masivnih podatkov, ogromnim obsegom transakcij in uporabnikov ter zahtevami po takojšnji razpoložljivosti storitev uporabljajo podatkovne baze NoSQL [36].

Podatkovne baze NoSQL temeljijo na različnih podatkovnih modelih. Glavne vrste podatkovnih baz NoSQL temeljijo na naslednjih modelih: dokument, ključ-vrednost, široki stolpec in graf. Zagotavljajo prilagodljive sheme in se enostavno prilagajajo velikim količinam podatkov in velikim obremenitvam uporabnikov [37].

Pogoste NoSQL podatkovne baze so [37]:
- široki stolpec: Azure Cosmos DB, Cassandra, Scylla, HBase,
- dokument: Azure Cosmos DB, Apache CouchDB,, BaseX, Clusterpoint, eXist-db, IBM Domino, MongoDB, OrientDB ,
- ključ-vrednost: Azure Cosmos DB, Apache Ignite, Berkeley DB,
- graf: Azure Cosmos DB, InfiniteGraph, Apache Giraph, Neo4J in druge.

Baze NoSQL programiramo na različne načine. Pogledali si bomo NoSQL podatkovno bazo MongoDB [38]. MongoDB je sistem za upravljanje podatkovnih baz (SUPB), ki temelji na dokumentih in je napisan v *C++*. MongoDB razvija istoimensko podjetje in ima licenco Server Side Public License (SSPL), ki je modifikacija GNU General Public License. Prva verzija podatkovne baze MongoDB je bila izdana leta 2009.

Zgradba podatkovne baze MongoDB [38]:
- Baza podatkov v MongoDB je vsebnik za zbirke dokumentov. Baza podatkov vsebuje več zbirk (angl. collection).
- Vsaka zbirka vsebuje več dokumentov.
- Dokument je sestavljen iz imen polj in vrednosti.

Pojem tabele v relacijskih bazah je ekvivalenten pojmu zbirka, pojem vrstica tabele JSON/BSON dokumentu in pojem kolone je ekvivalent JSON/BSON polju v dokumentu. JSON (JavaScript Object Notation) je zelo priljubljen standard za izmenjavo podatkov na spletu, na katerem temelji BSON (Binary JSON).

MongoDB ne uporablja stikov podatkov. Omogoča gnezdenje dokumentov (angl. embeded documents) in stike v uporabniškem programu.

Za namene učenja in testiranja lahko brezplačno preizkušamo MongoDB v oblaku, v okviru storitve MongoDB Atlas.Tam ustvarimo zbirke in dokumente.

Sintalsa stavka insert:

```
db.COLLECTION_NAME.insert({DOCUMENT})
```

Vrinemo nove dokumente:

```
>db.profesorji.insert({ime:"Mitja Novak", področje: "računalništvo"})
>db.profesorji.insert({ime:"Jure Novak",starost:55})
>db.profesorji.insert({ime:"Tim Novak", starost: 40 })
>db.profesorji.insert({ime:"Aleš Novak", spol:1})
>db.profesorji.Mulinsert({ime:"Črti Novak", spol:"moški"" })
```

V isti zbirki profesorji smo vstavili dokumente z različnimi polji: področje, starost in spol.
Izpišimo profesorje računalništva:

```
db.profesorji.find({področje: "računalništvo"}).pretty()
```

Izpis bo vseboval samo dokument:

```
({ime:"Mitja Novak", področje: "računalništvo"})
```

Izpišimo vse profesorje:

```
db.profesorji.find().pretty()
```

Izpišemo samo en dokument:

```
db.profesorjii.findOne()
```

Izpišimo profesorje, starejše od 30 let. Uporabimo operator $gt:

```
db.profesorji.find({{ starost: { $gt: 20 } })
```

Več pogojev lahko združujemo z operatorjema $or ali $and:

```
db.ime_zbirke.find({ $and: [ {pogoj}, {pogoj}, .... ] })
```

Izpišimo profesorje računalništva s priimkom Novak*:*

```
db.profesorji.find({ $and: [{ime: /.*Novak.*/ }, {smer: {$ne:
"računalništvo"}}] }).pretty()
```

MongoDB zapisu sam doda enolični identifikator id, to je polje, ki deluje kot primarni ključ.

Dokumenti so lahko tudi vgnezdeni. Vgnezden dokument se tako kot dokument začne in konča z zavitimi oklepaji.

```
{
  "_id": "5cf0029caff5056591b0ce7d",
```

```
  "priimek": "Frac ",
  "ime": "Novak",
  "naslov": {
    "ulica": "Glavni trg 1",
    "mesto": "Maribor",
    "država": "Slovenija"
  },
  "predmeti": ["Podatkovne baze", "Spletne tehnologije"]
}
```

Povpraševanje po vgnezdenem dokumentu:

```
db.profesorji.find ({ naslov:
{
    "ulica": "Glavni trg 1",
    "mesto": "Maribor",
    "država": "Slovenija"
  }
})
```

Povpraševanje po atributu vgnezdenega dokumenta:

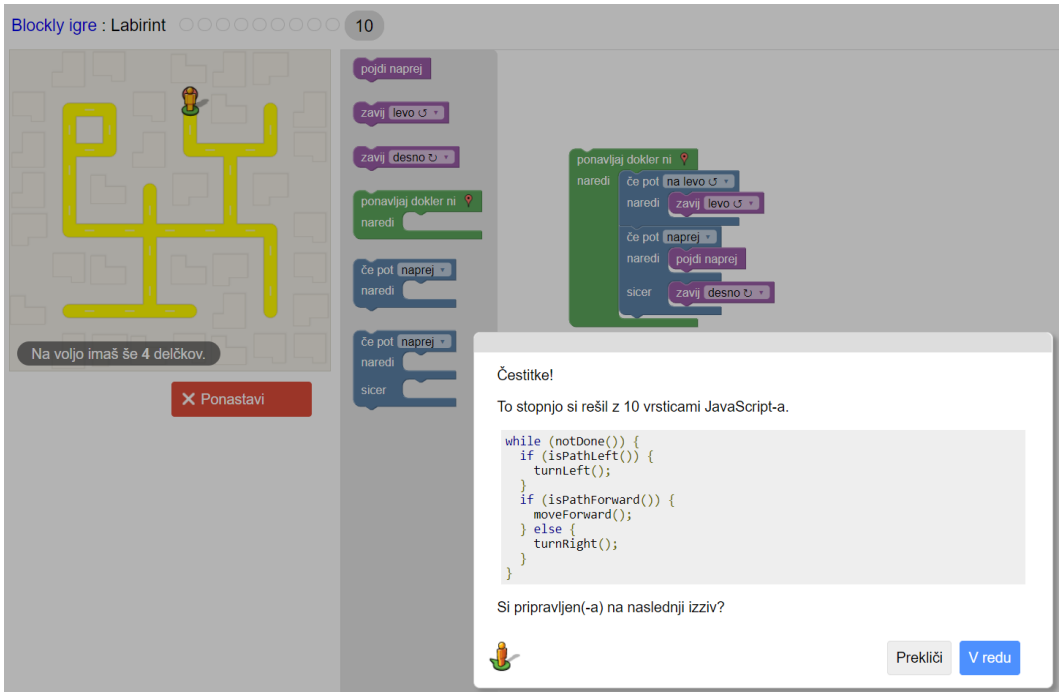
```
db.profesorji.find( { "naslov.mesto": "Maribor" } )
```

6 Programska oprema za učenje programiranja

6.1 Slikovni programski jeziki in okolja za bodoče programerje

Za začetke razvijanja računalniškega mišljenja so učinkoviti slikovni programski jeziki (angl. Visual Programming Languages). Obstaja cela množica slikovnih programskih jezikov, med najbolj znani so Alice, App Inventor, Blockly, Flowgorithm, Greenfoot, Lego Mindstorms NXT, Raptor, Scratch in drugi.

Google razvijalci so razvili niz spletnih izobraževalnih iger Blockly Games [39]. Igre so namenjene tistim brez predhodnih izkušenj v programiranju. Primer programa v Blockly Games prikazuje slika 10.



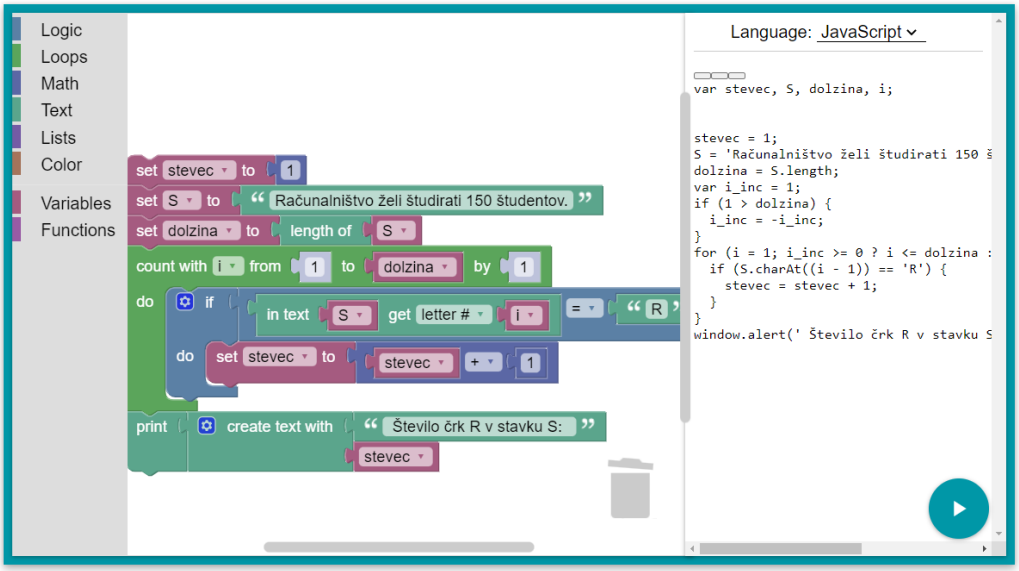
Slika 10: Prikaz izdelanega programa v Blockly Games z bloki v JavaScriptu.

Igre Blockly Games uvedejo stavke izbire in ponavljanja, spremenljivke za hranjenje vrednosti in funkcije. Funkcija je skupek programske kode, ki se lahko kliče večkrat. Vsaka funkcija ima svoje ime ter lahko sprejema in vrača vrednosti..

Izobraževalne igre uporabnika povedejo od blokovnega programiranja do pisanja tekstovnih programov v programskem jeziku JavaScript. Vse izobraževalne igre pa ob pravilnem zapisu algoritma z bloki prikažejo generiran zapis algoritma v JavaScriptu (slika 10).

Blokovno programiranje v igrah Blockly Games poteka v vizualnem programskem jeziku Blockly [40], ki so ga prav tako ustvarili razvijalci Googla, skupaj z

istoimenskim spletnim okoljem, ki omogoča izdelavo poljubnih programov. Algoritme sestavljamo iz pripravljenih, medsebojno različno skladnih blokov. Ti se delijo na kategorije logika, zanke, matematika, besedilo, seznami, barve, funkcije in spremenljivke. Blockly omogoča vizualno programiranje preproste in sintaktično pravilne kode, kar pospeši učenje programiranja. Z bloki izdelan algoritem se lahko izpiše tudi v programskih jezikih JavaScript, Pyton, PHP, Lua in Dart (slika 11).



Slika 11: Prikaz izdelane programa v Blockly v JavaScriptu

6.2 Strukturirano, proceduralno, deklarativno, dogodkovno in objektno programiranje

Pristopu k reševanju problemov, pri katerem delimo problem na vse manjše podprobleme, ki se jih reši s funkcijami, pravimo tudi strukturirano programiranje. Z uporabo funkcij in stavkov izbire (if/then/else) in ponavljanja (while and for), izboljšamo razumljivost, kakovosti in čas razvoja računalniškega programa.. K uvedbi izraza strukturirano programiranje je pripomogel Dijkstra [41] s člankom o škodljivosti GOTO stavka za razumevanje in jasno strukturo programa.

Ker zastavljen problem rešimo z zapisom zaporedja programskih stavkov, ki se izvajajo po vrsti, lahko rečemo, da smo računalniku podali postopek oz. proceduro reševanja. Zato takšno programiranje imenujemo tudi proceduralno (postopkovno) programiranje (angl. procedural programming).
Nasprotno od proceduralnega je (nepostopkovno) deklarativno programiranje, kjer samo opišemo, kakšne rezultate želimo oziroma kakšen problem moramo rešiti, ne da bi izrecno opisali postopek, kako naj se problem reši [42].
Na primer, s stavkom zapisanim v SQL (kot je na primer SELECT ime, priimek FROM študenti;) programiramo nepostopkovno, saj smo opisali kakšne podatke želimo in ne kako naj sistem za upravljane s podatkovno bazo pride do teh podatkov.

Dogodkovno programiranje (*ang. event-driven programming*) omogoča, da tok programa določajo različni dogodki, kot na primer klik z miško, pritisk tipke ali potek določenega časa. Dogodkovno programiranje je nujno za vse aplikacije z grafičnimi uporabniškimi vmesniki. Dogodkovni programi zaženejo poslušalce (*ang. listener*), ki čakajo na dogodke. Vsak poslušalec čaka na točno določene dogodke. Ko se dogodek zgodi, se izvede koda vseh poslušalcev, ki so čakali na ta dogodek [43].

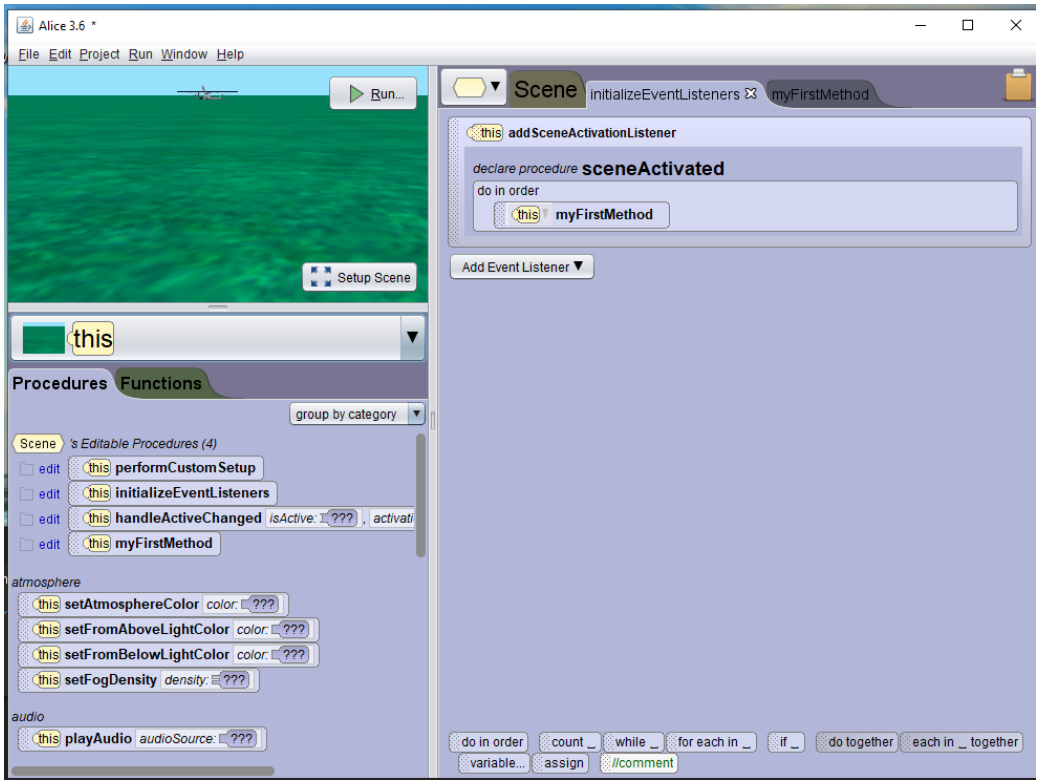
Pri objektno orientiranem programiranju (*ang. object-oriented programming – OOP*) [44], so spremenljivke in metode (funkcije in procedure), ki delujejo nad temi spremenljivkami, združene v samostojno enoto, ki jo imenujemo razred (angl. class). Za vsak razred lahko naredimo več objektov (*ang. object*). Na primer za nek razred lahko ustvarimo več objektov. Vsak ustvarjen objekt bo imel svoje vrednosti spremenljivk, vsi pa bodo imeli iste metode (funkcije ali procedure), kot jih določa razred. Vendar pa iste metode različnih objektov ne bodo vračale istih vrednosti, saj je delovanje metode praviloma odvisno od vrednosti spremenljivk v pripadajočem objektu. Poleg tega, da vrednosti spremenljivk objekta vplivajo na delovanje metod, lahko metode ob klicu tudi spremenijo vrednosti spremenljivk pripadajočega objekta. Objektno usmerjeno programiranje omogoča enostavno pisanje modularne kode. Vsak razred opisuje točno določen del realnega okolja Enostavno je pisanje kode za večkratno uporabo, saj je razrede lažje ponovno uporabiti in razširiti. Na posamezne razrede lahko gledamo kot na nove podatkovne tipe. Razrede lahko sestavimo tako, da so podatki, ki opisujejo določen objekt vedno smiselni oziroma v določenem stanju.
Koncept, ki ga najpogosteje povezujemo z objektno usmerjenim programiranjem je dedovanje (*ang. inheritance*). Dedovanje nam omogoča opis novih razredov, ki

so doplnjene različice obstoječih razredov in vsebuje vse spremenljivke in metode obstoječega razreda.

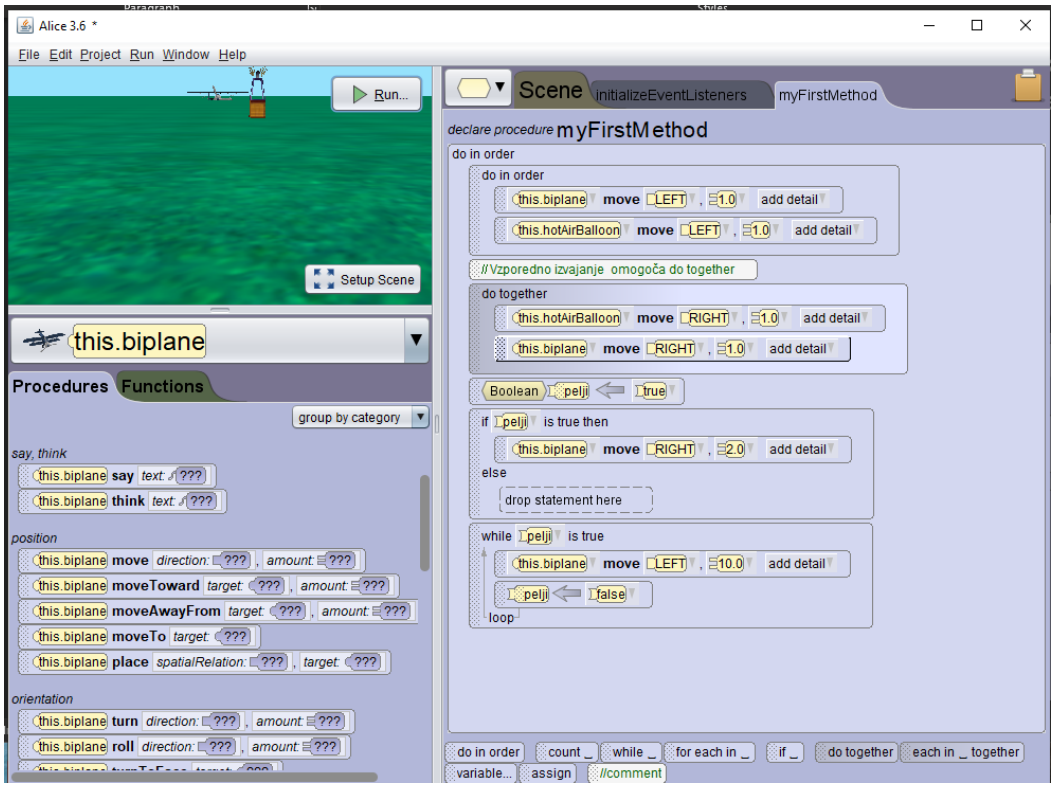
6.3 Alice

Alice [45] je vizualni programski jezik in okolje za učenje objektno-orientiranega programiranja. Vizualni jeziki omogočajo izdelavo sintaktično pravilnih programov s sestavljanjem blokov. Alice je brezplačno programsko okolje za ustvarjanje 3D svetov, ustvarjanje animacij, interaktivnih pripovedi ali programiranje preprostih iger v 3D. V ustvarjenih 3D svetovih združujemo različne objekte, ki imajo vsak svoje lastnosti. 3D okolje na preprost način prikaže, kako deluje koncept objektnega programiranja. Ponuja predpripravljene razrede in okolje za spletno vizualno programiranje. Za razliko od okolij za učenje programiranja, ki temeljijo na postavljenih nalogah, kot na primer Blockly Games, omogoča Alice učenje z ustvarjalnim raziskovanjem.

V okolju Alice najprej tvorimo objekte vnaprej pripravljenih razredov iz knjižnice razredov. Potem moramo napisati algoritem za izvedbo animacije. Ko se program začne, se prikaže scena in pokliče se prva metoda *MyFirstMethod (slika 13)*. To je metoda objekta Scene. Na sliki spodaj je v zgornjem levem kotu urejevalnik scene, pod njim je seznam objektov in v spodnjem levem delu zaslona so prikazane procedure in funkcije izbranega objekta. Na sliki 12 je izbran objekt Scene.

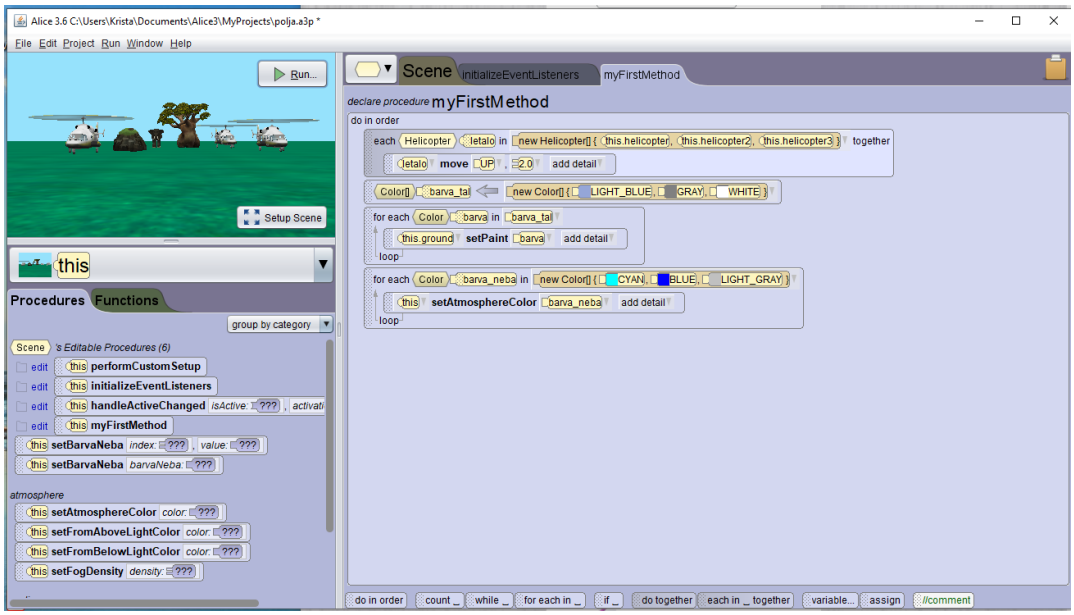


Slika 12: Okolje Alice, izbran je objekt Scene zato so prikazane procedure, ki so na voljo za objekt Scena.



Slika 13: Prva metoda, ki se izvede ob zagonu programa je MyFirstMethod.

V Alice se procedure in funkcije prikažejo, ko izberemo objekt. Procedure in funkcije, ki jih lahko izvedemo nad objekti se razlikujejo glede na izbran objekt. V spodnji kontrolni vrstici (slika 13) Alice ponuja izbirni stavek *if* in zanke *while*. Stavek *do in order* izvede stavke znotraj zanke zaporedno, medtem ko *do together* omogoča vzporedno izvajanje. Poleg tega Alice nudi stavka za delo s polji: *for each in* izvede stavke v jedru za vsak element polja zaporedno in *each in together,* ki izvede stavke v jedru za vsak element polja vzporedno (slika 14).

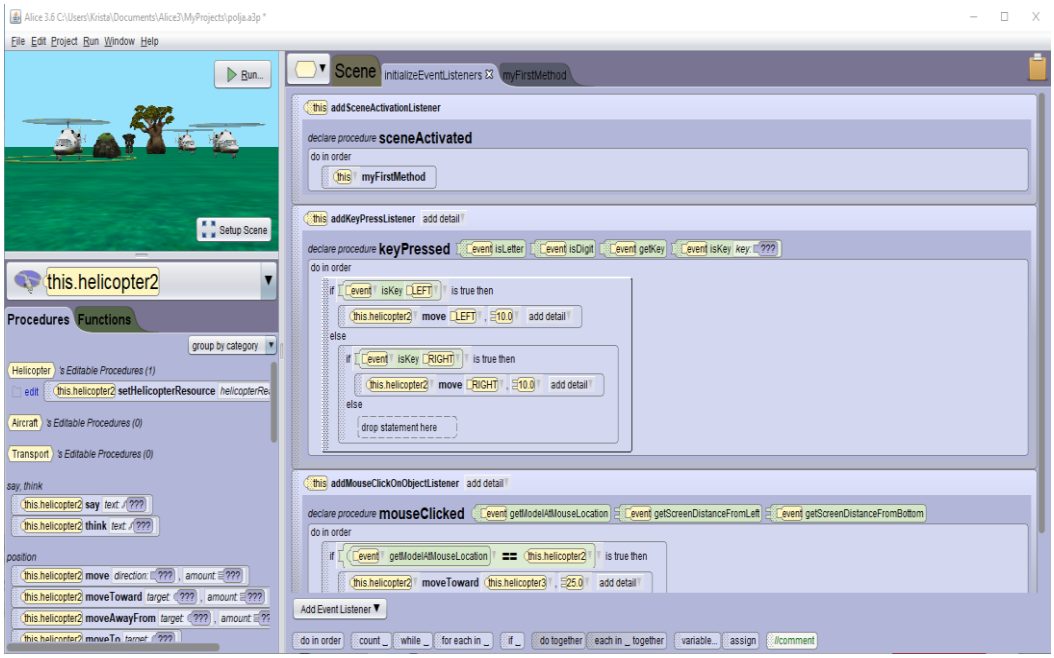


Slika 14: Prikaz uporabe ukaza *for each in* in ukaza *each in together* v okolju Alice.

Strukture lahko gnezdimo. Spreminjanje barve neba smo dali v neskončno zanko s stavkom *while true*.

Alice omogoča dogodke z izbiro določene tipke na tipkovnici, ob kliku miške in dogodke, ki se prožijo s časom trajanja animacije ali pa dogodke, ki jih proži lega ali usmerjenost objektov.
Dogodke, ko želimo izvesti akcijo ob izbiri objekta s klikom miške opišemo:

- Najprej izberemo stavek *if*.
- V *if* pogoju namesto *true* izberemo v menuju *Relational(SThing)* == in nato izberemo objekt, ki ga bomo izbrali s klikom miške.
- V blok *if* dodamo stavke, ki se bodo izvedli ob kliku objekta (slika 15).



Slika 15: Alice in opis dogodkov s keyPressedListener in MouseClickOnObjectListener

Literatura

[1] Software https://www.techopedia.com/definition/4356/software

[2] What is Free Software? https://www.gnu.org/philosophy/free-sw.html

[3] The GNU General Public License, https://www.gnu.org/licenses/#GPL

[4] The open source initiative, The open source definition https://opensource.org/OSD

[5] Open Source Licenses by Category https://opensource.org/licenses/category

[6] Freeware Definition http://www.linfo.org/freeware.html

[7] Word https://www.microsoft.com/sl-si/microsoft-365/word

[8] Open Office https://www.openoffice.org/

[9] Notepad++, https://notepad-plus-plus.org/

[10] Adobe InDesign, https://www.adobe.com/si/products/indesign.html

[11] Visual basic documentation https://docs.microsoft.com/en-us/dotnet/visual-basic/

[12] Donald E. Knuth, The TEXbook, Reading, Massachusetts: Addison-Wesley, 1984. ISBN 0-201-13448-9.

[13] Sir Timothy Berners-Lee OM, KBE, FRS, FREng, FRSA https://www.w3.org/People/Berners-Lee/Longer.html

[14] HTML Living Standard. https://html.spec.whatwg.org/multipage/

[15] Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification https://www.w3.org/TR/CSS2/

[16] David Flagavan, JavaScript The definitive guide (6 ed.). O'Reilly, 2011.

[17] What is the Document Object Model? https://www.w3.org/TR/WD-DOM/introduction.html

[18] JavaScript and HTML DOM Reference http://www.w3schools.com/jsref/default.asp

[19] JavaScript reference https://developer.mozilla.org/en/JavaScript/Reference

[20] Introduction to the DOM https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

[21] Document https://developer.mozilla.org/en-US/docs/Web/API/Document

[22] Microsoft Excel, programska oprema za preglednice.

[23] Calc. Preglednica za vse namene. https://www.microsoft.com › sl-si › microsoft-365 › excel

[24] Wolfram Alpha, https://www.wolframalpha.com/

[25] Balloni, Antonio & De Souza Bermejo, Paulo. (2010). Governance, Sociotechnical Systems and Knowledge Society: Challenges and Reflections. 109. 42-51. 10.1007/978-3-642-16402-6_5. https://www.researchgate.net/figure/As-mentioned-before-Web-30-refers-to-a-supposed-third-generation-of-Internet-based_fig2_221522851

[26] Wolfram Language https://www.wolfram.com/language/

[27] GNU Octave. https://www.gnu.org/software/octave/index

[28] Octave Online · Cloud IDE compatible with MATLAB. https://octave-online.net/

[29] Ramakrishnan, R., Gehrke, J., Database management systems-third edition. New York, McGraw-Hill 2003.

[30] Codd, E. F. (1970). A Relational Tedel of Data for Large Shared Data Banks. Communications of the ACM. 13 (6): 377–387. doi:10.1145/362384.362685I.

[31] "DB-Engines Ranking of Relational DBMS". https://db-engines.com/en/ranking/relational+dbms Doseženo 2021-03-24.

32] SQL C. J. Date with Hugh Darwen: A Guide to the SQL standard : a users guide to the standard database language SQL, 4th ed., Addison Wesley, USA 1997, ISBN 978-0-201-96426-4

[33] SQL Tuning https://sql-tuning.com/oracle-tuning-tools/

[34] MySQL Workbench https://www.mysql.com/products/workbench

[35] NoSQL http://nosql-database.org/ "NoSQL DEFINITION: Next Generation Databases mostly addressing some of the points : being non-relational, distributed, open-source and horizontally scalable".

[36] A. Keith D. Foote, Brief History of Non-Relational Databases, 2018. https://www.dataversity.net/a-brief-history-of-non-relational-databases/

[37] Strauch, Christof. "NoSQL Databases" (PDF). pp. 23–24. https://www.christof-strauch.de/nosqldbs.pdf

[38] The database for modern applications; https://www.mongodb.com/ (Get started free: https://www.mongodb.com/cloud/atlas/register). Dostopano: 2021-05-20.

[39] Blockly Games https://blockly.games

[40] Try Blockly https://developers.google.com/blockly

[41] Dijkstra, Edsger W. (March 1968). "Letters to the editor: Go to statement considered harmful" (PDF). Communications of the ACM. 11 (3): 147–148. doi:10.1145/362929.362947. ISSN 0001-0782. S2CID 17469809

[42] Lloyd, J.W., Practical Advantages of Declarative Programming, Conference Proceedings/Title of Journal: Joint Conference on Declarative Programming, 3-17, 1994.

[43] Vivek Gupta, Ethan Jackson, Shaz Qadeer and Sriram Rajamani (November 2012). "P: Safe Asynchronous Event-Driven Programming". Microsoft Research.

[44] Lewis, John; Loftus, William (2008). Java Software Solutions Foundations of Programming Design 6th ed. Pearson Education Inc. ISBN 978-0-321-53205-3., section 1.6 "Object-Oriented Programming"

[45] Alice https://www.alice.org/