

Programiranje 1

Beleške sa vežbi

Smer *Informatika*

Matematički fakultet, Beograd

Jelena Tomašević

Sadržaj

1		5
1.1	<code>L^AT_EX</code>	5
2		11
2.1	Formalni jezici i formalne gramatike	11
2.1.1	Jezici: zadaci	11
2.1.2	Gramatike i jezici: zadaci	12
3		15
3.1	Pozicioni brojni sistemi - konverzije	15
3.2	Prebacivanje iz sistema sa osnovom b u dekadni brojni sistem	15
3.3	Prebacivanje iz dekadnog brojnog sistema u sistem sa osnovom b	16
3.4	Rad sa realnim brojevima	17
3.4.1	Prevođenje realnih brojeva iz sistema sa osnovom B u dekadni brojni sistem	17
3.4.2	Prevođenje realnih brojeva iz dekadnog brojnog sistema u sistem sa osnovom B	18
3.5	Direktno prevođenje iz binarnog u heksadekadni sistem	18
3.6	Direktno prevođenje iz binarnog u oktalni sistem	19
4		21
4.1	Reprezentacija znakovnih podataka	21
4.1.1	Tekst je niz karaktera	21
4.1.2	Zapis karaktera u računaru	21
4.1.3	Skupovi karaktera	21
4.1.4	Kodiranje ostalih jezika	22
4.1.5	Osobine YUSCII koda	22
4.1.6	Kodne strane	23
4.1.7	Kodne strane kod nas	23
4.1.8	Višebajtni karakterski kodovi	23
4.1.9	Karakter, Glifovi, Fontovi	24
5		27
5.1	Meta jezici: EBNF, sintaksni dijagrami	27
5.1.1	BNF	27
5.1.2	EBNF	28
5.1.3	Sintaksni dijagrami	29
6 Programski jezik C		31
6.1	Funkcije <code>printf()</code> i <code>scanf()</code> - osnovna upotreba	31
6.2	Aritmetičke i relacijske operacije	32
6.3	Kontrola toka - <code>if</code> , <code>if-else</code> , <code>while</code> , <code>do-while</code> , <code>switch</code>	36
6.4	Funkcije - osnovni pojmovi	41

7 Programski jezik C	45
7.1 Učitavanje i ispis na izlaz - funkcije printf() i scanf()	45
7.2 Operator sizeof	47
7.3 Obrada teksta sa ulaza - getchar() i putchar()	48
7.4 Nizovi	52
7.4.1 Stringovi	55
7.4.2 Prenos niza u funkciju	56
8 Programski jezik C	57
8.1 Obrada teksta sa ulaza	57
8.2 Operator ,	58
8.3 Uslovni izrazi	59
8.4 Oblast važenja lokalnih promenljivih	59
8.5 Lenjo izračunavanje	59
8.6 Nizovi	61
8.6.1 Nizovi-prenos u funkciju	62
8.7 Break i continue	64
8.8 Prenos parametara po vrednosti	65
9 Programski jezik C	67
9.1 Ugnježdjena petlja	67
9.2 F-je za rad sa stringovima	68
10 Programski jezik C	73
10.1 Konverzije	73
10.2 Makroi	75
11 Programski jezik C	79
11.1 Bit-operatori	79
12 Programski jezik C	87
12.1 Formiranje HTML dokumenta	87
12.2 Strukture	88
12.3 Životni vek i oblast važenja promenljivih	92
13 Programski jezik C	95
13.1 Rad sa datotekama	95
14 Programski jezik C	103
14.1 Argumenti komandne linije	103
14.2 Pokazivači - osnovni pojmovi	106
14.3 Strukture - uvežbavanje	109
15	113
15.1 Zadaci sa prethodnih ispita i kolokvijuma	113

1

1.1 L^AT_EX

Za pravljenje tex dokumenata možemo koristiti bilo koji tekstualni editor. Na primer, u Linux-u, koristimo *emacs*, *kate* ili *kwite*.

Minimalni tex dokument je sledećeg oblika:

```
\documentclass{article}

\author{Ime i prezime}

\title{Naslov}

\begin{document}

\maketitle

\section{Uvod}

...

\end{document}
```

Ako smo ga sačuvali kao `rad.tex`, možemo ga prevesti naredbom `latex rad.tex` čime dobijamo DVI format. Ako želimo da dobijemo PDF format kucamo

```
dvipdf rad.dvi rad.pdf
```

Ukoliko prevođenje vršimo pod Windows operativnim sistemom pod pretpostavkom da imamo instaliran MikTeX onda u Command Prompt-u kucamo sledeće:

```
latex rad
```

Za prevođenje u ps format kucamo

```
dvips rad.dvi
```

a za prevođenje u pdf format kucamo

```
dvipdfm rad.dvi
```

Izgled dokumenta možemo videti pomoću naredbe

```
yap rad.dvi
```

ili

```
start rad.ps
```

ili

```
start rad.pdf
```

Kad napravimo osnovni `tex` dokument dalje ga formiramo po našoj želji. Ukoliko želimo da pređemo na novu stranu onda ukucamo `\newpage`. Naredbom `\section{Naziv poglavlja}` dodajemo *novo poglavlje* koje se prevođenjem automatski numerise u skladu sa svojim rednim brojem. Na sličan način dodajemo i podpoglavljia naredbom `\subsection{Naziv podpoglavlja}`.

Ako u nekom delu teksta želimo da se pozovemo na neko tvrđenje koje smo dokazali u nekom drugom poglavlju, to možemo uraditi označavanjem tog poglavlja (onog u kom se nalazi tvrđenje na koje se pozivamo) *labelom*. To radimo tako što posle naredbe početka poglavlja dodamo naredbu `\label{oznaka}` a u svakom drugom poglavlju koje se poziva na njega stavimo *referencu* naredbom `\ref{oznaka}`. *oznaka* je niska koju sami biramo i jedinstveno je dodeljena tom poglavlju.

U sledećoj tabeli su date naredbe za različite *stilove teksta*:

Boldovan tekst	<code>\textbf{tekst}</code>
<i>Italik</i>	<code>\emph{tekst}</code>
<u>Podvučen</u>	<code>\underline{tekst}</code>
<u>Nadvučen</u>	<code>\overline{tekst}</code>

Specijalne karaktere, one koji učestvuju u naredbama, ispisujemo dodavanjem `\` ispred određenog karaktera. To su `$`, `&`, `%`, `#`, `-`, `{`, `}`. Da bi ispisali `\` kucamo `\backslash`.

Da bi dobili *novi red* potrebno je da u `tex` dokumentu napravimo prazan red (odnosno da dva puta pritisnemo ENTER) ili da ukucamo `\\` ili `\newline`.

Za dobijanje *ćiriličnih slova* koristimo naredbe date u sledećoj tabeli:

ć	<code>\v c</code>
ć	<code>\' c</code>
đ	<code>\d</code>
ž	<code>\v z</code>

Sledećom tabelom su date još neke mogućnosti *formatiranja teksta*:

Tekst koji se nikad ne prelama	<code>\mbox{<i>text</i>}</code>
Uvučen tekst	Između <code>\begin{quote}</code> i <code>\end{quote}</code>
Doslovno se prepisuje na izlaz	Između <code>\begin{verbatim}</code> i <code>\end{verbatim}</code>
Matematički tekst	Između <code>\$</code> i <code>\$</code> ili između <code>\begin{math}</code> i <code>\end{math}</code>
Izdvojena matematička formula	Između <code>\begin{displaymath}</code> i <code>\end{displaymath}</code>
Tekst unutar formule	<code>\textrm{<i>text</i>}</code>
Levo poravnan tekst	Između <code>\begin{flushleft}</code> i <code>\end{flushleft}</code>
Desno poravnan tekst	Između <code>\begin{flushright}</code> i <code>\end{flushright}</code>
Centriran tekst	Između <code>\begin{center}</code> i <code>\end{center}</code>

Ako imamo neko nabrojavanje koje hoćemo da završimo sa ... kucamo `\ldots`.

Komentare kucamo iza znaka `%`. Sve u tom redu se neće videti na izlazu.

Grčko slovo Ω kucamo naredbom `\Omega`, slično i za ostala grčka slova. Ako je prvo slovo u naredbi veliko dobićemo veliko grčko slovo, inače dobijamo mala grčka slova. Malo slovo ω dobijamo naredbom `\omega`.

Možemo naznačiti na kojim mestima reči smeju biti podeljene pri prelaza iz jednog reda u drugi (da ne bi došlo do nepravilnih podela) i to radimo naredbom `\hyphenation{lista reči}`.

Pri čemu **lista reči** sadrži spisak svih reči za koje želimo da naglasimo na koji način se dele, a to radimo stavljanjem povlake (–) na mestima gde je podela dozvoljena.

Ako želimo *dužu povlaku* kucamo `--`, a ako želimo još dužu kucamo `---`. Latex ih automatski spaja u jednu dugu povlaku i dobijamo `---` odnosno `---`.

Razne *matematičke formule* se mogu otkucati korišćenjem naredbi iz sledeće tabele:

$\frac{1}{2}$	<code>\frac{1}{2}</code>	\vec{a}	<code>\vec{a}</code>
a^x	<code>a^x</code>	\log	<code>\log</code>
a_n	<code>a_n</code>	\max	<code>\max</code>
ϵ	<code>\epsilon</code>	ε	<code>\varepsilon</code>
\forall	<code>\forall</code>	\exists	<code>\exists</code>

Ako želimo da dodamo *fusnotu*¹ to radimo naredbom `\footnote{tekst fusnote}` na mestu gde želimo da nam se pojavi oznaka fusnote. Tekst fusnote će se automatski smestiti na dno strane.

Liste pravimo na više načina u zavisnosti od toga da li želimo numerisanu, nenumerisanu ili opisnu listu.

¹ovo je fusnota

Numerisanu listu dobijamo na sledeći način:

```
\begin{enumerate}
\item prva stavka      1. prva stavka
\item druga stavka     2. druga stavka
...                    ...
\end{enumerate}
```

Nenumerisanu listu dobijamo sledećom naredbom:

```
\begin{itemize}
\item prva stavka      • prvastavka
\item druga stavka     • drugastavka
\item [-] a može i crtica - a može i crtica
...                    ...
\end{itemize}
```

Opisnu listu dobijamo na sledeći način:

```
\begin{description}
\item{(a)} prva stavka      (a) prva stavka
\item{(b)} druga stavka     (b) druga stavka
...                          ...
\end{description}
```

Pri čemu umesto (a) i (b) mogu stajati proizvoljne reči.

Tabele pravimo naredbom

```
\begin{tabular}{specifikacija tabele}
```

pri čemu se navode elementi ćelija tabele sleva na desno i odozgo na dole. Prelaz iz jedne kolone u drugu se označava sa & a prelazak u novi red sa \\. Ako želimo horizontalnu liniju dobijamo je sa \hline. Kada završimo sa navođenjem svih elemenata tabele kucamo naredbu \end{tabular}. U *specifikaciji tabele* navodimo slova *r*, *l*, *c* u zavisnosti od toga da li je odgovarajuća kolona desno poravnana, levo poravnana ili centrirana. Pri tome treba da imamo isti broj slova koja specifikuju poravnanje kao i kolona. Ako želimo vertikalne linije njih navodimo u specifikaciji tabele stavljanjem znaka | između slova koja određuju poravnanje kolone ili oko njih u zavisnosti od toga kako želimo, i da li uopšte želimo, da ograničimo tabelu.

Na primer,

```
\begin{tabular}{|ccc|}
\hline
1 & 2 & 3 \\\
4 & 5 & 6 \\\
\hline
\end{tabular}
```

1	2	3
4	5	6

Da bi u dokument ubacili *sliku u EPS* formatu, moramo na početku dokumenta uključiti paket \usepackage{graphicx} i onda na mestu gde želimo da nam se pojavi slika kucamo naredbu

\includegraphics[width=w1, height=h1]{slika.eps} pri čemu je zadavanje širine i visine slike opciono.

Komanda `\tableofcontents` proizvodi *sadržaj* na mestu gde je izdata.

2

2.1 Formalni jezici i formalne gramatike

2.1.1 Jezici: zadaci

1

Zadatak 1 Dokazati da ne postoji nijedna reč x azbuke $\Sigma = \{a, b\}$ za koju važi jednakost $xa = bx$.

Rešenje:

Izvedimo dokaz matematičkom indukcijom po dužini reči x .

1° Za praznu reč e ne važi jednakost $ea = be$ (jer je $a \neq b$), pa tvrdjenje važi za reč x dužine 0.

Za $|x| = 1$, važi $x = a$ ili $x = b$. Međutim, kako je $aa \neq ba$ i $ba \neq bb$, sledi da tvrdjenje važi za reči x dužine 1.

2° Pretpostavimo da tvrdjenje važi za sve reči dužine $n - 2$ i dokažimo da onda važi i za reči dužine n .

Pretpostavimo suprotno – da postoji reč x dužine n takva da važi $xa = bx$. Dakle, reč x počinje slovom b , a završava se slovom a , pa reč x može biti napisana u obliku $x = bya$, gde je y reč dužine $|x| - 2 = n - 2$. Onda važi:

$$byaa = bbya ,$$

odakle na osnovu zakona skraćivanja sledi

$$ya = by ,$$

tj. reč y je rešenje zadate jednačine i $|y| = n - 2$, što je u kontradikciji sa induktivnom pretpostavkom, odakle sledi da ne postoji reč x dužine n takva da važi $xa = bx$.

Tvrdjenje je dokazano za reči dužine 0 i 1, pa, iz dokazanog induktivnog koraka, sledi da tvrdjenje važi za sve prirodne brojeve, čime je dokazano da ne postoji nijedna reč x azbuke $\Sigma = \{a, b\}$ za koju važi jednakost $xa = bx$.

Zadatak 2 Rešiti nad azbukom $\Sigma = \{a, b, c\}$ jednačinu po x : $ax = xa$.

Rešenje:

Skup rešenja jednačine je skup reči oblika a^n , ($n \geq 0$).

(\supseteq): Za svako $n \geq 0$, reč a^n jeste rešenje, jer $aa^n = a^{n+1} = a^na$.

¹Zasnovano na materijalu "Teorija algoritama jezika i automata" Predraga Janičića

(\subseteq): Dokaz indukcijom po dužini reči x , da je svako rešenje oblika a^n ($n \geq 0$).

(1°) Za $|x| = 0$ tj. $x = e$ i $ax = xa$ važi $x = a^0$.

Za $|x| = 1$ iz $ax = xa$ sledi $x = a$, tj. $x = a^1$.

(2°) Pretpostavimo da je tvrdjenje tačno za sve reči dužine $n - 2$ i dokažimo da je tačno i za reči dužine n . Neka je $|x| = n$ i neka je $ax = xa$. Dakle, reč x počinje i završava se slovom a , pa je $x = aya$, gde je y reč nad zadatom azbukom dužine $n - 2$. Skraćivanjem se iz $aaaya = aya$ dobija $ay = ya$, pa je reč y rešenje zadate jednačine dužine $n - 2$. Na osnovu induktivne pretpostavke, reč y je oblika a^n , $n \geq 0$, pa je reč x oblika a^{n+2} , $n \geq 0$.

Dakle, svako rešenje date jednačine je oblika a^n ($n \geq 0$).

Dakle, skup rešenja jednačine je skup reči oblika a^n ($n \geq 0$).

2.1.2 Gramatike i jezici: zadaci

2

Zadatak 3 Odrediti jezik generisan gramatikom $G = (N, \Sigma, P, S)$, gde je $N = \{S\}$, $\Sigma = \{a, b\}$, $P = \{S \rightarrow aS \text{ (1°), } S \rightarrow b \text{ (2°)}\}$.

Rešenje:

(Primer izvodjenja u gramatici G : $S \xrightarrow{1^\circ} aS \xrightarrow{1^\circ} aaS \xrightarrow{1^\circ} aaaS \xrightarrow{2^\circ} aaab$)

Dokažimo da važi:

$$L(G) = \{a^n b \mid n \in \mathbf{N}_0\}.$$

\subseteq : Svaka završna reč koja se izvodi u gramatici G je oblika $a^n b, n \in \mathbf{N}_0$.

Dokažimo indukcijom jače tvrdjenje:

Sve reči koje mogu biti izvedene u gramatici G su oblika $a^n S$ ili oblika $a^n b$ ($n \in \mathbf{N}_0$).

Dokaz indukcijom po dužini izvodjenja:

(1) $k = 0$: U izvodjenju nije primenjeno nijedno pravilo izvodjenja, tj. izvodjenje je trivijalno; za početni simbol S , dakle, dobija se takodje reč S . Kako je $S = a^0 S$, tvrdjenje važi za $k = 0$.

(2) Pretpostavimo da tvrdjenje važi za sva izvodjenja čija je dužina manja od k i dokažimo da važi i za izvodjenja dužine k .

Neka je $S \Rightarrow^k w$. Neka je $S \Rightarrow^{k-1} w' \Rightarrow w$. Reč w' je izvedena izvodjenjem dužine $k - 1$, pa je, na osnovu induktivne hipoteze, w' oblika $a^n S$ ili oblika $a^n b$ ($n \in \mathbf{N}_0$). Reč w se netrivialno (izvodjenjem dužine 1) izvodi iz reči w' , pa w' mora da ima nezavršnih simbola. Dakle, w' je oblika $a^n S$. Ako je u k -tom koraku primenjeno pravilo 1°, onda je w oblika $a^{n+1} S$, a ako je primenjeno pravilo 2°, onda je w oblika $a^n b$.

Dakle, sve reči koje mogu biti izvedene u gramatici G su oblika $a^n S$ ili oblika $a^n b$ ($n \in \mathbf{N}_0$), što je i trebalo dokazati.

Na osnovu leme, svi članovi izvodjenja (sve reči koje mogu biti izvedene u gramatici G) su oblika $a^n S$ ili oblika $a^n b$ ($n \in \mathbf{N}_0$), pa i završne reči – reči bez nezavršnih simbola. Završne reči ne mogu biti oblika $a^n S$ (jer je S nezavršni simbol), pa su sve završne reči oblika $a^n b$ ($n \in \mathbf{N}_0$). Dakle, $L(G) \subseteq \{a^n b \mid n \in \mathbf{N}_0\}$.

²Zasnovano na materijalu "Teorija algoritama jezika i automata" Predraga Janičića

\supseteq : Svaka reč $a^n b$, $n \in \mathbb{N}_0$, može biti izvedena u gramatici G .

Za proizvoljno n ($n \in \mathbb{N}_0$) reč $a^n b$ može biti izvedena u gramatici G :

$$S \xRightarrow{1^\circ} \underbrace{aS \xRightarrow{1^\circ} \dots \xRightarrow{1^\circ} a^n S}_n \xRightarrow{2^\circ} a^n b$$

Zadatak 4 Odrediti jezik generisan gramatikom $G = (N, \Sigma, P, S)$, gde je $N = \{S\}$,

$\Sigma = \{a, b\}$,

$P = \{S \rightarrow aSb \ (1^\circ), S \rightarrow e \ (2^\circ)\}$.

Zadatak 5 Odrediti gramatiku koja generiše jezik $W = \{a^{2^i} b^i \mid i > 0\}$.

Rešenje:

$G = (N, \Sigma, P, S)$, gde je $N = \{S\}$,

$\Sigma = \{a, b\}$,

$P = \{S \rightarrow aab \ (1^\circ), S \rightarrow aaSb \ (2^\circ)\}$

Dokažimo da važi $W = L(G)$.

\subseteq : Neka je $w \in W$, tj. neka je $w = a^{2^i} b^i$, gde je $i > 0$.

w se može izvesti u gramatici G .

$$S \xRightarrow{2^\circ} (aa)^{i-1} S b^{i-1} \xRightarrow{1^\circ} (aa)^i b^i.$$

Dakle, $W \subseteq L(G)$.

\supseteq : Indukcijom se može dokazati da je svaki član izvodjenja w oblika $a^{2^i} S b^i$ ($i \geq 0$) ili oblika $a^{2^i} b^i$ ($i > 0$). Završna reč w ne može da sadrži simbol S , pa je oblika $a^{2^i} b^i$ ($i > 0$), odakle sledi $W \supseteq L(G)$.

Zadatak 6 Odrediti gramatiku koja generiše jezik $W = \{a^n b^{\lfloor \frac{n}{2} \rfloor} \mid n \geq 0\}$.

Zadaci za domaći:

Zadatak 7 Dokazati da ne postoji nijedna reč y azbuke $\Sigma = \{c, d\}$ za koju važi jednakost $cy = yd$.

Zadatak 8 Rešiti nad azbukom $\Sigma = \{a, b, c\}$ jednačinu po y : $yb = by$.

Zadatak 9 Odrediti jezik generisan gramatikom $G = (N, \Sigma, P, S)$, gde je $N = \{S\}$,

$\Sigma = \{a, b\}$,

$P = \{S \rightarrow Sb \ (1^\circ), S \rightarrow a \ (2^\circ)\}$.

Zadatak 10 Odrediti gramatiku koja generiše jezik $W = \{b^i a^{3^i} \mid i \geq 0\}$.

Zadatak 11 Odrediti gramatiku koja generiše sve palindrome nad azbukom $\Sigma = \{a, b\}$.

3

3.1 Pozicioni brojni sistemi - konverzije

Pozicioni brojni sistemi su oni u kojima se težina cifre (njen udeo u celokupnoj vrednosti broja) određuje na osnovu njene pozicije u broju (što veća pozicija to je veći i udeo u vrednosti broja). Dekadni brojni sistem je pozicioni, dok rimski brojevi predstavljaju sistem koji nije pozicioni.

Kako su računari zasnovani na binarnoj aritmetici a mi smo navikli da radimo sa dekadnim brojevnim sistemom potrebno je obezbediti prevođenje brojeva iz sistema sa osnovom 10 u sistem sa osnovom 2 i obratno.

Da bi to uradili prvo ćemo posmatrati opštiji problem prevođenja brojeva iz sistema sa proizvoljnom osnovom b u sistem sa osnovom 10.

U bazi sa osnovom 10, na koju smo mi navikli, cifre koje koristimo su $0, 1, \dots, 9$ odnosno od 0 do $10 - 1$. Znači u proizvoljnoj bazi B korišćemo cifre od 0 do $B - 1$.

Najčešće korišćene baze (sem 10) su stepeni dvojke: 2, 8, 16. U sledećoj tabeli su prikazani nazivi odgovarajućih brojevnih sistema zajedno sa ciframa koje se u njima koriste.¹

Naziv	Osnova	Cifre
Dekadni	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Binarni	2	0, 1
Oktalni	8	0, 1, 2, 3, 4, 5, 6, 7
Heksadekadni	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

3.2 Prebacivanje iz sistema sa osnovom b u dekadni brojni sistem

Pozicioni brojni sistemi imaju svojstvo da niz od n cifara u sistemu sa osnovom B

$$\delta_n = d_1 d_2 \dots d_n$$

predstavlja broj

$$\tilde{\delta}_n = \sum_{i=1}^n \tilde{d}_i * B^{n-i} = (\dots (\tilde{d}_1 * B + \tilde{d}_2) * B + \dots + \tilde{d}_{n-1}) * B + \tilde{d}_n$$

u dekadnom brojnom sistemu, pri čemu \tilde{d}_i predstavlja numeričku vrednost karaktera d_i (odnosno za ako je $d_i = A$ onda je $\tilde{d}_i = 10$).

¹pri čemu u heksadekadnom sistemu slova A-F imaju redom vrednosti od 10 – 15

Na ovaj način su jedinstveno predstavljeni svi brojevi od 0 do $B^n - 1$. Ako prvih i cifara označimo sa $\delta_i = d_1 \dots d_i$ onda važi (za $\delta_0 = 0$)

$$\tilde{\delta}_i = \tilde{\delta}_{i-1} * B + \tilde{d}_i, \quad 0 \leq \tilde{d}_i < B$$

odatle takođe možemo da zaključimo da je:

$$\tilde{\delta}_{i-1} = \tilde{\delta}_i \text{ div } B, \quad d_i = \tilde{\delta}_i \text{ mod } B \quad (3.1)$$

Znači, ako pretpostavimo da su cifre broja u sistemu B redom d_1, \dots, d_n onda se njegova vrednost u dekadnom sistemu (označimo je sa x) može izračunati na sledeći način:

1. Neka je $x = 0$ i neka je i indeks tekuće cifre, $i = 1$ na početku.
2. Sada uračunavamo tekuću cifru u vrednost broja: $x = x * B + d_i$, $i = i + 1$
3. Ako je $i > n$ znači da smo uračunali sve cifre broja i da smo dobili vrednost x , inače treba da uračunamo sledeću cifru i vraćamo se na korak 2.

Primer 1 *Prevođenje iz osnova 2, 16 i 8 u osnovu 10:*

$$(1101)_2 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = (13)_{10}$$

$$(1101)_{16} = 1 * 16^3 + 1 * 16^2 + 0 * 16^1 + 1 * 16^0 = 4096 + 256 + 1 = (4353)_{10}$$

$$(F9A)_{16} = F * 16^2 + 9 * 16^1 + A * 16^0 = 15 * 16^2 + 9 * 16^1 + 10 * 16^0 = 3840 + 144 + 10 = (3994)_{10}$$

Zadatak 12 *Prebacite sledeće brojeve u dekadni brojni sistem (indeks predstavlja osnovu u kojoj su brojevi zapisani): $(110111100)_2$, $(77)_8$, $(FFFF)_{16}$*

3.3 Prebacivanje iz dekadnog brojnog sistema u sistem sa osnovom b

Inverzni algoritam koji računa niz cifara $d_1 \dots d_n = \delta_n$ koje predstavljaju broj $\tilde{\delta}_n$ u pozicionom sistemu sa osnovom B se dobija primenom jednačina 3.1.

Za dati broj $\tilde{\delta}_i = x$ ($0 \leq x < B^n$), njegova poslednja cifra u datoj reprezentaciji (sa osnovom B) se dobija kao ostatak pri deljenju broja x sa B . Da bi dobili ostale cifre potrebno je da izračunamo celobrojni količnik pri deljenju x sa B i da na njega primenimo isti algoritam.

1. Krećemo od broja x i u svakom koraku računamo $(n - i)$ -tu cifru. Na početku $i = 0$.
2. $c_i = x \text{ mod } B$, $x = x \text{ div } B$, $i = i + 1$.
3. Ako je $x = 0$ dobili smo cifre obrnutim redosledom (odnosno $d_i = c_{n-i}$), inače se vraćamo na korak 2.

Primetimo da ovaj algoritam možemo koristiti i za izdvajanje cifara dekadnog broja (proverite!).

Primer 2 *Prevođenje iz dekadnog u binarni brojni sistem*

$$(26)_{10} = (?)_2$$

Rešenje:

$$26 / 2 = 13 \text{ i ostatak } 0$$

$$13 / 2 = 6 \text{ i ostatak } 1$$

$$6 / 2 = 3 \text{ i ostatak } 0$$

$$3 / 2 = 1 \text{ i ostatak } 1$$

$$1 / 2 = 0 \text{ i ostatak } 1$$

Dakle, rešenje je broj čije se cifre dobijaju tako što se ostaci dobijeni prethodnim postupkom pročitaju obrnutim redosledom tj. $(11010)_2$

Zadatak 13 Odredite binarnu reprezentaciju sledećih brojeva: $(54)_{10}$, $(126)_{10}$, $(332)_{10}$.

Primer 3 Prevođenje iz dekadnog u oktalni brojni sistem

$$(181)_{10} = (?)_8$$

Rešenje:

$$181 / 8 = 22 \text{ i ostatak } 5$$

$$22 / 8 = 2 \text{ i ostatak } 6$$

$$2 / 8 = 0 \text{ i ostatak } 2$$

Dakle, rešenje je broj čije se cifre dobijaju tako što se ostaci dobijeni prethodnim postupkom pročitaju obrnutim redosledom tj. $(265)_8$

Zadatak 14 Odredite oktalnu prezentaciju sledećih brojeva: $(67)_{10}$, $(336)_{10}$, $(442)_{10}$

Primer 4 Prevođenje iz dekadnog u heksadekadni brojni sistem

$$(181)_{10} = (?)_{16}$$

Rešenje:

$$181 / 16 = 11 \text{ i ostatak } 5$$

$$11 / 16 = 0 \text{ i ostatak } 11 (\text{heksadekadna cifra } B)$$

Dakle, rešenje će biti broj čije se cifre dobiju tako što se ostaci dobijeni prethodnim postupkom pročitaju unazad tj. $(B5)_{16}$

Zadatak 15 Odredite heksadekadnu prezentaciju sledećih brojeva: $(48)_{10}$, $(1336)_{10}$, $(332)_{10}$.

3.4 Rad sa realnim brojevima

Kada radimo sa realnim brojevima možemo posebno posmatrati ceo deo broja i razlomljeni deo broja. Kako smo do sada radili sa celim brojevima posmatrajmo brojeve oblika $x = 0.d_1d_2 \dots d_n$, za koje je $0 \leq x < 1$.

3.4.1 Prevođenje realnih brojeva iz sistema sa osnovom B u dekadni brojni sistem

Neka je $\delta = 0.d_1d_2 \dots d_n$ razlomljeni deo broja x u sistemu sa osnovom B . Tada će njegova vrednost u dekadnom sistemu biti:

$$\tilde{\delta} = \sum_{i=1}^n d_i * B^{-i} = \frac{1}{B}(\tilde{d}_1 + \frac{1}{B}(\tilde{d}_2 + \dots + \frac{1}{B}\tilde{d}_n) \dots) \quad (3.2)$$

Pri čemu je \tilde{d}_i numerička vrednost "cifre" d_i . Odatle dobijamo sledeći algoritam:

1. Neka je $x = 0$ (dekadna vrednost broja), $i = 1$ indeks tekuće cifre koju uračunavamo u vrednost broja i $f = \frac{1}{B}$ tekući koeficijent sa kojim množimo cifru.
2. $x = x + d_i * f$, $i = i + 1$, $f = f * \frac{1}{B}$.
3. Ako je $i > n$ znači da smo uračunali sve cifre i da se u x nalazi dekadna vrednost broja, inače se vraćamo na korak 2.

Primer 5 $(0.1101)_2 = 1 * 2^{-1} + 1 * 2^{-2} + 0 * 2^{-3} + 1 * 2^{-4} = (0.6875)_{10}$

Zadatak 16 Prebacite sledeće brojeve u dekadni brojni sistem (indeks predstavlja osnovu u kojoj su brojevi zapisani): $(0.1011)_2$, $(0.77)_8$, $(0.FF)_{16}$

3.4.2 Prevođenje realnih brojeva iz dekadnog brojnog sistema u sistem sa osnovom B

Neka je $\delta_i = 0.d_i \dots d_n$. Na osnovu jednačine 3.2 vidimo da važi:

$$\tilde{\delta}_i = \frac{1}{B}(\tilde{d}_i + \tilde{\delta}_{i+1}), \quad 0 \leq \tilde{\delta}_i < 1$$

Odnosno

$$\tilde{d}_i = \text{trunc}(B * \tilde{\delta}_i)$$

$$\tilde{\delta}_{i+1} = B * \tilde{\delta}_i - \tilde{d}_i$$

pri čemu je $\text{trunc}(x)$ ceo deo broja x . Odatle direktno vidimo na koji način možemo izračunati cifre broja u sistemu sa osnovom B i dobijamo sledeći algoritam ²:

1. Neka je x broj čiji zapis određujemo, $i = 1$ indeks tekuće cifre koju računamo.
2. $d_i = \text{trunc}(B * x)$.
3. $x = B * x - d_i$, $i = i + 1$.
4. Ako je $i = n$ dobili smo n cifara razlomljenog dela broja, inače se vraćamo na korak 2.

Ovim algoritmom se cifre dobijaju u željenom redosledu, odnosno od prve ka poslednjoj.

Primer 6 Odrediti binarni zapis broja $x = (0.867)_{10}$ na 4 decimale.

$$0.867 * 2 = 1.734, \text{ ceo deo } 1$$

$$0.734 * 2 = 1.468, \text{ ceo deo } 1$$

$$0.468 * 2 = 0.936, \text{ ceo deo } 0$$

$$0.936 * 2 = 1.872, \text{ ceo deo } 1$$

Dakle rešenje se dobija tako što se cifre čitaju onim redosledom kojim su dobijene tj. $(0.1101)_2$

3.5 Direktno prevođenje iz binarnog u heksadekadni sistem

Za kodiranje heksadekadnih cifara dovoljne su binarne reči dužine četiri ($16 = 2^4$).

Heksadekadna cifra	Binarni kod	Heksadekadna cifra	Binarni kod
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

²pri čemu primetimo da iz petlje izlazimo kada dobijemo željeni broj cifara a ne kada x postane 0 iz razloga što radimo sa realnim brojevima koji ne moraju imati konačan zapis

Primetimo da je na ovaj način svakoj heksadekadnoj cifri jedinstveno dodeljen kod dužine četiri u binarnom sistemu što nam omogućava da obavljamo direktno prevođenje iz binarnog u heksadekadni sistem na sledeći način:

Binarne cifre se grupišu u grupe od 4 cifre, počev od bitova najmanje težine. Ako ukupan broj bitova nije deljiv sa četiri, onda se dopisuje potreban broj vodećih nula (one su bez uticaja na promenu vrednosti originalnog zapisa).

Primer 7 $(1111011100001101010000)_2 = (0011\ 1101\ 1100\ 0011\ 0101\ 0000)_2 = (3DC350)_{16}$

Zadatak 17 *Odredite heksadekadni zapis sledećeg binarnog broja $(11010100100)_2$*

3.6 Direktno prevođenje iz binarnog u oktalni sistem

Za kodiranje oktalnih cifara dovoljne su binarne reci dužine tri ($8 = 2^3$).

Oktalna cifra	Binarni kod	Oktalna cifra	Binarni kod
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

Sada smo svakoj oktalnoj cifri jedinstveno dodelili binarni kod dužine tri što nam omogućava direktno prevođenje. Binarne cifre se grupišu grupe od po 3 cifre, počev od bitova najmanje težine. Ako ukupan broj bitova nije deljiv sa tri, onda se dopisuje potreban broj vodećih nula.

Primer 8 $(11111010001010)_2 = (011\ 111\ 010\ 001\ 010)_2 = (37212)_8$

Zadatak 18 *Odredite oktalni zapis sledećeg binarnog broja $(11010100100)_2$*

4

1

4.1 Reprezentacija znakovnih podataka

4.1.1 Tekst je niz karaktera

- Iako obično tekst zamišljamo kao dvodimenzioni objekat, u računarima se tekst predstavlja kao jednodimenzioni (linearni) niz karaktera.
- Potrebno je, dakle, uvesti specijalne karaktere koji označavaju prelazak u novi red, tabulator, kraj teksta i slično

4.1.2 Zapis karaktera u računar

- Računari su zasnovani na binarnoj aritmetici
- Cele brojeve je moguće predstaviti u binarnom sistemu
- Osnovna ideja je svakom karakteru pridružiti određeni ceo broj na unapred dogovoreni način
- Ove brojeve zovemo kodovima karaktera (character codes)

4.1.3 Skupovi karaktera

- Koliko karaktera želimo da predstavimo u računarima? Tokom razvoja računarstva broj karaktera je postajao sve veći
 - Pošto je u početku razvoja englesko govorno područje bilo dominantno osnovno je bilo predstaviti sledeće karaktere :
 - Velika slova engleskog alfabeta : A,B,...,Z
 - Mala slova engleskog alfabeta : a,b,...,z
 - Cifre : 0,1,...,9
 - Interpunkcijske znake : ., ; i slično
 - Kontrolne znake : kraj reda, tabulator i slično
 - Standardni karakterski kodovi: Sedamdesetih godina su se pojavile tabele standardnih karakterskih kodova dovoljne za zapis pomenutih karaktera
- Najpoznatiji su

¹Zasnovano na materijalu "Zapis tekstova u računar" Filipa Marića

- EBCDIC IBM-ov standard, pogodan za bušene kartice
- ASCII Standard iz koga se razvila većina današnjih standarda
- ASCII (American Standard Code for Information Interchange)
ASCII je sedmobitan (broj karaktera koji je njime predstavljen je $128 = 2^7$)
- Npr. kod za A je $(41)_{16}$ tj. $0x41$ što je $(65)_{10}$ tj. $(1000001)_2$,
kod za a je $(61)_{16}$ tj. $0x61$ što je $(95)_{10}$ tj. $(1100001)_2$.
- PRIMER: transformisanje malih slova u velika
- Razmak SP se zapisuje kao $(20)_{16}$ što je $(32)_{10}$ tj. $(0100000)_2$.
- **Osobine ASCII koda:** Prvih 32 karaktera (kodovi $0x00-0x1F$) i poslednji karakter (kod $0x7F$) su kontrolni karakteri. To su karakteri bez grafije, kao CR (kod $0x0D$), LF (kod $0x0A$).
- Prvi karakter sa grafijom je blanko (kod $0x20$). Njegova grafija je belina.
- Skup velikih slova A-Z (kodovi $0x41-0x5A$), kao i skup malih slova a-z (kodovi $0x61-0x7A$), je u alfabetskom redosledu unutar kolacione sekvencije
($0x41 < 0x42$, prema tome $A < B$ to odgovara alfabetskom redosledu).
- Skup cifara 0 – 9 (kodovi $0x31 - 0x39$) je u rastućem brojčanom redosledu unutar kolacione sekvencije
($0x31 < 0x32$, prema tome $1 < 2$ što odgovara brojčanom redosledu).
- Skup velikih slova A-Z (kodovi $0x41 - 0x5A$), skup malih slova a-z (kodovi $0x61 - 0x7A$) i skup cifara 0-9 (kodovi $0x31 - 0x39$) su kontingentni unutar kolacione sekvencije (između slova A i slova Z nema drugih karaktera osim onih koji odgovaraju velim slovima engleske abecede). Sve cifre prethode svim velikim slovima, sva velika slova prethode svim malim slovima u kolacionoj sekvenciji. Specijalni i interpunkcijski znaci su izmešani između njih.
- Kod svakog velikog slova je za 32 (ili $0x20$) manji od koda odgovarajućeg malog slova. Na primer, za slovo E važi da je $0x45 + 0x20 = 0x65$, odnosno, $01000101 + 00100000 = 01100101$. Prema tome, binarni kodovi velih i malih slova razlikuju se samo u jednoj cifri, onoj koja odgovara petom stepenu osnove 2.

4.1.4 Kodiranje ostalih jezika

- Razvojem računarstva se javlja potreba kodiranja tekstova i na drugim jezicima
- Kroz istoriju su postojala mnoga rešenja, od kojih su se neka zadržala, a neka su nestala

4.1.5 Osobine YUSCII koda

- ASCII kod je jedna verzija međunarodnog standarda ISO 646 IRV koji predstavlja međunarodnu referentnu verziju za 7-bitni kod. Ovaj standard propisuje da se pozicijama $0x40$, $0x5B - 0x5E$, $0x60$ i $0x7B - 0x7E$ ne pridružuje obavezna grafija već se da se one u nacionalnim verzijama standarda i u određenim aplikacijama mogu slobodno koristiti.
- Standard JUS.B1.002 koristi ovih 10 pozicija za kodiranje slova specifičnih za srpsku latinicu Ž, Š, Đ, Č, Ć.
- Yu-ASCII skup zadržava sve navedene osobine ASCII koda osim jedne, a ta je da ni velika ni mala slova nisu u alfabetskom redosledu unutar kolacione sekvencije. Naime, $0x40 < 0x41$, dakle $\tilde{Z} < A$ što ne odgovara alfabetskom redosledu unutar kolacione sekvencije. Nakon slova Ž slede velika slova engleske abecede, zatim slova Š ($0x5B$), Đ ($0x5C$), Č ($0x5D$), Ć ($0x5E$), ž ($0x60$), zatim slede mala slova abecede, i na kraju slova š ($0x7B$), đ ($0x7C$), ć ($0x7D$), č ($0x7E$).

Zadatak 19 Poredati slova vašeg prezimena koristeći YUSCII kodnu šemu.

4.1.6 Kodne strane

- Pod *kodnom* stranom (Code page) tj. *skupom karaktera* (Character set, charset) podrazumevamo uređenu listu karaktera predstavljenih svojim karakterskim kodovima
- Podaci se u računarima obično zapisuju bajt po bajt
- ASCII je sedmobitni standard
- ASCII karakteri se zapisuju tao što se u svakom bajtu bit najveće težine postavi na 0
- To ostavlja prostor za novih 128 karaktera čiji binarni zapis počinje sa 1
- Ovaj prostor se može popuniti na razne načine
- Rešenje nije univerzalno, jer svakako na svetu postoji više od 256 različitih karaktera
- Postavljeni su razni standardi dopunjavanja ovih 128 karaktera
- Svim ovim kodnim stranama je zajedničko prvih 128 karaktera i oni se poklapaju sa ASCII
- Ovako napravljene kodne strane obično omogućuju kodiranje tekstova na više srodnih jezika (obično i geografski bliskih)
- Nama su uglavnom važne kodne strane napravljene za centralno-evropske (Central European) latinice, kao i ćirilicne kodne strane

4.1.7 Kodne strane kod nas

- Najčešće korišćene kodne strane kod nas (Prve dve su delo međunarodne organizacije za standardizaciju (International Standard organization), dok su naredne dve Microsoft-ovi standardi):
 - ISO 8859-2 (Latin2)
 - ISO 8859-5 (Ćirilicna)
 - Windows 1250
 - Windows 1251 (Ćirilicna)
- **Latin 1:** Poželjno je poznavati i osnovnu kodnu stranu ISO 8859-1 (Latin1) jer je veoma često postavljena kao podrazumevana kodna strana. Ona se koristi za zapis tekstova na zapadno evropskim jezicima (Western European)

4.1.8 Višebajtni karakterski kodovi

- Iako navedene kodne strane omogućuju kodiranje tekstova koji nisu na engleskom jeziku nije moguće npr. u istom tekstu mešati ćirilicu i našu latinicu.
- Azijskim jezicima nije dovoljno 256 mesta za zapis svih karaktera.
- Zbog toga se uvode višebajtni karakterski kodovi

- MBCS: Pre svega zbog potreba istočno azijskih korisnika uvedeni su tzv. višebajtni skupovi karaktera tj. Multi-Byte Character Sets (MBCS)
- Ideja je u tome da se najčešće korišćeni karakteri zapisuju koristeći samo jedan bajt, dok se ostali karakteri zapisuju koristeći dva bajta, tj. koristi se mešavina jednobajtnih i dvobajtnih karakterskih kodova (pod UNIX-om nekad čak i trobajtnih)
- Ovo značajno otežava tumačenje podataka
- **ISO 10646** je zamišljen kao 4 bajtni standard. Pri tome se prvih 65536 karaktera koriste kao osnovni višjezični skup karaktera dok je ostali prostor ostavljen kao proširenje za drevne jezike, celokupnu naucnu notaciju i slično.
- **UNICODE**: svakom karakteru dodeljuje dvobajtni kod
- Prvih 128 karaktera se poklapaju sa ASCII standardom, dok su sledećih 128 napravljeni tako da se poklapaju sa Latin1 standardom
- UCS-2: Unicode standard u suštini predstavlja veliku tabelu koja svakom karakteru dodeljuje broj.
- Standardi koji opisuju kako se niske karaktera onda prevode u nizove bajtova se dodatno definišu
- ISO definiše UCS-2 standard koji jednostavno svaki UNICODE karakter prevodi u odgovarajuća dva bajta
- UTF: A Unicode transformation format (UTF) algoritam koji svakom UNICODE karakteru dodeljuje određeni niz bajtova čija dužina varira od 1 do najviše 6.
- UTF je ASCII kompatibilan, što znaci da se ASCII karakteri zapisuju pomoću jednog bajta, na standardni način.
- Najčešće korišćena varijanta ovog algoritma je UTF-8 koja je dovoljna za zapis svih dvobajtnih UNICODE karaktera
- Pored ovoga ISO uvodi i UTF-16, UTF-32, kao i standard UCS-4

4.1.9 Karakteri, Glifovi, Fontovi

- Vrlo često se ne pravi jasna razlika između karaktera i njihove graficke reprezentacije
- Grafičku reprezentaciju karaktera nazivamo glifovima (*glyph*) Skupove glifova nazivamo fontovima (*font*)
- Korespondencija između karaktera i glifova ne mora biti jednoznačna
- Jedan glif može da predstavi više karaktera (ligature)
- Isti karakter može da se predstavlja različitim glifovima u zavisnosti od svoje pozicije u reči
- Za razliku od tradicionalnih fontova koji u sebi sadže glifove za karaktere jedne kodne strane, TrueType fontovi koji podržavaju WGL4 standard sadrže glifove za sve evropske karaktere

Zadatak 20 Zapisati cifru 3 u ASCII kodu.

Rešenje:

Broj 3 se zapisuje kao $(33)_{16}$ tj. $0x33$ što je $(51)_{10}$ tj. $(1010001)_2$

Zadatak 21 Zapisati reč Fakultet u ASCII kodu.

Zadatak 22 Zapisati reči MATF i lišće u kodnim stranama ISO 8859-2, Windows 1250, Windows 1251.

Rešenje:

Reč *MATF* se zapisuje isto u kodnim stranama ISO 8859-2, Windows 1250 i Windows 1251 zato što su njeni karakteri zapravo ASCII karakteri a svim ovim kodnim stranama zajedničko je prvih 128 karaktera i oni se poklapaju sa ASCII kodovima.

Dakle, reč *MATF* se u ovim kodnim stranama zapisuje preko 4 bajta i to $(4D)_{16}$, $(41)_{16}$, $(54)_{16}$, $(46)_{16}$ a to je isto što i $(77)_{10}$, $(65)_{10}$, $(84)_{10}$, $(70)_{10}$ odnosno $(01001101)_2$, $(01000001)_2$, $(01010100)_2$, $(01000110)_2$.

Reč *lišće* sadrži u sebi karaktere š i ć koji nisu ASCII karakteri pa se različito kodiraju u svakoj od kodnih strana.

U kodnoj strani ISO 8859-2 odgovarajući kod je $(6c)_{16}$, $(69)_{16}$, $(b9)_{16}$, $(e6)_{16}$, $(65)_{16}$ a to je isto što i $(108)_{10}$, $(105)_{10}$, $(185)_{10}$, $(230)_{10}$, $(101)_{10}$ odnosno $(01101100)_2$, $(01101001)_2$, $(10111001)_2$, $(11100110)_2$, $(01100101)_2$.

U kodnoj strani Windows 1250 karakteri š i ć se kodiraju sa $(9A)_{16}$, $(E6)_{16}$.

U kodnoj strani Windows 1251 karakteri š i ć se ne mogu kodirati.

Zadatak 23 Šta predstavlja niz kodova 138 65 111 33 u kodnoj strani ISO 8859-2? A u Latin1?

Rešenje:

U kodnoj strani ISO 8859-2 ovaj niz kodova predstavlja |Ao! a u Latin1 ŠAo!

5

5.1 Meta jezici: EBNF, sintaksni dijagrami

5.1.1 BNF

Meta jezik je jezik koji služi da se pomoću njega opiše neki drugi jezik ili isti taj jezik. Tako se na primer služimo srpskim jezikom da bismo opisali gramatiku srpskog jezika. Bitno je razlikovati term meta jezika od terma jezika koji se opisuje.

BNF (Bekusova normalna forma) je formalni meta jezik za predstavljanje kontekstno-slobodnih gramatika odnosno gramatika programskih jezika.

Meta simboli BNF-a su:

- `::=` u značenju "je definisano kao"
- `|` u značenju "ili"
- `<` `>` uglaste zagrade koje se koriste da uokvire odgovarajući ne-terminal.

Uloga uglastih zagrada je takođe da razdvoji ne-terminalne od terminalnih simbola koji se pišu pod navodnicima.

Sledi nekoliko primera BNF-a :

- BNF za cifru.

```
<cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0" ;
```

- BNF za neoznačen ceo broj.

```
<NeoznaceniCeoBroj> ::= <cifra> | <cifra> <NeoznaceniCeoBroj>;  
<cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0" ;
```

Drugo pravilo može da se napiše i kao:

```
<NeoznaceniCeoBroj> ::= <cifra> | <NeoznaceniCeoBroj> <cifra> ;
```

- BNF za ceo broj.

```
<CeoBroj> ::= <NeoznaceniCeoBroj>  
            | "+" <NeoznaceniCeoBroj>  
            | "-" <NeoznaceniCeoBroj>  
<NeoznaceniCeoBroj> ::= <cifra> | <cifra> <NeoznaceniCeoBroj>;  
<cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0" ;
```

- BNF za realne brojeve.

```

<RealanBroj> ::= "-" <NeoznaceniRealanBroj>
               | <NeoznaceniRealanBroj>;
<NeoznaceniRealanBroj> ::= <NeoznaceniCeoBroj>
                           | <NeoznaceniCeoBroj> "." <NeoznaceniCeoBroj>;
<NeoznaceniCeoBroj> ::= <cifra> | <cifra> <NeoznaceniCeoBroj>;
<cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0" ;

```

- BNF za identifikator.

```

<identifikator> ::= <slovo>
                  | <identifikator> <slovo>
                  | <identifikator> <cifra>;
<cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0" ;

```

5.1.2 EBNF

EBNF (proširena Bekusova normalna forma) je takođe meta jezik za predstavljanje kontekstno-slobodnih gramatika koji ima istu izražajnu moć kao i BNF, samo je zapis takav da je lakši za razumevanje. Zapravo, BNF koristi rekursiju da bi se izrazile relacije a EBNF koristi iteraciju.

Skup meta simbola BNF-a proširen je sa sledećim meta simbolima:

- Pravaugone zagrade "[...]" označavaju da se ono što se nalazi u njima pojavljuje opciono (ili se pojavljuje jednom ili se ne pojavljuje).
- sufiks "*" označava da se simbol pojavljuje 0 ili više puta. Istu ulogu imaju i vitičaste zagrade "{...}".
- sufiks "+" za jedno ili više pojavljivanja simbola
- sufiks "?" za nula ili jednu pojavu simbola.

Sve ove konstrukcije mogu biti izražene i u BNF-u što je pokazatelj toga da sve što se može zapisati u EBNF-u može i u BNF-u.

Sledi nekoliko primera EBNF-a :

- EBNF za neoznačen ceo broj.

```

<NeoznaceniCeoBroj> ::= ( <cifra> ) +;
<cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0" ;

```

- EBNF za ceo broj.

```

<CeoBroj> ::= [ + | - ] <NeoznaceniCeoBroj>
<NeoznaceniCeoBroj> ::= ( <cifra> ) +;
<cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0" ;

```

Pri čemu se prva dva pravila mogu napisati i kao:

```

<CeoBroj> ::= [ + | - ] ( <cifra> ) +

```

- EBNF za realne brojeve.

```

<RealanBroj> ::= "-"? <cifra>+ ( "." <cifra>+ )?
<cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0" ;

```

- EBNF za identifikator.

```

<identifikator> ::= <slovo>(<slovo>|<cifra>)*

```

ili

```

<identifikator> ::= <slovo> { <slovo> | <cifra> }

```

Zadatak 24 napisati BNF/EBNF za aritmetički izraz i sl.

Rešenje:

BNF:

```

<ArIzraz> ::= <term> | <ArIzraz> "+" <term>
<term> ::= <factor> | <term> "*" <factor>
<factor> ::= <broj> | "(" <ArIzraz> ")"

```

EBNF:

```

<ArIzraz> = <term> { "+" <term> }
<term> = <factor> { "*" <factor> }
<factor> = <broj> | "(" <ArIzraz> ")"

```

Zadatak 25 napisati BNF/EBNF za klauzu (nad nekim fiksni skupom iskaznih slova)

Zadatak 26 napisati BNF/EBNF za klauzu dužine 5 (nad nekim fiksni skupom iskaznih slova).

Zadatak 27 Napisati BNF/EBNF za iskaznu formulu (nad nekim fiksni skupom iskaznih slova)

5.1.3 Sintaksni dijagrami

Sintaksni dijagrami predstavljaju grafičku notaciju za predstavljanje kontekstno-slobodnih gramatika. To su dijagrami slični dijagramima toka. Čitanje sintaksnog dijagrama znači kretanje od leve ka desnoj strani prateći strelice. Ono što je bitno to je da oni imaju istu izražajnu moć kao i BNF ili EBNF.

Pogledajmo kako izgledaju sintaksni dijagrami nekoliko već pomenutih gramatika:

- Sintaksni dijagram za **cifra**:

```

<cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0" ;

```

```

cifra : ( '1' — '2' — '3' — '4' — '5' — '6' — '7' — '8' — '9' — '0' ) ;

```

- Sintaksni dijagram za **NeoznacenoBroj**:

```

<NeoznacenoBroj> ::= ( cifra ) +

```

```

broj : ( cifra + ) ;

```

- Sintaksni dijagram za **CeoBroj**:

```

<CeoBroj> ::= [ "+" | "-" ] ( <cifra> )+ ;

```

CeoBroj : ('+' — '-') ? (cifra +) ;

- Sintaksni dijagram za RealanBroj:

<RealanBroj> ::= "-"? <cifra>+ ("." <cifra>+) ?

RealanBroj : '-' ? (cifra +) ('.' (cifra +)) ? ;

- Sintaksni dijagram za identifikator:

<identifikator> ::= <slovo> { <slovo> | <cifra> }

ili

<identifikator> ::= <slovo> (<slovo> | <cifra>) *

identifikator : slovo ((cifra — slovo) *) ;

6

Programski jezik C

1

6.1 Funkcije printf() i scanf() - osnovna upotreba

Primer 9 Program na standardni izlaz štampa "Zdravo, svete!".

```
#include <stdio.h>

main()
/*iskazi f-je main su zatvoreni u zagrade */
{
/*poziv f-je printf da odštampa poruku*/
printf("Zdravo, svete!\n");
}
```

Izlaz iz programa:
Zdravo, svete!

Primer 10 Šta je izlaz iz sledećeg programa?

```
#include <stdio.h>

main()
{
printf("Zdravo, ");
printf("svete!");
printf("\n");
}
```

Izlaz iz programa:
Zdravo, svete!

¹Zasnovano na primerima sa sajtova <http://www.matf.bg.ac.yu/~milena>, <http://www.matf.bg.ac.yu/~filip>, <http://www.matf.bg.ac.yu/~jelenagr>.

6.2 Aritmetičke i relacijske operacije

Primer 11 *Program vrši oduzimanje dva cela broja.*

```
#include <stdio.h>

main()
{
    /*deklaracija vise promenljivih istog tipa */
    int rez,pom1,pom2; /*rezultat oduzimanja pom1-pom2 -> rez*/
    pom1=20;
    pom2=15;
    rez=pom1-pom2;

    /*ispisivanje rezultata*/
    printf("Rezultat je %d-%d=%d\n",pom1,pom2,rez);
}
```

Izlaz iz programa:
Rezultat je 20-15=5

Primer 12 *Program sabira dva uneta cela broja*

```
#include <stdio.h>

int main()
{
    int a, b, c;
    printf("Unesi prvi broj : ");
    scanf("%d", &a);
    printf("Unesi drugi broj : ");
    scanf("%d", &b);
    c = a + b;
    printf("%d + %d = %d\n", a, b, c);
    return 0;
}
```

Ulaz:
Unesi prvi broj : 2 <enter>
Unesi drugi broj : 3 <enter>
Izlaz:
2 + 3 = 5

Primer 13 *Program ilustruje neke od aritmetičkih operacija.*

```
#include <stdio.h>
main()
{
    int a, b;
    printf("Unesi prvi broj : ");
```



```

scanf("%d",&a);

printf("Unesi drugi broj : ");
scanf("%d",&b);

printf("Zbir a+b je : %d\n",a+b);
printf("Razlika a-b je : %d\n",a-b);
printf("Proizvod a*b je : %d\n",a*b);
printf("Celobrojni kolicnik a/b je : %d\n", a/b);
printf("Pogresan pokusaj racunanja realnog kolicnika a/b je : %f\n", a/b);
printf("Realni kolicnik a/b je : %f\n", (float)a/(float)b);
printf("Ostatak pri deljenju a/b je : %d\n", a%b);
}

```

Ulaz:
 Unesi prvi broj : 2 <enter>
 Unesi drugi broj : 3 <enter>
 Izlaz:
 Zbir a+b je : 5
 Razlika a-b je : -1
 Proizvod a*b je : 6
 Celobrojni kolicnik a/b je : 0
 Pogresan pokusaj racunanja realnog kolicnika a/b je : 0.000000
 Realni kolicnik a/b je : 0.666667
 Ostatak pri deljenju a/b je : 2

Primer 14 Program ilustruje celobrojno i realno deljenje.

```
#include <stdio.h>
```

```

main()
{
    int a = 5;
    int b = 2;
    int d = 5/2;    /* Celobrojno deljenje - rezultat je 2 */
    float c = a/b;  /* Iako je c float, vrsi se celobrojno deljenje jer su i a i b celi */

    /* Neocekivani rezultat 3.000000 */
    printf("c = %f\n",c);

    printf("Uzrok problema : 5/2 = %f\n", 5/2);

    printf("Popravljeno : 5.0/2.0 = %f\n", 5.0/2.0);

    printf("Moze i : 5/2.0 = %f i 5.0/2 = %f \n", 5/2.0, 5.0/2);

    printf("Za promenjive mora kastovanje : %f\n", (float)a/(float)b);
}

```

Izlaz iz programa:

c = 2.000000

Uzrok problema : $5/2 = 0.000000$
Popravljeno : $5.0/2.0 = 2.500000$
Može i : $5/2.0 = 2.500000$ i $5.0/2 = 2.500000$
Za promenljive mora kastovanje : 2.500000

Primer 15 *Ilustracija prefiksnog i postfiksno operatora ++*

```
#include <stdio.h>
main()
{
    int x, y;
    int a = 0, b = 0;

    printf("Na pocetku : \na = %d\nb = %d\n", a, b);

    /* Ukoliko se vrednost izraza ne koristi, prefiksni i
       postfiksni operator se ne razlikuju */
    a++;
    ++b;
    printf("Posle : a++; ++b; \na = %d\nb = %d\n", a, b);

    /* Prefiksni operator uvecava promenjivu, i rezultat
       je uvecana vrednost */
    x = ++a;

    /* Postfiksni operator uvecava promenjivu, i rezultat je
       stara (neuvecana) vrednost */
    y = b++;

    printf("Posle : x = ++a; \na = %d\nx = %d\n", a, x);
    printf("Posle : y = b++; \nb = %d\nny = %d\n", b, y);
}
```

Izlaz iz programa:

Na pocetku:

a = 0

b = 0

Posle : a++; ++b;

a = 1

b = 1

Posle : x = ++a;

a = 2

x = 2

Posle : y = b++;

b = 2

y = 1

Primer 16 *Ilustracija logičkih vrednosti (0 - netačno, razlicito od 0 - tačno).*

```
#include <stdio.h>

main()
{
    int a;

    printf("Unesi ceo broj : ");
    scanf("%d", &a);
    if (a)
        printf("Logicka vrednost broja je : tacno\n");
    else
        printf("Logicka vrednost broja je : netacno\n");
}
```

Ulaz:
Unesi ceo broj : 3 <enter>
Izlaz:
Logicka vrednost broja je : tacno

Ulaz:
Unesi ceo broj : 0 <enter>
Izlaz:
Logicka vrednost broja je : netacno

Primer 17 *Ilustracija ogičkih i relacijskih operatora.*

```
#include <stdio.h>

main()
{
    int a = 3>5, /* manje */
        b = 5>3, /* vece */
        c = 3==5, /* jednako */
        d = 3!=5; /* razlicito */

    printf("3>5 - %d\n5>3 - %d\n3==5 - %d\n3!=5 - %d\n", a, b, c, d);

    printf("Konjunkcija : 3>5 && 5>3 - %d\n", a && b);
    printf("Disjunkcija : 3>5 || 5>3 - %d\n", a || b);
    printf("Negacija : !(3>5) - %d\n", !a);
}
```

Izlaz iz programa:
3>5 - 0
5>3 - 1
3==5 - 0
3!=5 - 1
Konjunkcija : 3>5 && 5>3 - 0
Disjunkcija : 3>5 || 5>3 - 1
Negacija : !(3>5) - 1

6.3 Kontrola toka - if, if-else, while, do-while, switch

Primer 18 Program ilustruje if i ispisuje ukoliko je uneti ceo broj negativan.

```
#include <stdio.h>

int main()
{
    int b;
    printf("Unesi ceo broj:");
    scanf("%d", &b);
    if (b < 0)
        printf("Broj je negativan\n");
    return 0;
}
```

Ulaz:
Unesi ceo broj:-5
Izlaz:
Broj je negativan

Ulaz:
Unesi ceo broj:5
Izlaz:

Primer 19 Program ilustruje if-else konstrukciju i ispituje znak broja.

```
#include <stdio.h>

int main()
{
    int b;
    printf("Unesi ceo broj : ");
    scanf("%d", &b);
    if (b < 0)
        printf("Broj je negativan\n");
    else if (b == 0)
        printf("Broj je nula\n");
    else
        printf("Broj je pozitivan\n");
    return 0;
}
```

Ulaz:
Unesi ceo broj:-5
Izlaz:
Broj je negativan

Ulaz:
Unesi ceo broj:5
Izlaz:
Broj je pozitivan

Primer 20 *Pogresan program sa dodelom = umesto poredjenja ==.*

```
#include <stdio.h>

int main()
{
    int b;
    printf("Unesi ceo broj : ");
    scanf("%d", &b);

    /* Obratiti paznju na = umesto == Analizirati rad programa*/
    if (b = 0)
        printf("Broj je nula\n");
    else if (b < 0)
        printf("Broj je negativan\n");
    else
        printf("Broj je pozitivan\n");
    return 0;
}
```

Ulaz:
Unesi ceo broj:-5
Izlaz:
Broj je pozitivan

Primer 21 *Program ilustruje petlju - while.*

```
#include <stdio.h>

int main()
{
    int x;

    x = 1;
    while (x<10)
    {
        printf("x = %d\n",x);
        x++; /* x++ je isto kao i x=x+1 */
    }
}

Izlaz:
x = 1
```

```
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
x = 9
```

Primer 22 *Program ilustruje petlju do-while.*

```
#include <stdio.h>

int main()
{
    int x;

    x = 1;
    do
    {
        printf("x = %d\n",x);
        x++; /* x++ je isto kao i x=x+1 */
    }
    while (x<=10);
}
```

Izlaz:

```
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
x = 9
x = 10
```

Primer 23 *Program ilustruje petlju - for.*

```
#include <stdio.h>

int main()
{
    int x;

    for (x = 1; x < 10; x++)
        printf("x = %d\n",x);
}
```

Izlaz:
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
x = 9

Primer 24 *Ispisati prvih 15 članova Fibonačijevog niza.*

```
#include <stdio.h>
#define BROJ 15
main()
{
    int i; /*brojac u petlji */
    int fibonaci[BROJ]; /*niz koji cuva vrednosti iz f-lacije */

    /*inicijalizacije */
    fibonaci[0]=0;
    fibonaci[1]=1;

    /*formiranje vrednosti clana niza u zavisnosti od vrednosti prethodnika */
    for (i=2;i<BROJ;++i) fibonaci[i]=fibonaci[i-2]+fibonaci[i-1];

    /*ispis vrednosti clanova niza */
    for (i=0;i<BROJ;++i)
        printf("%d ", fibonaci[i]);
}
```

Izlaz:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

Primer 25 *Konverzija centimetara u inče - while petlja.*

```
#include <stdio.h>

/* Definicija simbolickih konstanti preko #define direktiva */
/* U fazi pretprocesiranja se vrsi doslovna zamena konstanti
   njihovim vrednostima */

#define POCETAK 0
#define KRAJ 20
#define KORAK 10

int main()
{
    int a;
```

```
a = PO CETAK;
while (a <= KRAJ)
{
    printf("%d cm = %f in\n", a, a/2.54);
    a += KORAK; /* isto sto i a = a + KORAK; */
}
return 0;
}
```

Izlaz:
0 cm = 0.000000 in
10 cm = 3.937008 in
20 cm = 7.874016 in

Primer 26 *Konverzija centimetara u inče - for petlja.*

```
#include <stdio.h>
#define PO CETAK 0
#define KRAJ 20
#define KORAK 10

int main()
{
    int a;
    for (a = PO CETAK; a <= KRAJ; a += KORAK)
        printf("%d cm = %f in\n", a, a/2.54);

    return 0;
}
```

Izlaz:
0 cm = 0.000000 in
10 cm = 3.937008 in
20 cm = 7.874016 in

Primer 27 *Ilustracija switch konstrukcije.*

```
#include<stdio.h>
int main()
{
    int n;
    printf("Unesi paran broj manji od 10\n");
    scanf("%d",&n);
    switch(n) { case 0:
        printf("Uneli ste nulu\n");
        break;
    case 2:
        printf("Uneli ste dvojku\n");
        break;
    case 4:
        printf("Uneli ste cetvorku\n");
        break;
```



```
case 6:
    printf("Uneli ste sesticu\n");
    break;
case 8:
    printf("Uneli ste osmicu\n");
    break;
default:
    printf("Uneli ste nesto sto nije paran broj\n");
}
return 0;
}
```

Ulaz:
Unesi paran broj manji od 10
2
Izlaz:
Uneli ste dvojku

6.4 Funkcije - osnovni pojmovi

Primer 28 *sum* - najjednostavnija funkcija koja sabira dva broja

```
/* Definicija funkcije */
int sum(int a, int b)
{
    return a+b;
}

main()
{
    /* Poziv funkcije */
    printf("%d\n", sum(3,5));
}
```

Primer 29

```
int zbir(int, int);

main()
{
    /* Poziv funkcije */
    printf("%d\n", zbir(3,5));
}

/* Definicija funkcije */
int zbir(int a, int b)
{
    return a+b;
}
```

Primer 30 *power* - funkcija koja stepenuje realan broj na celobrojni izlozilac

```
#include <stdio.h>

/* stepenuje x^k tako sto k puta pomnozi x */
int power(float x, int k)
{
    int i;
    float s = 1;
    for (i = 0; i<k; i++)
        s*=x;

    return s;
}
```

Primer 31 *Verzija koja radi i za negativne izloziocce*

```
int power_n(float x, int k)
{
    int i;
    int negative = k<0;

    if (negative)
        k = -k;

    float s = 1;
    for (i = 0; i<k; i++)
        s*=x;

    return negative ? 1.0/s : s;
}

main()
{
    /* Poziv funkcije */
    float s = power(2.0,8);
    printf("%f\n", s);
}
```

Zadaci za vežbu:

Zadatak 28 *Šta će biti ispisano nakon izvršavanja sledećeg programa?*

```
#include <stdio.h>
main()
{
    int x=506, y=3, z=21, t=2;
    printf("x=%d y=%d\n",x,y);
    printf("z - t=%d\n", z-t);
    printf("z / t =%d\n",z / t);
    printf("-x=%d\n",- x);
}
```

```
    printf("x %% y=%d\n", x/y);  
}
```

Zadatak 29 Napisati program za razmenu vrednosti dva cela broja.

Zadatak 30 Izvršiti štampanje parnih brojeva od 1 do 100 (for, while i do-while).

Zadatak 31 Napisati program koji izračunava sumu i maksimum brojeva koji se unose na standardni ulaz pri čemu je poslednji uneti broj 0 (for, while).

Zadatak 32 Napisati program koji niz celih brojeva veličine 10 popunjava vrednostima kvadrata odgovarajućih indeksa i onda to štampa na ekran(for, while).

Zadatak 33 Napisati program koji ispisuje kvadrate svih brojeva od 5 do 35. Nakon svakog petog kvadrata odštampati znak za novi red(for, while).

Zadatak 34 Ilustracija korišćenja funkcije za izračunavanje faktoriijela celog broja.

(a) Napisati program koji izračunava faktoriijel unetog broja.

(b) Napisati funkciju koja izračunava faktoriijel celog broja.

(c) Napisati program koji izračunava faktoriijel unetog broja koristeći prethodno definisanu funkciju.

Zadatak 35 Napisati program koji izračunava zbir recipročnih vrednosti prvih 10 brojeva.

Zadatak 36 Ilustracija korišćenja funkcije za proveru da li je broj prost.

(a) Napisati program koji za uneti broj proverava da li je prost.

(b) Napisati funkciju koja za ceo broj proverava da li je prost.

(c) Napisati program koji štampa prvih 100 prostih brojeva.

Zadatak 37 Napisati program koji računa n-ti član Fibonačijevog niza.

7

Programski jezik C

1

7.1 Učitavanje i ispis na izlaz - funkcije printf() i scanf()

Primer 32 Šta će biti izlaz iz sledećeg programa?

```
#include <stdio.h>

main()
{
    printf("\"Zdravo, svima\"\n");
    printf("\\n\\tprelazak u novi red\n");
    printf("\\t\\ttabulator\n");
    printf("\\\\kosa crta\n");
    printf("%%\\tprocenat\n");
}
```

Izlaz iz programa:

```
"Zdravo, svima"
\n  prelazak u novi red
\t  tabulator
\\   kosa crta
%%   procenat
```

Primer 33 A šta iz ovog?

```
#include <stdio.h>

main()
{
    putchar('\\');
    putchar('t');
    putchar('\\t');
    printf("Za %d ispisujem %c", '\\', '\\');
}
```

¹Zasnovano na primerima sa sajtova <http://www.matf.bg.ac.yu/~milena>, <http://www.matf.bg.ac.yu/~filip>, <http://www.matf.bg.ac.yu/~jelenagr>.

```
    printf("\n\n\\n\\\n\\\n\n");
}
```

Izlaz iz programa:
\t Za 92 ispisujem \

```
\n\
\\n
```

Primer 34 *A iz ovog?*

```
#include <stdio.h>
main()
{
    printf("Slova:\n%3c\n%5c\n", 'z' , 'Z');
}
```

Izlaz iz programa:
Slova
 z
 Z

Primer 35

```
#include <stdio.h>
main()
{
    int vrednost;
    vrednost='A';
    printf("%s\nkarakter=%3c\nvrednost=%3d\n",
    "Veliko slovo",vrednost,vrednost);
    vrednost='a';
    printf("%s\nkarakter=%3c\nvrednost=%3d\n",
    "Malo",vrednost,vrednost);
}
```

Izlaz (u slucaju ASCII):
Veliko slovo
karakter= A
vrednost= 65
Malo
karakter= a
vrednost= 97

Primer 36 *Program prikazuje unos i ispis realnih brojeva.*

```
#include <stdio.h>
main()
{
    float x;
    printf("Unesi realan broj : ");

    /* Eksperimentisati sa vrednoscu npr. 34.56 */
    scanf("%f",&x);
```

```

    printf("Uneli ste broj (%%f): %f\n", x);
    printf("U naucnoj notaciji (%%g): %g\n", x);
}
Ulaz:
Unesi realan broj: 1.234
Izlaz iz programa:
Uneli ste broj (%%f): 1.234000
U naucnoj notaciji (%%g): 1.234

```

Primer 37 Program prikazuje zaokruživanje realnog broja prilikom ispisa.

```

#include <stdio.h>

main()
{
    float x;
    printf("Unesi realan broj : ");
    scanf("%f",&x);
    printf("Broj zaokruzen na dve decimale je : %.2f\n", x);
}

Ulaz:
Unesi realan broj: 1.234
Izlaz iz programa:
Broj zaokruzen na dve decimale je : 1.23

```

7.2 Operator sizeof

Primer 38 Demonstracija sizeof operatora. sizeof operator izračunava veličinu tipa odnosno promenjive.

```

#include<stdio.h>
main()
{
    int i;
    float f;
    int n[10];

    printf("sizeof(int)=%d\n", sizeof(int));
    printf("sizeof(long)=%d\n", sizeof(long));
    printf("sizeof(short)=%d\n", sizeof(short));
    printf("sizeof(signed)=%d\n", sizeof(signed));
    printf("sizeof(unsigned)=%d\n", sizeof(unsigned));
    printf("sizeof(char)=%d\n", sizeof(char));
    printf("sizeof(float)=%d\n", sizeof(float));
    printf("sizeof(double)=%d\n", sizeof(double));

    printf("sizeof(i)=%d\n", sizeof(i));
    printf("sizeof(f)=%d\n", sizeof(f));
    printf("sizeof(n)=%d\n", sizeof(n));
    printf("Broj elemenata niza n : %d\n", sizeof(n)/sizeof(int));
}

```

```
}

```

Izlaz iz programa(u konkretnom slucaju):

```
sizeof(int)=4
sizeof(long)=4
sizeof(short)=2
sizeof(signed)=4
sizeof(unsigned)=4
sizeof(char)=1
sizeof(float)=4
sizeof(double)=8
sizeof(i)=4
sizeof(f)=4
sizeof(n)=40
Broj elemenata niza n : 10

```

7.3 Obrada teksta sa ulaza - getchar() i putchar()

Primer 39 Program cita jedan karakter i ispisuje ga - demonstracija putchar i getchar.

```
#include <stdio.h>

main()
{
    int  c1, c2;
    c1 = getchar();
    printf("-----\n");
    c2 = getchar();

    printf("c1 = %d, c2 = %d\n",c1, c2);
    printf("c1 = %c, c2 = %c\n",c1, c2);

    putchar(c1); /* isto je kao i printf("%c",c1); */
    putchar(c2); /* isto je kao i printf("%c",c2); */
    putchar('\n');

    /* Za ispisivanje karaktera a */
    putchar('a');
    /* dozvoljeno je : printf("abc"); printf("a"); */
    /* nedozvoljeno je : printf('a'); putchar('abc'); putchar("abc"); */
}

```

Ulaz:

ab

Izlaz:

c1 = 97, c2 = 98

c1 = a, c2 = b

ab

a

Primer 40 Program cita jedan karakter i ispisuje ga - demonstracija `putchar` i `getchar`.

```
#include <stdio.h>

main()
{
    int c;          /* Karakter - obratiti paznju na int */
    c = getchar(); /* cita karakter sa standardnog ulaza */
    putchar(c);     /* pise karakter c na standardni izlaz */

    putchar('\n'); /* prelazak u novi red */
    putchar('a');  /* ispisuje malo a */
    putchar(97);   /* ekvivalentno prethodnom */
}
Ulaz:
s
Izlaz iz programa:
s
s
aa
```

Primer 41 Program prepisuje standardni ulaz na standardni izlaz. Ilustracija redirekcije standardnog ulaza i izlaza : pokrenuti program sa :

```
./a.out <primer.c
./a.out >tekst.txt
./a.out <primer.c >kopija.c

/
#include <stdio.h>

main()
{
    int c;
    /* Obratiti paznju na raspored zagrada */
    while ((c = getchar()) != EOF)
        putchar(c);
}
```

Primer 42 Program vrši prebrojavanje cifara unetih na ulazu.

```
#include <stdio.h>

/* zbog isdigit */
#include <ctype.h>
main()
{
    int c;
    int br_cifara = 0;
    while ((c = getchar()) != EOF)
        if ('0'<=c && c<='9') /* moze i if (isdigit(c)) */
            br_cifara++;
}
```

```
    printf("Broj cifara je : %d\n", br_cifara);  
}
```

Primer 43 Program vrši brojanje pojavljivanja karaktera 0, 1 i 2 (ilustruje switch).

```
#include <stdio.h>  
  
main()  
{  
    int c;  
    int br_0=0, br_1=0, br_2=0;  
  
    while ((c = getchar()) != EOF)  
    {  
        switch(c)  
        {  
            /* Obratiti paznju da nije case 0: niti case '0'; */  
            case '0':  
                br_0++;  
                break; /* Isprobati veziju bez break */  
            case '1':  
                br_1++;  
                break;  
            case '2':  
                br_2++;  
                break;  
        }  
    }  
    printf("Br 0 : %d\nBr 1 : %d\nBr 2 : %d\n",br_0, br_1, br_2);  
}
```

Primer 44 Program broji linije i znakove na ulazu.

```
#include <stdio.h>  
main()  
{  
    int znak; /*prihvata znak sa ulaza */  
    long linije=0 ; /*brojac linija */  
    long br_znak=0; /*brojac znakova na ulazu */  
  
    while ( (znak=getchar() ) != EOF)  
    {  
        br_znak++;  
        if (znak=='\n') linije ++;  
    }  
  
    printf("Prelazaka u novi red: %ld, karaktera: %ld \n",linije,br_znak);  
}
```

Primer 45 Program broji blankove, horizontalne tabulatore i linije na ulazu.

```

#include <stdio.h>
main()
{
    int znak; /*prihvata znak sa ulaza */
    int Blanks=0; /*brojac blankova */
    int Tabs=0; /*brojac horizontalnih tabulatora */
    int NewLines=0; /*brojac linija */

    /*ucitavanje karakter do markera kraja fajla i ...*/
    while( (znak=getchar())!=EOF )
        /* ... brojanje blankova */
        if( znak==' ' ) ++Blanks;
        /* ... brojanje tab-ova */
        else if( znak=='\t' ) ++Tabs;
        else if( znak=='\n' ) ++NewLines;

    /*UOCITI: blok naredbi while ciklus NIJE OGRADJEN viticastim zagradama*/

    /*izdavanje rezultata na standardni izlaz*/
    printf("Blankova: %d. Tabulatora: %d. Prelazaka u novi red: %d\n", Blanks, Tabs, NewLines);
}

```

Primer 46 *Prepisuje ulaz na izlaz čineći tabulatore, nove linije i backslash-ove vidljivim.*

```

#include <stdio.h>
main()
{
    int znak;
    znak=getchar();
    while( znak!=EOF )
    {
        if( znak=='\t' ) /*uciniti tab vidljivim */
        { putchar('\\'); putchar('t'); }
        else if( znak=='\n' ) /*uciniti new line vidljiv */
        { putchar('\\'); putchar('n'); putchar('\n'); }
        else if( znak=='\\' ) /*backslash udvojiti */
        { putchar('\\'); putchar('\\'); }
        else putchar(znak);

        znak=getchar();
    } /* while( znak!=EOF ) */

} /*main() */

```

Primer 47 *Program koji prepisuje ulaz na izlaz pri čemu više blanko znakova zamenjuje jednim.*

```

#include <stdio.h>

```

```
#define GRANICA '0'
main()
{
    int znak; /*tekuci znak sa ulaza*/
    int preth; /*znak koji prethodi tekucem */
    preth=GRANICA;
    while ( (znak=getchar() ) !=EOF)
    {
        if (znak !=' ' || preth != ' ') putchar(znak);
        preth=znak;
    }
}
```

Primer 48 Program proverava da li su zagrade (i) dobro uparene.

```
#include <stdio.h>
#include <stdlib.h>

main()
{
    int c;
    int br_otv = 0;
    while((c=getchar()) != EOF)
    {
        switch(c)
        {
            case '(':
                br_otv++;
                break;
            case ')':
                br_otv--;
                if (br_otv<0)
                {
                    printf("Visak zatvorenih zagrada\n");
                    exit(1);
                }
        }
    }

    if (br_otv == 0)
        printf("Zagrade su u redu\n");
    else
        printf("Visak otvorenih zagrada\n");
}
```

7.4 Nizovi

Primer 49 Program ilustruje korišćenje statičkih nizova. Ispisuje 10 unetih brojeva unazad.

```
#include <stdio.h>

main()
{
    int a[10];
    int i;
    for (i = 0; i<10; i++)
    {   printf("a[%d]=",i);
        scanf("%d",&a[i]);
    }

    printf("Unazad : \n");

    for (i = 9; i>=0; i--)
        printf("a[%d]=%d\n",i,a[i]);
}
```

Primer 50 *Program pronalazi maksimum brojeva sa ulaza - verzija sa nizom.*

```
#include <stdio.h>
#define BR_ELEM 5
main()
{
    int a[BR_ELEM];
    int i;
    int max;

    /* Ucitavamo niz brojeva */
    for (i = 0; i < BR_ELEM; i++)
        scanf("%d",&a[i]);

    /* Pronalazimo maksimum */
    max = a[0];
    for (i = 1; i < BR_ELEM; i++)
        if (a[i]>max)
            max = a[i];

    /* Ispisujemo maksimum */
    printf("Max = %d\n",max);
}
```

Primer 51 *Program pronalazi maksimum brojeva sa ulaza - verzija bez niza.*

```
#include <stdio.h>
#define BR_ELEM 5

main()
{
    int a, max, i;
    scanf("%d",&a);
```

```
    max = a;
    for (i = 1; i < BR_ELEM; i++)
    {
        scanf("%d",&a);
        if (a>max)
            max = a;
    }

    printf("Max : %d\n", max);
}
```

Primer 52 *Brojanje pojavljivanja svake od cifara. Koriscenje niza brojača.*

```
#include <stdio.h>
#include <ctype.h>
main()
{
    /* Niz brojaca za svaku od cifara */
    int br_cifara[10];
    int i, c;

    /* Resetovanje brojaca */
    for (i = 0; i < 10; i++)
        br_cifara[i] = 0;

    /* Citamo sa ulaza i povecavamo odgovarajuce brojace */
    while ((c = getchar()) != EOF)
        if (isdigit(c))
            br_cifara[c-'0']++;

    /* Ispis rezultata */
    for (i = 0; i < 10; i++)
        printf("Cifra %d se pojavila %d put%s\n",
            i, br_cifara[i], br_cifara[i]==1?"":"a");
}
```

Primer 53 *Program ilustruje inicijalizaciju nizova.*

```
#include <stdio.h>

main()
{
    /* Niz inicijalizujemo tako sto mu navodimo vrednosti
       u viticasnim zagradama. Dimenzija niza se odredjuje
       na osnovu broja inicijalizatora */
    int a[] = {1, 2, 3, 4, 5, 6};

    /* Isto vazi i za niske karaktera */
    char s[] = {'a', 'b', 'c'};
```

```
/* Ekvivalentno prethodnom bi bilo
char s[] = {97, 98, 99};
*/

/* Broj elemenata niza */
int a_br_elem = sizeof(a)/sizeof(int);
int s_br_elem = sizeof(s)/sizeof(char);

/* Ispisujemo nizove */

int i;
for (i = 0; i < a_br_elem; i++)
    printf("a[%d]=%d\n", i, a[i]);

for (i = 0; i < s_br_elem; i++)
    printf("s[%d]=%c\n", i, s[i]);

}
```

7.4.1 Stringovi

Primer 54 Program uvodi niske karaktera terminisane nulom.

```
#include <stdio.h>

main()
{
    /* Poslednji bajt niske karaktera se postavlja na '\0' tj. 0 */
    char s[] = {'a', 'b', 'c', '\0' };

    /* Kraci nacin da se postigne prethodno */
    char t[] = "abc";

    /* Ispis niske s karakter po karakter*/
    int i;
    for (i = 0; s[i] != '\0'; i++)
        putchar(s[i]);
    putchar('\n');

    /* Ispis niske s koristeći funkciju printf */
    printf("%s\n", s);

    /* Ispis niske t karakter po karakter*/
    for (i = 0; t[i] != '\0'; i++)
        putchar(t[i]);
    putchar('\n');

    /* Ispis niske t koristeći funkciju printf */
    printf("%s\n", t);
}
```

}

7.4.2 Prenos niza u funkciju

8

Programski jezik C

1

8.1 Obrada teksta sa ulaza

Primer 55 *Program ispisuje ascii tabelu.*

```
#include <stdio.h>

main()
{
    int c;
    for (c = 0; c<128; c++)
        printf("%d - %c\n",c,c);
}
```

Primer 56 *Program broji reči sa ulaza.*

```
#include<stdio.h>
#define NIJE_U_TOKU_CITANJE_RECI 1
#define JESTE_U_TOKU_CITANJE_RECI 2

main()
{   int c;
    int br_reci = 0;
    int stanje = NIJE_U_TOKU_CITANJE_RECI;

    while ((c=getchar())!=EOF)
    {
        if (isalpha(c))
        {
            if (stanje == NIJE_U_TOKU_CITANJE_RECI)
            {
                stanje = JESTE_U_TOKU_CITANJE_RECI;
            }
        }
    }
}
```

¹Zasnovano na primerima sa sajtova <http://www.matf.bg.ac.yu/~milena>, <http://www.matf.bg.ac.yu/~filip>, <http://www.matf.bg.ac.yu/~jelenagr>.

```

        br_reci ++;
    }
}
else if (isspace(c))
    stanje = NIJE_U_TOKU_CITANJE_RECII;
}

printf("Broj reci je : %d\n", br_reci);
}

```

Primer 57 *Program menja mesta cifara u broju.*

```

#include <stdio.h>
main(){
    int n,t=0;
    printf("Unesite broj\n");
    scanf("%d",&n);
    do
    { t=t*10+n%10;
      n/=10;
    }
    while(n);
    printf("Novi broj je %d\n", t);
    return 0;
}

```

Izlaz:
 Unesite broj
 1234
 Novi broj je 4321

8.2 Operator ,

Primer 58

```

#include <stdio.h>
void Zbir_Kvad(int n); /*f-ja koja vrsi zeljeno izracunavanje */
main()
{
    Zbir_Kvad( 5);
    Zbir_Kvad( 23);
}
void Zbir_Kvad(int n)
{
    int br; /* lokalna promenljiva funkcije, brojac u ciklusu */
    long Zbir=0; /* lokalna promenljiva funkcije, suma kvadrata brojeva od 1..n */
    for (br=1; br<=n; Zbir+= (long) br*br, ++br) ;
    printf(" Zbir kvadrata brojeva od 1 do %d jese %ld\n", n,Zbir);
}

```

Izlaz:

Zbir kvadrata brojeva od 1 do 5 jese 55
 Zbir kvadrata brojeva od 1 do 23 jese 4324

8.3 Uslovni izrazi

Slično kao if.

Primer 59

```
izraz1 ? izraz2 : izraz3

z=(a<b)? a : b; /*z=min(a,b)*/ max = (a>b)? a : b;
```

8.4 Oblast važenja lokalnih promenljivih

Primer 60 #include <stdio.h>

```
main()
{
    int pom=1;
    printf("Pre ulaska u unutrasnji blok pom=%d\n",pom);
    {
        int pom=50;
        printf("Pre izlaska iz unutrasnjeg bloka pom=%d\n",pom);
    }
    printf("Nakon izlaska iz unutrasnjeg bloka pom=%d\n",pom);
}
```

Izlaz:
 Pre ulaska u unutrasnji blok pom=1
 Pre izlaska iz unutrasnjeg bloka pom=50
 Nakon izlaska iz unutrasnjeg bloka pom=1

8.5 Lenjo izračunavanje

Primer 61 Ilustracija lenjog izračunavanja logičkih operatora.

Prilikom izračunavanja izraza - A && B, ukoliko je A netačno, izraz B se ne izračunava.
 Prilikom izračunavanja izraza - A || B, ukoliko je A tačno, izraz B se ne izračunava.

```
#include <stdio.h>

int b = 0;

/* Funkcija ispisuje da je pozvana i uvecava promenjivu b.
   Funkcija uvek vraca vrednost 1 (tacno)
*/
int izracunaj()
{
    printf("Pozvano izracunaj()\n");
    b++;
    return 1;
}
```

```

main()
{
    /* Funkcija izracunaj() ce se pozivati samo za parne vrednosti a */
    int a;
    for (a = 0; a < 10; a++)
        if (a%2 == 0 && izracunaj())
            printf("Uslov ispunjen : a = %d, b = %d\n", a, b);
        else
            printf("Uslov nije ispunjen : a = %d, b = %d\n", a, b);

    printf("-----\n");

    /* Funkcija izracunaj() ce se pozivati samo za neparne vrednosti a */
    b = 0;
    for (a = 0; a < 10; a++)
        if (a%2 == 0 || izracunaj())
            printf("Uslov ispunjen : a = %d, b = %d\n", a, b);
        else
            printf("Uslov nije ispunjen : a = %d, b = %d\n", a, b);
}

```

Izlaz:

```

Pozvano izracunaj()
Uslov ispunjen : a = 0, b = 1
Uslov nije ispunjen : a = 1, b = 1
Pozvano izracunaj()
Uslov ispunjen : a = 2, b = 2
Uslov nije ispunjen : a = 3, b = 2
Pozvano izracunaj()
Uslov ispunjen : a = 4, b = 3
Uslov nije ispunjen : a = 5, b = 3
Pozvano izracunaj()
Uslov ispunjen : a = 6, b = 4
Uslov nije ispunjen : a = 7, b = 4
Pozvano izracunaj()
Uslov ispunjen : a = 8, b = 5
Uslov nije ispunjen : a = 9, b = 5
-----
Uslov ispunjen : a = 0, b = 0
Pozvano izracunaj()
Uslov ispunjen : a = 1, b = 1
Uslov ispunjen : a = 2, b = 1
Pozvano izracunaj()
Uslov ispunjen : a = 3, b = 2
Uslov ispunjen : a = 4, b = 2
Pozvano izracunaj()
Uslov ispunjen : a = 5, b = 3
Uslov ispunjen : a = 6, b = 3
Pozvano izracunaj()
Uslov ispunjen : a = 7, b = 4
Uslov ispunjen : a = 8, b = 4

```

Pozvano izracunaj()
 Uslov ispunjen : a = 9, b = 5

8.6 Nizovi

Primer 62 *Napisati program u C-u koji prikazuje sve proste brojeve u datom intervalu kojima je zbir cifara složen broj. Interval se zadaje učitavanjem gornje i donje granice (dva prirodna broja). Brojeve prikazati u opadajućem poretku.*

```
#include <stdio.h>
#include <stdlib.h>

int prost (int n); /*testira da li je broj n prost broj */
/*Prirodni brojevi (sem 1)imaju najmanje dva delioca:jedinicu i samog sebe.
Brojevi koji nemaju drugih delioca,sem ova dva, nazivaju se prostim */

int zbirCifara (int n); /*vraca zbir cifara broja n */
main()
{
    int donja,gornja; /*granice intervala */
    int i; /*brojac u petlji */
    int pom; /*posrednik u eventualnoj zameni */

    /*ucitavanja granice intervala */
    scanf("%d%d", &donja, &gornja);
    if (donja > gornja) /*obezbedjivanje relacije: donja <=gornja */
    {
        pom=donja;
        donja=gornja;
        gornja=pom;
    }
    for(i=gornja;i>=donja; i--)
        if (prost (i) && !prost(zbirCifara(i) ) ) printf("%d\n",i);
}

int prost(int n)
/*Ispituje se da li je broj n prost tako to se proverava da li ima delioce
medju brojevima od 2 do n/2. Pri implementaciji se koristi tvrdjenje da je
broj prost ako je jednak 2, ili ako je neparan i ako nema delitelja medju
neparnim brojevima od 3 do n/2 */
{
    int prost; /*indikator slozenosti broja n */
    int i; /*potencijalni delitelj broja n */
    if (n==1) return 0;
    /*parni brojevi razliciti od od dva nisu prosti brojevi */
    prost= (n%2!=0) || (n==2);

    /*najmanji potencijalni kandidat za delitelje medju
    neparnim brojevima razlicitim od jedan */
    i=3;
    while ( (prost) && (i<=n/2) )
```

```

    {
        prost=n%i != 0;
        i=i+2; /*proveravamo kandidate za delitelje samo medju neparnim brojevma */
    }
    return prost;
}
int zbirCifara (int n)
{ int Suma=0;
  while (n>0)
  {
      Suma+= n%10; /*dodavanje cifre tekuceg razreda,pocev od razreda jedinica ,
                  a iduci ka visim razredima cifara */
      n=n/10;      /*prelaz ka visem razredu */
  }
  return Suma;
}

```

Ulaz:

1 20

Izlaz:

19

17

13

8.6.1 Nizovi-prenos u funkciju

Primer 63 *Funkcija za ispis niske karaktera - demonstrira prenos niske karaktera u funkciju.*

```
#include <stdio.h>
```

```
/* Uz nisku karaktera nije potrebno prenositi dimenziju ukoliko se
po\v stuje dogovor da se svaka niska zavr\v sava karakterom '\0'.*/
```

```
void print_string(char s[])
{
    int i;
    for (i = 0; s[i]; i++)
        putchar(s[i]);
}

```

```
main()
{
    print_string("Zdravo\n");
}

```

Izlaz:

Zdravo

Primer 64 *Program izračunava skalarni proizvod dva niza celih brojeva.*

```
#include <stdio.h>
long mnozi(int x[],int y[],int n);
main()

```

```
{
    int a[]={1,2,3,4,5,6}, b[]={8,7,6,5,4,3};
    printf("Skalarno a*b= %ld\n",mnozi(a,b,6));
}
```

```
long mnozi(int x[],int y[],int n)
{
    int br;
    long suma=0;
    for(br=0;br<n;br++) suma=suma+x[br]*y[br];
    return suma;
}
```

Izlaz:
Skalarno a*b= 98

Primer 65 Program izračunava vrednost polinoma(1.način).

```
/*polinom 1 nachin*/
#include <stdio.h>
float stepen(float, int);
main()
{
    float x,P=0,a[100];
    int n,i;
    printf("unesite stepen\n");
    scanf("%d",&n);
    for (i=0;i<=n;i++)
    {
        printf("unesite a[%d]\n",i);
        scanf("%f",&a[i]);
    }
    printf("unesite x\n");
    scanf("%f",&x);
    for (i=0;i<=n;i++) P=P+a[i]*stepen(x,i);
    printf("%f",P);
}
```

```
float stepen(float a ,int n)
{
    int i;
    float p=1;
    for (i=0;i < n;i++) p*=a;
    return p;
}
```

Ulaz:
unesite stepen
2
unesite a[0]
1
unesite a[1]
2
unesite a[2]

```

3
unesite x
4
Izlaz:
57.000000

```

Primer 66 Program izračunava vrednost polinoma (2.način).

```

#include <stdio.h>
main()
{
    int n,i;
    float
    a[100],x,P;
    printf("Unesite najveci stepen: ");
    scanf("%d",&n);
    for (i=0;i<=n;i++)
    {
        printf("Unesite a[%d]: ",i);
        scanf("%f",&a[i]);
    }
    printf("Unesite X: ");
    scanf("%f",&x);
    P=0;
    for (i=n;i>=0;i--)
    {
        P=P*x+a[i];
    }
    printf("Vrednost polinoma je: %f\n",P);
}

```

```

Ulaz:
Unesite najveci stepen: 2
Unesite a[0]: 1
Unesite a[1]: 2
Unesite a[2]: 3
Unesite X: 4

```

```

Izlaz:
Vrednost polinoma je: 57.000000

```

8.7 Break i continue

Primer 67 Uklanja beline, tabulatore ili znak za kraj reda sa kraja stringa

```

int trim(char s[])
{
    int n;
    for (n = strlen(s)-1; n >= 0; n--)
        if (s[n] != ' ' && s[n] != '\t' && s[n] != '\n')
            break;
    s[n+1] = '\0';
    return n;
}

```



```
}
```

Continue se ređe koristi, on prouzrokuje da se pređe na sledeću iteraciju u petlji.

Primer 68

```
for(i=0; i<n; i++)
{
    if (a[i]==0) continue;
    ...
    /* obradi pozitivne elemente nekako*/ } }
```

8.8 Prenos parametara po vrednosti

Primer 69 #include <stdio.h>

```
void f(int a)
{
    a++;
    printf("Vrednost promenjive a je : %d\n",a);
}

main()
{
    int b = 3;
    f(b);
    printf("Vrednost promenjive b je : %d\n",b);
}
```

Izlaz:

```
Vrednost promenjive a je : 4
Vrednost promenjive b je : 3
```

Primer 70 *Demonstracija prenosa parametara po vrednosti - preneti parametri se ne mogu menjati*

```
#include <stdio.h>

/* Funkcija pokusava da ucita ceo broj.
   Zbog prenosa parametara po vrednosti ovo ne uspeva */
void get_num(int a) {
    scanf("%d", &a);
    printf("a = %d\n", a);
}

main() {
    int x = 0;
    printf("Unesi broj : ");
    get_num(x);
    printf("x = %d\n", x);
}
```

Ulaz:

```
Unesi broj : 5
Izlaz iz programa:
a = 5
x = 0
```

Primer 71 *Demonstrira prenos nizova u funkciju - preneti niz se može menjati.*

```
#include <stdio.h>
#include <ctype.h> /* Zbog funkcije isspace */

/* Funkcija učitava rec sa standardnog ulaza i smesta je u niz
karaktera s. Ovo uspeva zbog toga sto se po vrednosti prenosi
adresa pocetka niza, a ne ceo niz */

void get_word(char s[])
{
    int c, i = 0;
    while (!isspace(c=getchar()))
        s[i++] = c;
    s[i] = '\0';
}

main()
{
    /* Obavezno je alocirati memoriju za niz karaktera */
    char s[100];
    get_word(s);
    printf("%s\n", s);
}
```

```
Ulaz:
Ovo je jedna recenica.
Izlaz:
Ovo
```

Zadatak 38 *Broj je Nivenov ako je deljiv sumom svojih cifara. Napisati program koji za uneto n ispisuje prvih n Nivenovih brojeva.*

9

Programski jezik C

1

9.1 Ugnježdjena petlja

Primer 72 *Ilustracija dve ugnježdjene petlje.*

```
#include<stdio.h>
int main()
{
    int i,j;

    for(i=1; i<=3; i++)
    {
        for(j=1; j<=3; j++)
            printf("%d * %d = %d\t", i, j, i*j);
        printf("\n");
    }
}
```

Izlaz:

```
1 * 1 = 1      1 * 2 = 2      1 * 3 = 3
2 * 1 = 2      2 * 2 = 4      2 * 3 = 6
3 * 1 = 3      3 * 2 = 6      3 * 3 = 9
```

Primer 73 *Napisati funkciju koja izračunava zbir n -tih stepena brojeva od 1 do granice i program koji ilustruje rad ove funkcije.*

```
#include <stdio.h>
void Zbir_stepena (int n, int granica);
main()
{
    Zbir_stepena(2,5);
    Zbir_stepena(3,5);
    Zbir_stepena(4,10);
}
```

¹Zasnovano na primerima sa sajtova <http://www.matf.bg.ac.yu/~filip>, <http://www.matf.bg.ac.yu/~milena>, <http://www.matf.bg.ac.yu/~jelenagr>.

```

    return 0;
}

void Zbir_stepena (int n, int granica)
{
    int i,j;    /*brojaci u for petljama */
    long Zbir=0 ,   stepenovan ;

    /*spoljasnji for ciklus obavlja sumiranja*/
    for (i=1; i<=granica; Zbir +=stepenovan, ++i)
        /*unutrasnji for ciklus obavlja stepenovanje */
        for( stepenovan=1,j=1; j<=n; stepenovan*= (long) i, ++j) ;
    printf(" Zbir %d. stepena od 1 do %d jeste %ld\n", n,granica,Zbir);
}

```

Izlaz:

```

Zbir 2. stepena od 1 do 5 jeste 55
Zbir 3. stepena od 1 do 5 jeste 225
Zbir 4. stepena od 1 do 10 jeste 25333

```

9.2 F-je za rad sa stringovima

Primer 74 `string_reverse` - *obrće nisku karaktera.*

```

#include <stdio.h>

/* Zbog funkcije strlen */
#include <string.h>

/* Ova funkcija racuna duzinu date niske karaktera.
   Umesto nje, moguće je koristiti standardnu funkciju strlen .
   */
int string_length(char s[]) {
    int i;
    for (i = 0; s[i]; i++)
        ;

    return i;
}

/* Funkcija obrće nisku karaktera */
void string_reverse(char s[])
{
    int i, j;
    for (i = 0, j = string_length(s)-1; i<j; i++, j--)
    {
        int tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
    }
}

```

```

    /* Napomena : razlikovati prethodnu petlju od dve ugnjezdjene petlje:
    for ( i = 0; ....)
        for ( j = duzina(s)-1; ...
        */
}

main() {
    char s[] = "Zdravo svima";
    string_reverse(s);
    printf("%s\n", s);
}

```

Izlaz:
amivs ovardZ

Primer 75 strlen, strcpy, strcat, strcmp, strchr, strstr - *manipulacija niskama karaktera.*
Vezbe radi, implementirane su funkcije biblioteke string.h

```

#include <stdio.h>

/* Izracunava duzinu stringa */
int string_length(char s[]) {
    int i;
    for (i = 0; s[i]; i++)
        ;
    return i;
}

/* Kopira string src u string dest.
   Pretpostavlja da u dest ima dovoljno prostora. */
void string_copy(char dest[], char src[]) {
    /* Kopira karakter po karakter, sve dok nije iskopiran karakter '\0' */
    int i;
    for (i = 0; (dest[i]=src[i]) != '\0'; i++)
        ;

    /* Uslov != '\0' se, naravno, moze izostaviti :

    for (i = 0; dest[i]=src[i]; i++)
        ;
    */
}

/* Nadovezuje string t na kraj stringa s.
   Pretpostavlja da u s ima dovoljno prostora. */
void string_concatenate(char s[], char t[]) {
    int i, j;
    /* Pronalazimo kraj stringa s */
    for (i = 0; s[i]; i++)
        ;
}

```

```

    /* Vrsi se kopiranje, slicno funkciji string_copy */
    for (j = 0; s[i] = t[j]; j++, i++)
        ;
}

/* Vrsi leksikografsko poredjenje dva stringa.
   Vraca :
       0 - ukoliko su stringovi jednaki
       <0 - ukoliko je s leksikografski ispred t
       >0 - ukoliko je s leksikografski iza t
*/ int string_compare(char s[], char t[]) {
    /* Petlja tece sve dok ne naidjemo na prvi razliciti karakter */
    int i;
    for (i = 0; s[i]==t[i]; i++)
        if (s[i] == '\0') /* Naisli smo na kraj oba stringa,
                           a nismo nasli razliku */
            return 0;

    /* s[i] i t[i] su prvi karakteri u kojima se niske razlikuju.
       Na osnovu njihovog odnosa, odredjuje se odnos stringova */
    return s[i] - t[i];
}

/* Pronalazi prvu poziciju karaktera c u stringu s, odnosno -1
   ukoliko s ne sadrzi c */
int string_char(char s[], char c) {
    int i;
    for (i = 0; s[i]; i++)
        if (s[i] == c)
            return i;
    /* nikako
       else
            return -1;
    */
    /* Nije nadjeno */
    return -1;
}

/* Pronalazi poslednju poziciju karaktera c u stringu s, odnosno
   -1
   ukoliko s ne sadrzi c */
int string_last_char(char s[], char c) {
    /* Pronalazimo kraj stringa s */
    int i;
    for (i = 0; s[i]; i++)
        ;

    /* Krecemo od kraja i trazimo c unazad */
    for (i--; i>=0; i--)
        if (s[i] == c)
            return i;

```

```
/* Nije nadjeno */
return -1;

/*
Koristeci string_length :

for (i = string_length(s) - 1; i>0; i--)
    if (s[i] == c)
        return i;

return -1;
*/
}

/* Proverava da li string str sadrzi string sub.
Vraca poziciju na kojoj sub pocinje, odnosno -1 ukoliko ga nema
*/ int string_string(char str[], char sub[]) {
    int i, j;
    /* Proveravamo da li sub pocinje na svakoj poziciji i */
    for (i = 0; str[i]; i++)
        /* Poredimo sub sa str pocevsi od poziciji i
        sve dok ne naidjemo na razliku */
        for (j = 0; str[i+j] == sub[j]; j++)
            /* Nismo naisli na razliku a ispitali smo
            sve karaktere niske sub */
            if (sub[j+1] == '\0')
                return i;
    /* Nije nadjeno */
    return -1;
}

main() {
    char s[100];
    char t[] = "Zdravo";
    char u[] = " svima";

    string_copy(s, t);
    printf("%s\n", s);

    string_concatenate(s, u);
    printf("%s\n", s);

    printf("%d\n", string_char("racunari", 'n'));
    printf("%d\n", string_last_char("racunari", 'a'));

    printf("%d\n", string_string("racunari", "rac"));
    printf("%d\n", string_string("racunari", "ari"));
    printf("%d\n", string_string("racunari", "cun"));
    printf("%d\n", string_string("racunari", "cna"));
}
```

Izlaz:

Zdravo

Zdravo svima

4

5

0

5

2

-1

10

Programski jezik C

1

10.1 Konverzije

Primer 76 `atoi` - konverzija niske cifara u brojnu vrednost. Program omogućuje čitanje celog broja bez korišćenja funkcije `scanf("%d")`.

Napomena : funkcija `atoi` već postoji u zaglavlju `<stdio.h>` i ovde je implementirana samo za vežbu.

```
#include <stdio.h>
#include <ctype.h> /* Zbog funkcije isspace */

/* Maksimalna duzina reci */
#define MAX_DIGITS 10

/* AsciiToInteger
   Funkcija izracunava vrednost celog broja koji je zapisan u datom
   nizu karaktera. Za izracunavanje se koristi Hornerova shema. */

int atoi(char s[]) {
    int sum = 0;

    /* Obradjuju se karakteri sve dok su cifre */
    int i;
    for (i = 0; isdigit(s[i]); i++)
        sum = 10*sum + (s[i]-'0');

    return sum;
}

main() {
    char s[MAX_DIGITS];
    int num;
```

¹Zasnovano na primerima sa sajtova <http://www.matf.bg.ac.yu/~filip>, <http://www.matf.bg.ac.yu/~milena>, <http://www.matf.bg.ac.yu/~jelenagr>.

```

/* Ucitavamo broj u vidu niske karaktera */
printf("Unesi ceo broj : ");

/* Obratiti paznju da uz s ne stoji & */
scanf("%s", s);

/* Izracunava se njegova vrednost */
num = atoi(s);

/* Ispisuje se vrednost */
printf("Uneo si broj : %d\n", num);
printf("Broj za tri veci je : %d\n", num+3);
}

```

```

Ulaz:
Unesi ceo broj : 5
Izlaz:
Uneo si broj : 5
Broj veci za tri je : 8

```

Primer 77 *btoi - konverzija iz datog brojnog sistema u dekadni.*

```

#include <stdio.h>
#include <ctype.h>

/* Pomocna funkcija koja izracunava vrednost koju predstavlja
karakter u datoj osnovi
Funkcija vraca -1 ukoliko cifra nije validna.

Npr.
cifra 'B' u osnovi 16 ima vrednost 11
cifra '8' nije validna u osnovi 6

*/

int digit_value(char c, int base) {
    /* Proveravamo obicne cifre */
    if (isdigit(c) && c < '0'+base)
        return c-'0';

    /* Proveravamo slovne cifre za mala slova */
    if ('a'<=c && c < 'a'+base-10)
        return c-'a'+10;

    /* Proveravamo slovne cifre za velika slova */
    if ('A'<=c && c < 'A'+base-10)
        return c-'A'+10;

    return -1;
}

```

```

/* Funkcija izracunava vrednost celog broja koji je zapisan u
datum
nizu karaktera u datoj osnovi. Za izracunavanje se koristi Hornerova shema.
*/ int btoi(char s[], int base) {
    int sum = 0;

    /* Obradjuju se karakteri sve dok su cifre */
    int i, vr;
    for (i = 0; (vr = digit_value(s[i], base)) != -1; i++)
        sum = base*sum + vr;

    return sum;
}

main() {
    char bin[] = "11110000";
    char hex[] = "FF";

    printf("Dekadna vrednost binarnog broja %s je %d\n", bin, btoi(bin, 2));
    printf("Dekadna vrednost heksadekadnog broja %s je %d\n", hex, btoi(hex, 16));
}

```

Izlaz:

Dekadna vrednost binarnog broja 11110000 je 240

Dekadna vrednost heksadekadnog broja FF je 255

10.2 Makroi

Primer 78 Šta je rezultat rada sledeceg programa :

```

#include <stdio.h>
#define IN 1
#define OUT 0
main( )
{ int c, nw , state;
  state=OUT;
  nw=0;
  while ( (c=getchar() ) != EOF)
  {
    if (c==' ' || c=='\n' || c=='\t') state=OUT;
    else if (state==OUT) {
      state=IN;
      ++nw;
    }
  }
  printf("\n%d\n",nw);
}

```

Rezultat:

U sledecem novom redu se ispisuje broj reci teksta sa ulaza, pa se predje u novi red. Smatramo da rec je ma koji niz znakova koji ne sadri blanko, tab, new line.

Primer 79 *Demonstracija pretprocesorske direktive #define*

```
#include<stdio.h>
/* Racuna sumu dva broja */
#define sum(a,b) ((a)+(b))

/* Racuna kvadrat broja - pogresna verzija */
#define square_w(a) a*a

/* Racuna kvadrat broja */
#define square(a) ((a)*(a))

/* Racuna minimum tri broja */
#define min(a, b, c) (a)<(b) ? ((a)<(c) ? (a) : (c)) : ((b)<(c) ? (b) : (c))

main()
{
    printf("sum(3,5) = %d\n", sum(3,5));
    printf("square_w(5) = %d\n", square_w(5));
    printf("square_w(3+2) = %d\n", square_w(3+2));
    printf("square(3+2) = %d\n", square(3+2));
    printf("min(1,2,3) = %d\n", min(1,2,3));
    printf("min(1,3,2) = %d\n", min(1,3,2));
    printf("min(2,1,3) = %d\n", min(2,1,3));
    printf("min(2,3,1) = %d\n", min(2,3,1));
    printf("min(3,1,2) = %d\n", min(3,1,2));
    printf("min(3,2,1) = %d\n", min(3,2,1));
}
```

Izlaz iz programa:

```
sum(3,5) = 8
square_w(5) = 25
square_w(3+2) = 11
square(3+2) = 25
min(1,2,3) = 1
min(1,3,2) = 1
min(2,1,3) = 1
min(2,3,1) = 1
min(3,1,2) = 1
min(3,2,1) = 1
```

Primer 80 *Demonstracija pretprocesorske direktive #define*

```
#include <stdio.h>
#define KUBW(a) (a * a * a)
#define KUB(a)  ( (a) * (a) * (a) )

main()
```

```
{

int b=1;

printf("KUB(%d) = %d\n", 2*b+4, KUBW(2*b+4) );
printf("KUB(%d) = %d\n", 2*b+4, KUB(2*b+4) );

}
```

```
Izlaz:
KUB(6) = 22
KUB(6) = 216
```

Primer 81 *Ilustracija beskonačne petlje:*

```
#define forever for(;;);
```

Moguće je definisati makroe sa argumentima tako da tekst zamene bude različit za različita pojavljivanja makroa.

Primer 82

```
#define max(A, B) ((A)>(B) ? (A) : (B))
```

na osnovu ovoga će linija

```
x=max(p+q, r+s)
```

biti zamenjena linijom

```
x=((p+q) > (r+s) ? (p+q) : (r+s));
```

Treba voditi računa o sporednim efektima. Sledeća linija koda prouzrokuje uvećanje vrednosti i i j za dva.

```
max(i++, j++)
```

Takođe treba voditi računa o zagradama. Sledeći makro prouzrokuje neočekivane rezultate za poziv square(a+1)

```
#define square(x) x*x
```

Primer 83

```
#include <stdio.h>
#define max1(x,y) (x>y?x:y)
#define max2(x,y) ((x)>(y)?(x):(y))
#define swapint(x,y) { int z; z=x; x=y; y=z; }
#define swap(t,x,y) { \
    t z; \
    z=x; \
    x=y; \
    y=z; }
```

```
main()
{
```

```
int x=2,y=3;
```

```
printf( "max1(x,y) = %d\n", max1(x,y) );
/* max1(x,y) = 3 */

/* Zamena makroom se ne vrsi
unutar niski pod navodnicima*/
printf( "max1(x=5,y) = %d\n", max1(x,y) );
/* max1(x=5,y) = 3 */

printf( "max1(x++,y++) = %d\n", max1(x++,y++) );
/* max1(x++,y++) = 4 */

printf( "x = %d, y = %d\n", x, y );
/* x = 3, y = 5 */

swapint(x,y);

printf( "x = %d, y = %d\n", x, y );
/* x = 5, y = 3 */

swap(int,x,y);
printf( "x = %d, y = %d\n", x, y );
/* x = 3, y = 5 */
}

Izlaz:
max1(x,y) = 3
max1(x=5,y) = 3
max1(x++,y++) = 4
x = 3, y = 5
x = 5, y = 3
x = 3, y = 5
```

11

Programski jezik C

1

11.1 Bit-operatori

!!!Ne mešati sa logičkim operatorima!!!

& bitsko AND
| bitsko OR
^ bitsko ekskluzivno OR
<< levo pomeranje
>> desno pomeranje
~ jedinичni komplement

Primer 84 *Demonstracija bitskih operatora*

```
#include <stdio.h>

main()
{ printf("%o %o\n",255,15);
  printf( "255 & 15 = %d\n", 255 & 15 );
  printf( "255 | 15 = %d\n", 255 | 15 );
  printf( "255 ^ 15 = %d\n", 255 ^ 15 );
  printf( "2 << 2   = %d\n", 2 << 2 );
  printf( "16 >> 2  = %d\n", 16 >> 2 );
}
```

Izlaz iz programa je:

```
377 17
255 & 15 = 15
255 | 15 = 255
255 ^ 15 = 240
2 << 2   = 8
16 >> 2  = 4
```

Primer 85 *print_bits - stampa bitove u zapisu datog celog broja x.*

¹Zasnovano na primerima sa sajtova <http://www.matf.bg.ac.yu/~filip>, <http://www.matf.bg.ac.yu/~milena>, <http://www.matf.bg.ac.yu/~jelenagr>.


```

#include <stdio.h>

void print_bits(int x);

main(){
    int n,k;
    printf("Unesite broj i poziciju tog broja koju zelite da proverite:\n");
    scanf("%d %d",&n,&k);
    printf("Binarno, une\v seni broj je\n");
    print_bits(n);
    printf("Novi broj je %d\n",(n |(1<<k)));
    printf("Binarno, novi broj je\n");
    print_bits((n |(1<<k)));
    return 0;
}

```

Izrazom $a \gg b$ vrši se pomeranje sadržaja operanda a predstavljenog u binarnom obliku za b mesta u desno. Popunjavanje upraznjenih mesta na levoj strani zavisi od tipa podataka i vrste računara. Ako se pomeranje primenjuje nad operandom tipa unsigned popunjavanje je nulama. Ako se radi o označenom operandu popunjavanje je jedinicama kada je u krajnjem levom bitu jedinica, a nulama kada je u krajnjem levom bitu nula.

Primer 88 *Funkcija koja broji bitove postavljene na 1 u broju*

```

int bitcount(unsigned x)
{
    int b;
    for(b=0; x!=0; x>>=1)
        if (x & 01) b++;
    return b;
}

```

Primer 89 *sum_of_bits - izračunava sumu bitova datog neoznačenog broja.*

```

#include <stdio.h>

/* Pomocna funkcija - stampa bitove neoznacnog broja */
void print_bits(unsigned x)
{
    int wl = sizeof(unsigned)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');

    putchar('\n');
}

/*
int sum_of_bits(unsigned x)
{

```

```

    int wl = sizeof(unsigned)*8;
    int br = 0;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask>>=1)
        if (x&mask)
            br++;

    return br;
}

*/

/* Efikasnija verzija */
int sum_of_bits(unsigned x)
{
    int br;
    for (br = 0; x; x>>=1)
        if (x&1)
            br++;

    return br;
}

main()
{
    printf("Binarni zapis broja 127 je\n");
    print_bits(127);
    printf("Suma bitova broja 127 je %d\n",sum_of_bits(127));
    printf("Binarni zapis broja 128 je\n");
    print_bits(128);
    printf("Suma bitova broja 128 je %d\n",sum_of_bits(128));
    printf("Binarni zapis broja 0x00FF00FF je\n");
    print_bits(0x00FF00FF);
    printf("Suma bitova broja 0x00FF00FF je %d\n",sum_of_bits(0x00FF00FF));
    printf("Binarni zapis broja 0xFFFFFFFF je\n");
    print_bits(0xFFFFFFFF);
    printf("Suma bitova broja 0xFFFFFFFF je %d\n",sum_of_bits(0xFFFFFFFF));
}

```

Primer 90 *get_bits, set_bits, invert_bits - izdvajanje, postavljanje i invertovanje pojedinačnih bitova*

```

#include <stdio.h>

/* Pomocna funkcija - stampa bitove neoznacnog broja */
void print_bits(unsigned x)
{
    int wl = sizeof(unsigned)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');
}

```

```

    putchar('\n');
}

/* Funkcija vraca n bitova broja x koji pocinju na poziciji p */
unsigned get_bits(unsigned x, int p, int n)
{
    /* Gradimo masku koja ima poslednjih n jedinica
       0000000...00011111
       tako sto sve jedinice ~0 pomerimo u levo za n mesta
       1111111...1100000
       a zatim komplementiramo
    */
    unsigned last_n_1 = ~(~0 << n);

    /* x pomerimo u desno za odgovarajuci broj mesta, a zatim
       konjunkcijom sa konstruisanom maskom obrisemo pocetne cifre */

    return (x >> p+1-n) & last_n_1;
}

/* Funkcija vraca modifikovano x tako sto mu je izmenjeno n bitova
   pocevsi od pozicije p i na ta mesta je upisano poslednjih n bitova
   broja y */
unsigned set_bits(unsigned x, int p, int n, unsigned y)
{
    /* Maska 000000...000111111 - poslednjih n jedinica */
    unsigned last_n_1 = ~(~0 << n);

    /* Maska 1111100...000111111 - n nula pocevsi od pozicije p */
    unsigned middle_n_0 = ~(last_n_1 << p+1-n);

    /* Brisemo n bitova pocevsi od pozicije p */
    x = x & middle_n_0;

    /* Izdvajamo poslednjih n bitova broja y i pomeramo ih na poziciju p */
    y = (y & last_n_1) << p+1-n;

    /* Upisujemo bitove broja y u broj x i vracamo rezultat */
    return x | y;
}

/* Invertuje n bitova broja x pocevsi od pozicije p */
unsigned invert_bits(unsigned x, int p, int n)
{
    /* Maska 000000111...1100000 - n jedinica pocevsi od pozicije p */
    unsigned middle_n_1 = ~(~0 << n) << p+1-n;

    /* Invertujemo koristeći ekskluzivnu disjunkciju */
    return x ^ middle_n_1;
}

```

```

}

main()
{
    unsigned x = 0x0AA0AFA0;
    print_bits(x);

    print_bits(get_bits(x, 15, 8));
    print_bits(set_bits(x, 15, 8, 0xFF));
    print_bits(invert_bits(x, 15, 8));
}

```

Izlaz iz programa:

```

0000101010101000001010111110100000
00000000000000000000000010101111
0000101010101000001111111110100000
000010101010100000101000010100000

```

Primer 91 *right_rotate_bits, mirror_bits - rotiranje i simetrija bitova.*

```

#include <stdio.h>

/* Pomocna funkcija - stampa bitove neoznacnog broja */
void print_bits(unsigned x)
{
    int wl = sizeof(unsigned)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');

    putchar('\n');
}

/* Funkcija vrši rotaciju neoznacnog broja x za n pozicija u desno */
unsigned right_rotate(unsigned x, int n)
{
    int i;
    int wl = sizeof(unsigned)*8;

    /* Postupak se ponavlja n puta */
    for (i = 0; i < n; i++)
    {
        /* Poslednji bit broja x */
        unsigned last_bit = x & 1;

        /* x pomeramo za jedno mesto u desno */
        x >>= 1;

        /* Zapamceni poslednji bit stavljamo na pocetak broja x*/
    }
}

```

```

        x |= last_bit<<wl-1;
    }

    return x;
}

/* Funkcija obrće binarni zapis neoznačenog broja x tako sto bitove cita unatrag */
unsigned mirror(unsigned x)
{
    int i;
    int wl = sizeof(unsigned)*8;

    /* Rezultat inicijalizujemo na poslednji bit broja x */
    unsigned y = x & 1;

    /* Postupak se ponavlja wl-1 puta */
    for (i = 1; i<wl; i++)
    {
        /* x se pomera u desno za jedno mesto */
        x >>= 1;
        /* rezultat se pomera u levo za jedno mesto */
        y <<= 1;

        /* Poslednji bit broja x upisujemo na poslednje mesto rezultata */
        y |= x & 1;
    }
    return y;
}

main()
{
    unsigned x = 0xFAF0FAF0;
    print_bits(x);
    print_bits(mirror(x));
    print_bits(right_rotate(x, 2));
}

```

Izlaz iz programa:

```

11111010111100001111101011110000
00001111010111110000111101011111
00111110101111000011111010111100

```

Zadaci za vežbu:

Zadatak 39 Napisati operator dodeljivanja koji će broju x tipa `unsigned` sačuvati n krajnjih desnih bitova, a ostale postaviti na nulu.

$x = x \& \sim(0 \ll n)$; ili $x \&= \sim(0 \ll n)$;

Zadatak 40 Napisati operator dodeljivanja koji će u x očistiti n bitova (postaviti nule) počev od pozicije p .

$x \&= \sim(\sim(0 \ll n) \ll (p-1))$

Zadatak 41 Napisati operator dodeljivanja kojim se invertuje x (prevodi jedan u nulu i nula u jedan) počev od pozicije p na dužini n .

$x^{\wedge}=(\sim 0<<n)<<(p-1));$

12

Programski jezik C

1

12.1 Formiranje HTML dokumenta

Primer 92 *Prilikom pokretanja programa koristiti redirekciju:*

a.out >primer.html

kako bi se rezultat rada programa upisao u datoteku primer.html.

```
/*Ovaj program formira html dokument*/
#include <stdio.h>
main()
{
printf("<html><head><title>Ova stranica
      je napravljena u c-u</title></head>");
printf("<body><h3 align=center>
      Rezultat </h3></body></html>");
}
```

Primer 93 *Napisati program koji generiše html dokument sa engleskim alfabetom.*

```
#include <stdio.h>
main()
{
int i;
printf("<HTML><head><title>Engleski alfabet</title><head>\n");
printf("<body><ul>");
for(i=0;i<=25;i++)
    printf("<li> %c %c \n",'A'+i,'a'+i);
printf("</ul></body></HTML>\n"); }
```

Primer 94 *Napisati program koji generise html dokument koji prikazuje tablicu mnozenja za brojeve od 1 do 10.*

```
#include<stdio.h>
main()
{
```

¹Zasnovano na primerima sa sajtova <http://www.matf.bg.ac.yu/~filip>, <http://www.matf.bg.ac.yu/~milena>, <http://www.matf.bg.ac.yu/~jelenagr>.

```

int i,j;
printf("<html><head><title>Mnozenje</title></head>");
printf("<body><h3 align=center> Rezultat </h3>");
printf("<table border=1>\n");

/* Prva vrsta sadrzi brojeve od 1 do 10*/
printf("<tr>");
printf("<th></th>");
for(i=1; i<=10; i++)
    printf("<th> %d </th>\n", i);
printf("</tr>");

for(i=1; i<=10; i++)
{
    printf("<tr>");

    /* Na pocetku svake vrste stampamo broj
    odgovarajuće vrste*/
    printf("<th>%d</th>", i);

    for(j=1; j<=10; j++)
        printf("<td>%d\t</td>\n", i*j);

    printf("</tr>");
}
printf("</table>");
printf("</body></html>");
return 0;
}

```

12.2 Strukture

Primer 95 *Napisati program koji izračunava obim i površinu trougla i kvadrata.*

```

/* Program uvodi strukture - geometrijske figure */
#include <stdio.h>

/* Zbog funkcije sqrt. */
#include <math.h>
/* Upozorenje : pod linux-om je potrebno program prevoditi sa
    gcc -lm primer.c
    kada god se koristi <math.h>
*/

/* Tacke su predstavljene sa dve koordinate. Strukturom gradimo novi tip podataka. */
struct point
{
    int x;
    int y;
};

/* Izracunava duzinu duzi zadatu sa dve tacke */

```



```
float segment_length(struct point A, struct point B)
{
    int dx = A.x - B.x;
    int dy = A.y - B.y;
    return sqrt(dx*dx + dy*dy);
}

/* Izracunava povrsinu trougla Heronovim obrascem.
   Argumenti funkcije su tri tacke koje predstavljaju temena trougla */
float Heron(struct point A, struct point B, struct point C)
{
    /* Duzine stranica */
    float a = segment_length(B, C);
    float b = segment_length(A, C);
    float c = segment_length(A, B);

    /* Poluobim */
    float s = (a+b+c)/2;

    return sqrt(s*(s-a)*(s-b)*(s-c));
}

/* Izracunava obim poligona. Argumenti funkcije su niz tacaka
   koje predstavljaju temena poligona kao i njihov broj */
float circumference(struct point polygon[], int num)
{
    int i;
    float o = 0.0;

    /* Dodajemo duzine stranica koje spajaju susedna temena */
    for (i = 0; i<num-1; i++)
        o += segment_length(polygon[i], polygon[i+1]);

    /* Dodajemo duzinu stranice koja spaja prvo i poslednje teme */
    o += segment_length(polygon[num-1], polygon[0]);

    return o;
}

/* Izracunava povrsinu konveksnog poligona. Argumenti funkcije su niz tacaka
   koje predstavljaju temena poligona kao i njihov broj */
float area(struct point polygon[], int num)
{
    /* Povrsina */
    float a = 0.0;
    int i;

    /* Poligon delimo na trouglove i posebno izracunavamo povrsinu svakoga od njih */
    for (i = 1; i < num -1; i++)
        a += Heron(polygon[0], polygon[i], polygon[i+1]);

    return a;
}
```

```
}

main()
{
    /* Definisemo dve promenljive tipa tacke */
    struct point a;

    /* Inicijalizujemo tacku b na (1,2) */
    struct point b = {1, 2};

    /* triangle je niz od tri tacke - trougao (0,0), (0,1), (1,0) */
    struct point triangle[3];

    /* square je niz od cetiri tacke - jedinicni kvadrat.
       Obratiti paznju na nacin inicijalizacije niza struktura */
    struct point square[4] = {{0, 0}, {0, 1}, {1, 1}, {1, 0}};

    /* Postavljamo vrednosti koordinata tacke a */
    a.x = 0; a.y = 0;

    /* Gradimo trougao (0,0), (0,1), (1,0) */
    triangle[0].x = 0; triangle[0].y = 0;
    triangle[1].x = 0; triangle[1].y = 1;
    triangle[2].x = 1; triangle[2].y = 0;

    /* Ispisujemo velicinu strukture tacka */
    printf("sizeof(struct point) = %d\n", sizeof(struct point));

    /* Ispisujemo vrednosti koordinata tacaka */
    printf("x koordinata tacke a je %d\n", a.x);
    printf("y koordinata tacke a je %d\n", a.y);
    printf("x koordinata tacke b je %d\n", b.x);
    printf("y koordinata tacke b je %d\n", b.y);

    printf("Obim trougla je %f\n",
           circumference(triangle, 3));
    printf("Obim kvadrata je %f\n",
           circumference(square, 4));
    printf("Povrsina trougla je %f\n",
           Heron(triangle[0], triangle[1], triangle[2]));
    /* Broj tacaka je moguće odrediti i putem sizeof */
    printf("Povrsina kvadrata je %f\n",
           area(square, sizeof(square)/sizeof(struct point)));
}
```

Izlaz:

```
sizeof(struct point) = 8
x koordinata tacke a je 0
y koordinata tacke a je 0
x koordinata tacke b je 1
```

```
y koordinata tacke b je 2
Obim trougla je 3.414214
Obim kvadrata je 4.000000
Povrsina trougla je 0.500000
Povrsina kvadrata je 1.000000
```

Primer 96 *Ilustracija korišćenja typedef.*

```
/* Koriscenje typedef radi lakseg rada */
#include <stdio.h>

#include <math.h>
/* Ovim se omogućava da se nadalje u programu umesto int moze
koristiti ceo_broj */
typedef int ceo_broj ;

/* Ovim se omogućuje da se nadalje u programu umesto struct point
moze koristiti POINT */
typedef struct point POINT;

struct point
{
    int x;
    int y;
};

main()
{
    /* Umesto int mozemo koristiti ceo_broj */
    ceo_broj x = 3;

    /* Definisemo promenljivu tipa tacke.
    Umesto struct point mozemo koristiti POINT */
    POINT a;

    printf("x = %d\n", x);

    /* Postavljamo vrednosti koordinata tacke a*/
    a.x = 1; a.y = 2;
    /* Ispisujemo velicinu strukture tacka */
    printf("sizeof(struct point) = %d\n", sizeof(POINT));

    /* Ispisujemo vrednosti koordinata tacaka */
    printf("x koordinata tacke a je %d\n", a.x);
    printf("y koordinata tacke a je %d\n", a.y);
}
```

Izlaz:

```
x = 3
sizeof(struct point) = 8
x koordinata tacke a je 1
y koordinata tacke a je 2
```

12.3 Životni vek i oblast važenja promenljivih

Primer 97 *Demonstracija zivotnog veka i oblasti vazenja promenljivih (scope).*

```
#include <stdio.h>

/* Globalna promenjiva */
int a = 0;

/* Uvecava se globalna promenjiva a */
void increase()
{
    a++;
    printf("increase::a = %d\n", a);
}

/* Umanjuje se lokalna promenjiva a. Globalna promenjiva zadrzava svoju vrednost. */
void decrease()
{
    /* Ovo a je nezavisna promenjiva u odnosu na globalno a */
    int a = 0;
    a--;
    printf("decrease::a = %d\n", a);
}

void nonstatic_var()
{
    /* Nestaticke promenjive ne cuvaju vrednosti kroz pozive funkcije */
    int s=0;
    s++;
    printf("nonstatic::s=%d\n",s);
}

void static_var()
{
    /* Staticke promenjive cuvaju vrednosti kroz pozive funkcije.
       Inicijalizacija se odvija samo u okviru prvog poziva. */
    static int s=0;
    s++;
    printf("static::s=%d\n",s);
}

main()
{
    /* Promenjive lokalne za funkciju main */
    int i;
    int x = 3;

    printf("main::x = %d\n", x);
}
```

```

for (i = 0; i<3; i++)
{
    /* Promenjiva u okviru bloka je nezavisna od spoljne promenjive.
       Ovde se koristi promenjiva x lokalna za blok petlje koja ima
       vrednost 5, dok originalno x i dalje ima vrednost 3*/
    int x = 5;
    printf("for::x = %d\n", x);
}

/* U ovom bloku x ima vrednost 3 */
printf("main::x = %d\n", x);

increase();
decrease();

/* Globalna promenjiva a */
printf("main::a = %d\n", a);

/* Demonstracija nestatickih promenljivih */
for (i = 0; i<3; i++)
    nonstatic_var();

/* Demonstracija statickih promenljivih */
for (i = 0; i<3; i++)
    static_var();
}

```

Izlaz iz programa:

```

main::x = 3
for::x = 5
for::x = 5
for::x = 5
main::x = 3
increase::a = 1
decrease::a = -1
main::a = 1
nonstatic::s=1
nonstatic::s=1
nonstatic::s=1
static::s=1
static::s=2
static::s=3

```

Primer 98 *Ilustruje vidljivost imena*

```

#include <stdio.h>

int i=10;

void main() {
    {
        int i=3;
    }
}

```

```
    {
        int i=1;
        printf("%d\n", i);
    }
    printf("%d\n",i);
}
printf("%d\n",i);
}
```

13

Programski jezik C

1

13.1 Rad sa datotekama

1. /* Program demonstrira otvaranje datoteka ("r" - read i "w" - write mod) i osnovne tehnike rada sa datotekama */

```
/* U datoteku se upisuje prvih 10 prirodnih brojeva, a zatim se iz iste datoteke
citaju brojevi dok se ne stigne do kraja i ispisuju se na standardni izlaz */
```

```
#include <stdio.h>
```

```
/* Zbog funkcije exit */
```

```
#include <stdlib.h>
```

```
main()
```

```
{
```

```
    int i;
```

```
    int br;
```

```
    /* Otvaramo datoteku sa imenom podaci.txt za pisanje */
```

```
    FILE* f = fopen("podaci.txt", "w");
```

```
    /* Ukoliko otvaranje nije uspelolo, fopen vraca NULL. U tom slucaju,
```

```
       prijavljujemo gresku i završavamo program */
```

```
    if (f == NULL)
```

```
    {
```

```
        printf("Greska prilikom otvaranja datoteke podaci.txt za pisanje\n");
```

```
        exit(1);
```

```
    }
```

```
    /* Upisujemo u datoteku prvih 10 prirodnih brojeva (svaki u posebnom redu) */
```

```
    for (i = 0; i<10; i++)
```

```
        fprintf(f, "%d\n", i);
```

¹Zasnovano na primerima sa sajta <http://www.matf.bg.ac.yu/~filip>

```

    /* Zatvaramo datoteku */
    fclose(f);

    /* Otvaramo datoteku sa imenom podaci.txt za citanje */
    f = fopen("podaci.txt", "r");

    /* Ukoliko otvaranje nije uspjelo, fopen vraca NULL. U tom slucaju,
       prijavljujemo gresku i završavamo program */
    if (f == NULL)
    {
        printf("Greska prilikom otvaranja datoteke podaci.txt za citanje\n");
        exit(1);
    }

    /* Citamo brojeve iz datoteke dok ne stignemo do kraja i ispisujemo ih
       na standardni izlaz */

        /* Pokušavamo da procitamo broj */
        while(fscanf(f, "%d", &br) == 1)
            /* Ispisujemo procitani broj */
            printf("Procitano : %d\n", br);

    /* Zatvaramo datoteku */
    fclose(f);
}

2. /* Program demonstrira "a" - append mod datoteka - nadovezivanje */
#include <stdio.h>

main()
{
    FILE* datoteka;

    /* Otvaramo datoteku za nadovezivanje i proveravamo da li je doslo do greske */
    if ( (datoteka=fopen("dat.txt","a"))==NULL)
    {
        fprintf(stderr,"Greska : nisam uspeo da otvorim dat.txt\n");
        return 1;
    }

    /* Upisujemo sadržaj u datoteku */
    fprintf(datoteka,"Zdravo svima\n");

    /* Zatvaramo datoteku */
    fclose(datoteka);
}

3. /* Program ilustruje rad sa datotekama. Program kopira
    datoteku ulaz.txt u datoteku izlaz.txt. */
/* Uz svaku liniju se zapisuje i njen broj */
#include <stdio.h>

```



```
#define MAX_LINE 256

/* Funkcija getline iz K&R jednostavno realizovana preko funkcije fgets */

int getline(char s[], int lim)
{
    char* c = fgets(s, lim, stdin);
    return c==NULL ? 0 : strlen(s);
}

main()
{
    char line[MAX_LINE];
    FILE *in, *out;
    int line_num;

    if ((in = fopen("ulaz.txt", "r")) == NULL)
    {
        fprintf(stderr, "Neuspesno otvaranje datoteke %s\n", "ulaz.txt");
        return 1;
    }

    if ((out = fopen("izlaz.txt", "w")) == NULL)
    {
        fprintf(stderr, "Neuspesno otvaranje datoteke %s\n", "izlaz.txt");
        return 1;
    }

    /* Prepisivanje karakter po karakter je moguce ostvariti preko:
       int c;
       while ((c=fgetc(in)) != EOF)
           putc(c,out);
    */

    line_num = 1;
    /* Citamo liniju po liniju sa ulaza*/
    while (fgets(line, MAX_LINE, in) != NULL)
    {
        /* Ispisujemo broj linije i sadrzaj linije na izlaz */
        fprintf(out, "%-3d :\t", line_num++);
        fputs(line, out);
    }

    /* Zatvaramo datoteke */
    fclose(in);
    fclose(out);
}
```

4. /* Citanje niza struktura iz tekstualne datoteke - artikli prodavnice */

```
/* Datoteka cije se ime unosi sa standardnog ulaza sadrzi podatke o
   proizvodima koji se prodaju u okviru odredjene prodavnice.
```

Svaki proizvod se odlikuje sledecim podacima :

```
bar-kod   - petocifreni pozitivan broj
ime       - niska karaktera
cena      - realan broj zaokruzen na dve decimale
pdv       - stopa poreza - realan broj zaokruzen na dve decimale
```

Pretpostavljamo da su podaci u datoteci korektno zadati.

Pretpostavljamo da se u prodavnici ne prodaje vise od 1000 razlicitih artikala.

Na standardni izlaz ispisati podatke o svim proizvodima koji se prodaju.

*/

```
#include <stdio.h>
```

```
/* Maksimalna duzina imena proizvoda */
```

```
#define MAX_IME 30
```

```
/* Struktura za cuvanje podataka o jednom artiklu */
```

```
typedef struct _artikal
```

```
{
```

```
    int bar_kod;
    char ime[MAX_IME];
    float cena;
    float pdv;
```

```
} artikal;
```

```
/* Maksimalni broj artikala */
```

```
#define MAX_ARTIKALA 1000
```

```
/* Niz struktura u kome se cuvaju podaci o artiklima */
```

```
artikal artikli[MAX_ARTIKALA];
```

```
/* Broj trenutno ucitanih artikala */
```

```
int br_artikala = 0;
```

```
/* Ucitava podatke o jednom artiklu iz date datoteke.
```

```
Vraca da li su podaci uspesno procitani */
```

```
int ucitaj_artikal(FILE* f, artikal* a)
```

```
{
```

```
    /* Citamo podatke */
```

```
    if((fscanf(f, "%d", &(a->bar_kod))==1)
```

```
    && (fscanf(f, "%s", a->ime)==1)
```

```
    && (fscanf(f, "%f", &(a->cena))==1)
```

```
    && (fscanf(f, "%f", &(a->pdv))==1))
```

```
        /* Prijavljujemo uspeh */
```

```
        return 1;
```

```
    else
```

```
        /* Prijavljujemo neuspeh. */
```

```
        return 0;
```

```
}
```

```
/* Izracunava ukupnu cenu datog artikla */
```

```
float cena(artikal a)
```

```

{
    return a.cena*(1+a.pdv);
}

/* Ispisuje podatke o svim artiklima */
void ispisi_artikle()
{
    int i;
    for (i = 0; i<br_artikala; i++)
        printf("%-5d %-10s %.2f %.2f    = %.2f\n",
               artikli[i].bar_kod, artikli[i].ime,
               artikli[i].cena, artikli[i].pdv, cena(artikli[i]));
}

main()
{
    FILE* f;

    /* Ucitavamo ime datoteke */
    char ime_datoteke[256];
    printf("U kojoj datoteci se nalaze podaci o proizvodima: ");
    scanf("%s", ime_datoteke);

    /* Otvaramo datoteku i proveravamo da li smo uspeli */
    if ( (f = fopen(ime_datoteke, "r")) == NULL)
    {
        printf("Greska : datoteka %s ne moze biti otvorena\n",
               ime_datoteke);
    }

    /* Ucitavamo artikle */
    while (ucitaj_artikal(f, &artikli[br_artikala]))
        br_artikala++;

    /* Ispisujemo podatke o svim artiklima */
    ispisi_artikle();

    /* Zatvaramo datoteku */
    fclose(f);
}

```

Primer 99 Program ilustruje čitanje etiketa iz neke HTML datoteke.

```

#include <stdio.h>
#include <ctype.h>

/* Maksimalna duzina etikete */
#define MAX_TAG 100

#define OTVORENA 1
#define ZATVORENA 2

```

```

#define GRESKA 0

/* Funkcija ucitava sledecu etiketu
   i smesta njen naziv u niz s duzine max.
   Vraca OTVORENA za otvorenu etiketu,
   ZATVORENA za zatvorenu etiketu,
   odnosno GRESKA inace */
int gettag(FILE *f, char s[], int max)
{
    int c, i;
    int zatvorenost=OTVORENA;

    /* Preskacemo sve do znaka '<' */
    while ((c=fgetc(f))!=EOF && c!='<')
        ;
    /* Nismo naisli na etiketu */
    if (c==EOF)
        return GRESKA;

    /* Proveravamo da li je etiketa zatvorena */
    if ((c=fgetc(f))=='/')
        zatvorenost=ZATVORENA;
    else
        ungetc(c,f);

    /* Citamo etiketu dok nailaze slova
       i smestamo ih u nisku */
    for (i=0; isalpha(c=fgetc(f))
        && i<max-1; s[i++] = c)
        ;
    /* Vracamo poslednji karakter na ulaz
       jer je to bio neki karakter koji nije
       slovo*/
    ungetc(c,f);

    s[i]='\0';

    /* Preskacemo atribut do znaka > */
    while ((c=fgetc(f))!=EOF && c!='>')
        ;

    /* Greska ukoliko nismo naisli na '>' */
    return c=='>' ? zatvorenost : GRESKA;
}

main()
{
    char tag[MAX_TAG];
    int zatvorenost;

    FILE* f;

```

```
/* Ucitavamo ime datoteke */
char ime_datoteke[256];
printf("Unesite naziv html dokumenta iz kog se vrsi citanje etiketa: ");
scanf("%s", ime_datoteke);

/* Otvaramo datoteku i proveravamo da li smo uspeli */
if ( (f = fopen(ime_datoteke, "r")) == NULL)
{
    printf("Greska : datoteka %s ne moze biti otvorena\n",
           ime_datoteke);
}

while ((zatvorenost = gettag(f,tag,MAX_TAG))>0)
{
    if (zatvorenost==OTVORENA)
        printf("Otvoreno : %s\n",tag);
    else
        printf("Zatvoreno : %s\n",tag);
}

fclose(f);
}
```


14

Programski jezik C

1

14.1 Argumenti komandne linije

Primer 100 *Ilustracija rada sa argumentima komandne linije.*

```
/* Program pozivati sa npr.:
    ./a.out
    ./a.out prvi
    ./a.out prvi drugi treci
    ./a.out -a -bc ime.txt
*/

#include <stdio.h>

/* Imena ovih promenljivih mogu biti proizvoljna. Npr.

    main (int br_argumenata, char* argumenti[]);

    ipak, uobicajeno je da se koriste sledeca imena:
*/

main(int argc, char* argv[])
{
    int i;

    printf("argc = %d\n", argc);
    for (i = 0; i<argc; i++)
        printf("argv[%d] = %s\n", i, argv[i]);
}
```

Primer 101 *Program ispisuje opcije navedene u komandnoj liniji. K&R rešenje.*

¹Zasnovano na primerima sa sajtova <http://www.matf.bg.ac.yu/~filip>.

```

/* Opcije se navode koriscenjem znaka -, pri cemu je moguće da iza jednog -
   sledi i nekoliko opcija.
   Npr. za -abc -d -fg su prisutne opcije a b c d f g */

/* Resnje se intenzivno zasniva na pokazivackoj aritmetici i prioritetu operatora */

#include <stdio.h>

int main(int argc, char* argv[])
{
    char c;
    /* Dok jos ima argumenata i dok je karakter na poziciji 0 upravo crtica */
    while(--argc>0 && (argv[0]=='-'))
        /* Dok god ne dodjemo do kraja tekuceg stringa */
        while (c=argv[0][1])
            printf("Prisutna opcija : %c\n",c);
}

```

Izlaz:

```

Prisutna opcija : a
Prisutna opcija : b
Prisutna opcija : c
Prisutna opcija : d
Prisutna opcija : f
Prisutna opcija : g

```

Primer 102 Program ispisuje opcije navedene u komandnoj liniji - jednostavnija verzija.

```

#include <stdio.h>

main(int argc, char* argv[])
{
    /* Za svaki argument komande linije, pocevsi od argv[1]
       (preskacemo ime programa) */
    int i;
    for (i = 1; i < argc; i++)
    {
        /* Ukoliko i-ti argument pocinje crticom */
        if (argv[i][0] == '-')
        {
            /* Ispisujemo sva njegova slova pocevsi od pozicije 1 */
            int j;
            for (j = 1; argv[i][j] != '\0'; j++)
                printf("Prisutna je opcija : %c\n", argv[i][j]);
        }
        /* Ukoliko ne pocinje crticom, prekidamo */
        else
            break;
    }
}

```

Primer 103 Napisati program koji sa standardnog ulaza učitava pozitivan ceo broj, a na standardni izlaz ispisuje vrednost tog broja sa razmenjenim vrednostima bitova na poziciji i, j. Pozicije i, j se učitavaju kao parametri komandne linije. Smatrati da krajnji desni bit binarne

reprezentacije je 0-ti bit. Pri rešavanju nije dozvoljeno koristiti pomoćni niz niti aritmetičke operatore +, -, /, *, %.

```
#include <stdio.h>
unsigned Trampa(unsigned n, int i, int j);

main(int argc, char **argv)
{
    unsigned x; /*broj sa standardnog ulaza ciji se bitovi razmenjuju*/
    int i,j;    /*pozicije bitova za trampu*/

    /*ocitavanje parametara komandne linije i broja sa standarnog ulaza*/
    sscanf(argv[1], "%d", &i);
    sscanf(argv[2], "%d", &j);
    scanf("%u", &x);

    printf("\nNakon trampe vrednost unetog broja je %u\n", Trampa(x,i,j));
}

unsigned Trampa(unsigned n, int i, int j)
{
    //ako se bit na poziciji i razlikuje od bita na poziciji j, treba ih invertovati
    if ( ((n>>i)&1) != ((n>>j)&1) )    n^= (1<<i) | (1<<j);
    return n;
}
```

Primer 104 Iz datoteke čije se ime zadaje kao argrument komandne linije, učitati cele brojeve sve dok se ne učita nula, i njihov zbir ispisati u datoteku čije se ime takode zadaje kao argument komandne linije.

```
#include<stdio.h>
main(int argc, char* argv[])
{
    int n, S=0;
    FILE* ulaz, *izlaz;
    /* Ukoliko su imena datoteka navedena kao argumenti...*/
    if (argc>=3)
    {
        /* ...otvaramo datoteku i proveravamo da li smo uspeli */
        if ( (ulaz = fopen(argv[1], "r")) == NULL)
            printf("Greska : datoteka %s ne moze biti otvorena\n", argv[1]);
        if ( (izlaz = fopen(argv[2], "w")) == NULL)
            printf("Greska : datoteka %s ne moze biti otvorena\n", argv[2]);
    }
    else
    {
        char ime_datoteke_ulaz[256], ime_datoteke_izlaz[256];
        /* Ucitavamo ime datoteke */
        printf("U kojoj datoteci se nalaze brojevi: ");
        scanf("%s", ime_datoteke_ulaz);
        /* Otvaramo datoteku i proveravamo da li smo uspeli */
        if ( (ulaz = fopen(ime_datoteke_ulaz, "r")) == NULL)
            printf("Greska : datoteka %s ne moze biti otvorena\n", ime_datoteke_ulaz);
```

```

    printf("U kojoj datoteci treba ispisati rezultat: ");
    scanf("%s", ime_datoteke_izlaz);
    /* Otvaramo datoteku i proveravamo da li smo uspeli */
    if ( (izlaz = fopen(ime_datoteke_izlaz, "w")) == NULL)
        printf("Greska : datoteka %s ne moze biti otvorena\n", ime_datoteke_izlaz);
}

fscanf(ulaz, "%d", &n);
while(n!=0)
{
    S+=n;
    fscanf(ulaz, "%d", &n);
}
fprintf(izlaz, "Suma brojeva učitanih iz datoteke je %d.", S);
return 0;
}

```

14.2 Pokazivači - osnovni pojmovi

Primer 105 *Ilustracija rada sa pokazivačkim promenljivim.*

```

#include <stdio.h>
main()
{
    int x = 3;

    /* Adresu promenljive x zapamticemo u novoj promenljivoj.
       Nova promenljiva je tipa pokazivaca na int (int*) */
    int* px;

    printf("Adresa promenljive x je : %p\n", &x);
    printf("Vrednost promenljive x je : %d\n", x);

    px = &x;
    printf("Vrednost promenljive px je (tj. px) : %p\n", px);
    printf("Vrednost promenljive na koju ukazuje px (tj. *px) je : %d\n", *px);

    /* Menjamo vrednost promenljive na koju ukazuje px */
    *px = 6;
    printf("Vrednost promenljive na koju ukazuje px (tj. *px) je : %d\n", *px);

    /* Posto px sadrzi adresu promenljive x, ona ukazuje na x tako da je
       posredno promenjena i vrednost promenljive x */
    printf("Vrednost promenljive x je : %d\n", x);
}

```

Izlaz (u konkretnom slucaju):
 Adresa promenljive x je : 0012FF88
 Vrednost promenljive x je : 3

Vrednost promenljive px je (tj. px) : 0012FF88
Vrednost promenljive na koju ukazuje px (tj. *px) je : 3
Vrednost promenljive na koju ukazuje px (tj. *px) je : 6
Vrednost promenljive x je : 6

Primer 106 *swap : Demonstracija prenosa argumenata preko pokazivača.*

```
#include <stdio.h>

/* Pogresna verzija funkcije swap. Zbog prenosa po vrednosti, funkcija
   razmenjuje kopije promenljivih iz main-a, a ne samih promenljivih */
void swap_wrong(int x, int y)
{
    int tmp;

    printf("swap_wrong: ");
    printf("Funkcija menja vrednosti promenljivim na adresama : \n");
    printf("x : %p\n", &x);
    printf("y : %p\n", &y);

    tmp = x;
    x = y;
    y = tmp;
}

/* Resenje je prenos argumenata preko pokazivaca */
void swap(int* px, int* py)
{
    int tmp;

    printf("swap : Funkcija menja vrednosti promenljivim na adresama : \n");
    printf("px = %p\n", px);
    printf("py = %p\n", py);

    tmp = *px;
    *px = *py;
    *py = tmp;
}

main()
{
    int x = 3, y = 5;
    printf("Adresa promenljive x je %p\n", &x);
    printf("Vrednost promenljive x je %d\n", x);
    printf("Adresa promenljive y je %p\n", &y);
    printf("Vrednost promenljive y je %d\n", y);

    /* Pokusavamo zamenu koristeći pogresnu verziju funkcije */
    swap_wrong(x, y);

    printf("Posle swap_wrong:\n");
    printf("Vrednost promenljive x je %d\n", x);
```

```

printf("Vrednost promenljive y je %d\n", y);

/* Vrsimo ispravnu zamenu. Funkciji swap saljemo adrese promenljivih
   x i y, a ne njihove vrednosti */
swap(&x, &y);

printf("Posle swap:\n");
printf("Vrednost promenljive x je %d\n", x);
printf("Vrednost promenljive y je %d\n", y);
}

```

Izlaz u konkretnom slucaju:

```

Adresa promenljive x je 0012FF88
Vrednost promenljive x je 3
Adresa promenljive y je 0012FF84
Vrednost promenljive y je 5
swap_wrong: Funkcija menja vrednosti promenljivim na adresama :
x : 0012FF78
y : 0012FF7C
Posle swap_wrong:
Vrednost promenljive x je 3
Vrednost promenljive y je 5
swap : Funkcija menja vrednosti promenljivim na adresama :
px = 0012FF88
py = 0012FF84
Posle swap:
Vrednost promenljive x je 5
Vrednost promenljive y je 3

```

Primer 107 *Demonstracija više povratnih vrednosti funkcije koristeći prenos preko pokazivača.*

```

/* Funkcija istovremeno vraca dve vrednosti - kolicnik i ostatak dva broja.
   Ovo se postize tako sto se funkciji predaju vrednosti dva broja (x i y) koji se dele
   i adrese dve promenljive na koje ce se smestiti rezultati */
void div_and_mod(int x, int y, int* div, int* mod)
{
    printf("Kolicnik postavljam na adresu : %p\n", div);
    printf("Ostatak postavljam na adresu : %p\n", mod);
    *div = x / y;
    *mod = x % y;
}

main()
{
    int div, mod;
    printf("Adresa promenljive div je %p\n", &div);
    printf("Adresa promenljive mod je %p\n", &mod);

    /* Pozivamo funkciju tako sto joj saljemo vrednosti dva broja (5 i 2)
       i adrese promenljivih div i mod na koje ce se postaviti rezultati */
    div_and_mod(5, 2, &div, &mod);

    printf("Vrednost promenljive div je %d\n", div);
}

```

```
    printf("Vrednost promenljive mod je %d\n", mod);  
}
```

Izlaz u konkretnom slucaju:
Adresa promenljive div je 0012FF88
Adresa promenljive mod je 0012FF84
Kolicnik postavljam na adresu : 0012FF88
Ostatak postavljam na adresu : 0012FF84
Vrednost promenljive div je 2
Vrednost promenljive mod je 1

14.3 Strukture - uvežbavanje

Primer 108 *Strukture se u funkcije prenose po vrednosti. Moguće je koristiti pokazivače na strukture.*

```
#include <stdio.h>  
  
typedef struct point  
{  
    int x, y;  
} POINT;  
  
/* Zbog prenosa po vrednosti tacka ne moze biti ucitana */  
void get_point_wrong(POINT p)  
{  
    printf("x = ");  
    scanf("%d", &p.x);  
    printf("y = ");  
    scanf("%d", &p.y);  
}  
  
/* Koriscenjem prenosa preko pokazivaca, uspevamo */  
void get_point(POINT* p)  
{  
    /* p->x je skraceni zapis za (*p).x */  
  
    printf("x = ");  
    scanf("%d", &p->x);  
    printf("y = ");  
    scanf("%d", &p->y);  
}  
  
main()  
{  
    POINT a = {0, 0};  
  
    printf("get_point_wrong\n");  
    get_point_wrong(a);  
    printf("a: x = %d, y = %d\n", a.x, a.y);  
}
```

```

    printf("get_point\n");
    get_point(&a);
    printf("a: x = %d, y = %d\n", a.x, a.y);
}

```

Primer 109 *Datoteka cije se ime unosi sa komandne linije sadrži podatke o studentima (ime, prezime, broj indeksa). Podaci su korektno zadati. Nema više od 1000 studenata. Prvo formirati niz struktura u memoriji, a onda ih ispisiati.*

```

#include <stdio.h>

#define MAXL 100
#define MAXN 1000

typedef struct student {
    char ime[MAXL];
    char prezime[MAXL];
    short int indeks;
} student;

int main(int argc, char *argv[])
{
    FILE *in; int j,i=0;
    student niz[MAXN];

    if(argc!=2)
    {
        fprintf(stderr,"Neispravno pozivanje! Koriscenje: %s <ime datoteke>\n",argv[0]);
        return -1;
    }
    if(!(in=fopen(argv[1],"r")))
    {
        fprintf(stderr,"Ne mogu da otvorim datoteku %s za citanje.",argv[1]);
        return -1;
    }

    while(!feof(in))
    {
        fscanf(in,"%s %s %d",&niz[i].ime, &niz[i].prezime, &niz[i].indeks);
        i++;
    }

    /* Sredjujemo zadnji scanf koji učitava EOF */
    i--;

    for(j=0;j<i;j++)
    {
        printf(stdout,"Ime: %s\nPrezime: %s\nIndeks: %d\n\n",
            niz[j].ime, niz[j].prezime, niz[j].indeks);
    }
}

```

```

    fclose(in);
    return 0;
}

```

Primer 110 Sa standardnog ulaza se učitava niz od n ($n < 100$) tačaka u ravni takvih da nikoje tri tačke nisu kolinearne. Tačke se zadaju parom svojih koordinata (celi brojevi). Ispitati da li taj niz tačaka određuje konveksni mnogougao i rezultat ispisati na standardni izlaz.

```

#include<stdio.h>

typedef struct tacka
{
    int x;
    int y;
} TACKA;

/* F-ja ispituje da li se tacke T3 i T4 nalaze sa iste strane prave
   odredjene tackama T1 i T2.*/
int SaIsteStranePrave(TACKA T1, TACKA T2, TACKA T3, TACKA T4)
{
    int t3 = (T3.y - T1.y)*(T2.x - T1.x) - (T2.y - T1.y) * (T3.x - T1.x);
    int t4 = (T4.y - T1.y)*(T2.x - T1.x) - (T2.y - T1.y) * (T4.x - T1.x);
    return (t3 * t4 > 0);
}

main()
{
    TACKA mnogougao[100];
    int j,i;
    int n;
    int konveksan = 1;

    do
    {
        printf("Unesite broj temena mnogougla:\n");
        scanf("%d",&n);
        if(n<3)
            printf("Greska! Suviše malo tacaka! Pokusajte ponovo!\n");
    }
    while(n<3);

    printf("Unesite koordinate temena mnogougla takve da nikoja tri
           temena nisu kolinearna!\n");
    for(i=0;i<n;i++)
        scanf("%d %d", &mnogougao[i].x, &mnogougao[i].y);

    /* Da bi mnogougao bio konveksan potrebno (i dovoljno) je da kada se
       povuce prava kroz bilo koja dva susedna temena mnogougla sva ostala
       temena budu sa iste strane te prave.*/
    for(i=0;konveksan&&i<n-1;i++)
    {
        for(j=0;konveksan&&j<i-1;j++)

```

```

        konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
            mnogougao[i+1],mnogougao[j],mnogougao[j+1]);
    for(j=i+2;konveksan&&j<n-1;j++)
        konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
            mnogougao[i+1],mnogougao[j],mnogougao[j+1]);
    if(i!=0&&i!=n-1&&i+1!=0&&i+1!=n-1)
        konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
            mnogougao[i+1],mnogougao[0],mnogougao[n-1]);
    }
    for(j=1;konveksan&&j<n-2;j++)
        konveksan=konveksan && SaIsteStranePrave(mnogougao[0],
            mnogougao[n-1],mnogougao[j],mnogougao[j+1]);

    if(konveksan)
        printf("Uneti mnogougao jeste konveksan!\n");
    else
        printf("Uneti mnogougao nije konveksan!\n");
}

```


15

15.1 Zadaci sa prethodnih ispita i kolokvijuma

Zadatak 42 I kolokvijum, februar 2005.

1. Napisati funkciju koja ispituje da li dve niske (koje se prenose kao parametri funkcije) su anagrami. Anagrami su niske koje se sastoje od istih karaktera. Npr. vetar, trave, verat su anagrami.
2. Napisati program koji testira funkciju iz prvog dela.

Zadatak 43 I kolokvijum, februar 2005. Napisati program koji učitava sa standardnog ulaza dve niske sa ne više od 80 karaktera u svakoj i prirodan broj k i ispisuje na standardni izlaz poruku da li se prva niska dobila cikličnim pomeranjem druge niske za k mesta. Na primer za $k=3$, niska CDEAB"" se dobila cikličnim pomeranjem niske "ABCDE"

Zadatak 44 I kolokvijum, februar 2005. Napisati program koje će učitati sa tastature broj s (unsigned int) i brojeve m i n (int), pri čemu je $0 \leq m \leq n < \text{sizeof}(\text{unsigned}) * 8$ i formirati vrednost d (unsigned int) u kojoj je bit na poziciji i jednak 1 akko je $m \leq i \leq n$ (pozicije se broje od nule sdesna na levo). Program treba da na standardnom izlazu ispiše broj koji se dobija od s postavljanjem na 0 svih bitova koji su u d jednaki 1.

Zadatak 45 Septembar, 2005. Napisati program koji sa standardnog ulaza učitava pozitivan ceo broj, a na standardni izlaz ispisuje vrednost tog broja sa razmenjenim vrednostima bitova na poziciji i , j . Pozicije i , j se učitavaju kao parametri komandne linije. Smatrati da krajnji desni bit binarne reprezentacije je 0-ti bit. Pri rešavanju nije dozvoljeno koristiti pomoćni niz niti aritmetičke operatore +, -, /, *, %.

Zadatak 46 Septembar, 2005. Sa standardnog ulaza se učitava niz od n ($n < 100$) tačaka u ravni takvih da nikoje tri tačke nisu kolinearne. Tačke se zadaju parom svojih koordinata (celi brojevi). Ispitati da li taj niz tačaka određuje konveksni mnogougao i rezultat ispisati na standardni izlaz.

Zadatak 47 Jun, 2004. Datoteka Matrice.txt sadrži dve celobrojne kvadratne matrice. U datoteci su prvo zapisane dimenzije matrica n i m ($n > m$) a zatim i elementi prvo jedne a zatim i druge matrice. Napisati program koji proverava da li se manja matrica sadrži u većoj. Matrica se sadrži u matrici veće dimenzije ukoliko postoji podmatrica veće matrice identična manjoj matrici tj. ako postoji blok veće matrice dimenzije $m \times m$ čiji su elementi jednaki elementima manje matrice na odgovarajućim pozicijama. npr. U matrici

```
1 1 1
2 2 2
3 3 3
```

se sadrži matrica

```
1 1
2 2
```

a ne sadrži matrica

```
1 1
3 3
```

Zadatak 48 Jun, 2004. Napisati funkciju koja računa multiplikativnu otpornost datog pozitivnog broja. Multiplikativna otpornost se računa na sledeći način $n0 = n$, nk je jednak proizvodu cifara broja n $k-1$, $k = 1, 2, \dots$, multiplikativna otpornost je najmanje k za koje je nk jednocifren broj. Napisati program koji iz datoteke čije se ime zadaje kao prvi argument komandne linije čita brojeve, gde su brojevi zapisani po jedan u svakom redu i u drugu datoteku čije se ime zadaje kao drugi argument komandne linije upisuje red po red date brojeve i njihovu multiplikativnu otpornost.

Zadatak 49 Prvi kolokvijum za II tok 2004.godine - rad na racunaru Napisati program koji generiše HTML fajl Boje.html koji sadrži tabelu boja. Tabela treba da ima 8 kolona pri čemu ćelije neparnih kolona treba da sadrže heksadekadnu vrednost boje i to u formatu ROGOBO a ćelije odgovarajuće parne kolone treba da budu obojene tom bojom.

Zadatak 50 Prvi kolokvijum za II tok 2004.godine - rad na racunaru Sa standardnog ulaza se unose veliki, celi, neoznačeni brojevi sa najviše 100 cifara. Ovih brojeva ima manje od 100 ali njihov broj nije unapred poznat. Napisati program koji sabira ovako unete brojeve i na standardni izlaz ispisuje njihov zbir.

Napomena : Svaki broj se unosi u posebnom redu a potrebno je voditi računa o korektnosti ulaznih podataka.

Zadatak 51 Prvi kolokvijum za II tok 2004.godine - rad na papiru Sa standardnog ulaza se unose dve niske koje predstavljaju elemente dva skupa. Skupovi nemaju više od 20 elemenata. Napisati program koji na standardni izlaz ispisuje niske koje predstavljaju:

1. presek,
2. uniju i
3. razliku

elemenata dva skupa.

Zadatak 52 Januar, 2002. Datoteka "izrazi.dat" sadrži izraze koji se sastoje od celobrojnih i realnih konstanti i operacija $+, -, *, /$ i zapisani su u inverznoj poljskoj notaciji (operandi pa operacija). Na primer, izraz $(1+2)/(3-4)$ zapisan je kao $1\ 2\ +\ 3\ 4\ -\ /\$, a izraz $21+7*6$ kao $21\ 7\ 6\ *\ +$. Svaki izraz je u datoteci zapisan u novom redu i podrazumeva se da su izrazi sintaksno ispravni. Napisati program koji izračunava i štampa na ekran vrednosti svih izraza u datoteci. Rešenje napisati modularno i obavezno ga komentarisati.

Zadatak 53 Januar, 2002. Program sa standardnog ulaza učitava raspored 8 topova na šahovskoj tabli. Raspored se sastoji od 8 linija sa po 8 brojeva u svakoj liniji. Svaka linija odgovara jednom redu table, a svaki broj jednom polju. Broj ima vrednost 0 ako na datom polju nema topa i vrednost 1 ako na datom polju postoji top. Program treba da ispita da li je uneseni raspored validan (tj. da li je svaki učitani broj 1 ili 0 i da li ima ukupno 8 topova na tabli), kao i da odredi da li se u datom rasporedu neka dva topa tuku (topovi se tuku ukoliko se nalaze u istom redu ili istoj koloni table). Program treba da ispiše na standardnom izlazu "raspored nije validan" ukoliko ulazni podaci nisu dobri, a u suprotnom "ne tuku se" ukoliko je raspored takav da se nijedan par topova međusobno ne tuče, odn. "tuku se" ukoliko ima topova koji se tuku

Zadatak 54 Januar, 2002. Sa standardnog ulaza se učitava u jednoj liniji prirodan broj n , a potom i linije teksta do markera kraja fajla. Napisati program koji štampa n reči koje se najčešće pojavljuju i to počev od najfrekventije reči. Uz reč odštampati i broj pojava. Reč je po definiciji ma koji niz karaktera koji ne sadrži blanko, tabulator, znak za novi red. Sve poruke o greškama ispisati na standardnom izlazu za poruke o grešci.

Zadatak 55 Januar, 2002. Napisati program koji čita ulaznu datoteku `ulaz.htm` i štampa na standardni izlaz samo linije koje imaju 70 karaktera van etiketa, pri čemu se tekst markiran u obliku `&entity;` (npr. `<` i `>`) ili `&#number;` (npr. `č`) broji kao 1 karakter. Programi da budu pisani čitko i izdašno komentarisani.

Zadatak 56 Februar, 2002. Neka se relacija nad nekim skupom elemenata opisuje kvadratnom matricom na sledeći način: ako je u preseku i -te vrste i j -te kolone 1, to znači da je i -ti element u relaciji sa j -tim, ako je 0 to znači da nije u relaciji. Sa standardnog ulaza zadaje se najpre dimenzija ovakve matrice, pa zatim elementi matrice, jedan za drugim, po vrstama. Dimenzija matrice nije ograničena. napisati program koji, pošto proveriti korektnost ulaza, za ovako zadatu relaciju ispituje njenu refleksivnost, simetričnost i tranzitivnost i odgovarajuće poruke štampa na ekran.

Zadatak 57 Februar, 2002. Datoteka `prica.txt` sadrži niz reči (reč je niz karaktera koji ne sadrži blanko, tabulator ili znak za novi red). Sa standardnog ulaza učitava se jedna reč. Nijedna reč, nema više od 20 karaktera. Napisati program koji broji i štampa na ekran koliko se puta data reč pojavila u datoteci, ako se zna da su neke reči pogrešno unete. Smatramo da je neka reč jednaka učitanoj i onda kada:

- je zamenjeno jedno slovo nekim drugim slovom
- ili je izostavljeno jedno slovo u jednoj od te dve reči

Zadatak 58 Februar, 2002. Napisati program koji za dva data pravougaonika R_0 i R_1 sa stranicama paralelnim koordinatnim osama izračunava i na standardni izlaz ispisuje površine njihovih unija ($R_0 \cup R_1$), preseka ($R_0 \cap R_1$) i razlike ($R_0 \setminus R_1$). Pravougaonici se učitavaju sa standardnog ulaza i zadati su koordinatama donjeg leveg, odn. gornjeg desnog tjemenca. Ove koordinate su realni brojevi. Za čuvanje podataka koji određuju neki pravougaonik deklarirati odgovarajuću strukturu. Sve operacije nad pravougaonikom (ili pravougaonicima) izdvojiti u posebne funkcije. Primjer: za pravougaonike zadate na sledeći način:

```
10 20 30 40
20 30 40 50
```

program treba da ispiše:

```
Povrsina unije iznosi 700
Povrsina preseka iznosi 100
Povrsina razlike iznosi 300"
```

Zadatak 59 Februar, 2002. U datoteci `tajna.txt` nalazi se riječ dužine ne veće od 20 karaktera. Riječ se sastoji isključivo od malih slova. Napisati program za pogađanje riječi. Program treba da učita riječ iz datoteke, a zatim da sa standardnog ulaza čita jedno po jedno slovo koja daje korisnik pogađajući da li ih riječ sadrži. Po učitavanju svakog slova program treba da ispiše ona slova u riječi koja su dotad pogodena. Na mjestima ostalih slova treba da budu karakteri *. Voditi računa o mogućnosti da korisnik greškom unese nešto što nije slovo, takode i neko slovo koje je ranije već unosio. Program ne treba da pravi razliku između malih i velikih slova, tj. ako korisnik unese neko veliko slovo, program treba da ga tretira kao malo slovo. Kada sva slova budu pogodena, program treba da ispiše ukupan broj pokušaja. Primjer sesije za slučaj kada je riječ koja se pogađa **zdravo** bi mogao biti:

```

a
***a**
e
***a**
i
***a**
o
***a*o
r
**ra*o
m
**ra*o
b
**ra*o
d
*dra*o
v
*dravo
z
zdravo
Ukupan broj pokusaja: 10

```

Zadatak 60 Februar, 2002. Napisati program koji učitava kvadratnu matricu sa standardnog ulaza čiji su članovi celi brojevi i proverava da li je matrica ortogonalna. Ne koristiti pomoćne matrice! U prvoj liniji nalaze se dimenzija matrice, a zatim se u svakoj liniji nalaze vrste matrice. Elementi unutar vrste su razdvojeni blanko znakovima. Dimenzija matrice nije unapred poznata. Pretpostaviti da su sve linije sem prve u ispravnom formatu i u slučaju greške izdati poruku na standardnom izlazu za poruke o grešci.

Zadatak 61 Februar, 2002. Parametri komandne linije su imena dve datoteke i ceo broj n . Napisati program koji poslednjih n linija prve datoteke upisuje u drugu datoteku. Može se pretpostaviti da prva datoteka ne sadrži linije duže od 80 karaktera, ali broj linija u datoteci nije unapred ograničen. U slučaju greške izdati poruku na standardnom izlazu za poruke o grešci.

Program komentarisati i programski kod pisati čitko.

Zadatak 62 April, 2002. . Prvi red standardne ulazne datoteke sadrži 2 cela broja manja od 50 koji predstavljaju redom broj vrsta i broj kolona realne matrice A . Svaki sledeći red sadrži po jednu vrstu matrice. Napisati program koji :

1. nalazi sve elemente matrice A koji su jednaki zbiru svih svojih susednih elemenata i štampa ih u obliku (broj vrste, broj kolone, vrednost elementa)
2. nalazi i štampa sve četvorke oblika
 $(A(i,j), A(i+1,j), A(i,j+1), A(i+1,j+1))$ u kojima su svi elementi međusobno različiti.

Zadatak 63 April, 2002. Parametri komandne linije su nazivi 2 datoteke. Prva datoteka sadrži niz reči čiji broj i čija dužina nije ograničena (mogu biti proizvoljno veliki brojevi) . Reč je bilo kakav niz karaktera koji nije blanko, tabulator ili oznaka za kraj reda. Napisati program koji u drugu datoteku prepisuje samo one reči iz prve datoteke koje su parne dužine i koje i počinju i završavaju se slovom. (napomena: obavezno voditi računa o tome da se dužina reči ne može ograničiti!)

Zadatak 64 April, 2002. Napisati program koji ispisuje kalendar za zadati mesec i godinu XX vijeka. Poznato je da je 1. januar 1901. bio utorak. Program prima dva argumenta u komandnoj

liniji: broj u intervalu $[1, 12]$ koji predstavlja mjesec i broj u intervalu $[1901, 2000]$ koji predstavlja godinu (obavezno proveriti validnost ovih argumenata). Program treba da ispiše kalendar na standardni izlaz i to tako što će u prvom redu biti ispisani mjesec (punim imenom) i godina, u narednom redu dvoslovne skraćenice od imena dana, počev od ponedjeljka i sa po jednim blanko znakom između skraćenica, a zatim u narednim redovima datumi, pri čemu se za svaki dan odvajaju po 2 mjesta u kojima broj treba da je poravnat udesno, a između dana se ostavlja po jedan blanko znak. Tako npr, ako su argumenti koji su zadati u komandnoj liniji 1 1970, ispis treba da ima sledeći oblik:

Januar 1970.

Po	Ut	Sr	Ce	Pe	Su	Ne
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Zadatak 65 April, 2002. Napisati program koji učitava sa standardnog ulaza prvo jednu liniju teksta a zatim još jednu liniju sa karakterima koje treba izbaciti iz prve linije. Program treba da izbaci specificirane karaktere iz prve linije i ispiše ono što preostane od iste. Dužina prve linije nije unapred ograničena, tj. za čuvanje te linije treba koristiti listu pri čemu će po jedan karakter biti smješten u svaki element liste. Primjer: ako je unos imao sledeći oblik:

```
Hello, world!
aeiou,
```

program treba da ispiše:

```
Hll wrld!
```

Zadatak 66 April, 2002. . Sastaviti program koji ispisuje $n < 19$ redova Pascalovog trougla koristeći samo 1-dimenzionalni niz i ne koristiti rekurziju. Broj n se zadaje kao jedini sadržaj linije standardnog ulaza. Izveštaj o eventualnim greškama na ulazu ispisati i na standardnom izlazu za poruke o grešci.

Zadatak 67 April, 2002. Argumenti komandne linije su imena tri datoteke. Preve dve datoteke u svakom redu sadrže do 80 cifara i u obe datoteke sadržaj je sortiran strogo rastuće po numeričkoj vrednosti broja predstavljenog tim ciframa. Napisati program koji će spojiti te dve datoteke u treću čiji će sadržaj takode biti sortiran strogo rastuće po numeričkoj vrednosti brojeva koje sadrži.

Zadatak 68 Jun, 2002. Neka je $P = (p_1, p_2, \dots, p_n)$ permutacija brojeva $1, 2, \dots, n$. Napisati PASCAL program koji za učitani prirodan broj $n < 50$ i za učitani tablicu inverzije ispisuje odgovarajuću permutaciju. Pod tablicom inverzije permutacije P se podrazumeva niz $S = (s_1, s_2, \dots, s_n)$ u kom je s_i jednako broju elemenata permutacije P koji (u P) stoje levo od broja i , a veći su od broja i .

Zadatak 69 Jun, 2002. Slika je opisana u kvadratnoj matrici tako da elementi koji određuju sliku popunjeni su cifrom 1, a ostali elementi su popunjeni cifrom 0. kao parametar komandne linije zadaje se ime datoteke u čijoj prvoj liniji se nalazi dimenzija matrice koaj opisuje sliku, a zatim u svakoj liniji nalaze se vrste matrice. Elementi unutar vrste su razdvojeni blankom. napisati program koji u svakoj liniji datoteke REPORT.DAT ispisuje poruke o simetričnosti matrice u odnosu na horizontalnu osu, vertikalnu osu, glavnu dijagonalu, sporednu dijagonalu, centar.

Zadatak 70 Jun, 2002. Sprovedena je anketa o popularnosti televizijskih emisija. Broj emisija za koje se glasalo nije veći od 50. Ispitanici su podeljeni u 4 kategorije: mškarci do 30 godina, žene do 30 godina, muškarci stariji od 30 godina, žene starije od 30 godina. Svi su glasali za 3 emisije. Svaka linija u datoteci čije se ime zadaje kao prvo u komandnoj liniji, sadrži podatke o glasanju jednog ispitanika i to sledećim redom: pol ispitanika (m ili z), broj godina, pa zatim šifre emisija za koje je ta osoba glasala. Šifra emisije je niz od najviše 5 karaktera. Napisati program koji u datoteku čije ime se zadaje kao drugo u komandnoj liniji, ispisuje šifre emisija i odgovarajući broj glasova poredane nerstuće po broju osvojenih glasova i to za svaku od kategorija posebno. Emisije za koje se nije glasalo treba preskočiti u ispisivanju.

Zadatak 71 Jun, 2002. Sa standardnog ulaza unosi se najpre jedna linija teksta čija dužina nije ograničena, pa zatim još jedna linija koja sadrži samo karakter koji iz prve linije treba izbaciti. Napisati program koji treba da ispiše rezultat odnosno šta je preostalo od linije, pa zatim unošenjem sledećeg karaktera koji se želi izbaciti da ponovi postpak za novodobijenu liniju, i tako dalje po istom principu sve dok se za karakter koji se želi izbaciti ne unese * ili dok od linije ne ostane ništa. Pošto dužina linije nije ograničena, za njeno čuvanje treba koristiti povezanu listu kod koje svaki element čuva po jedno slovo iz linije. Koristiti modularni pristup. Primer jedne sesije bi mogao biti:

```
programiranje
p
rogramiranje
e
rogramiranj
s
rogramiranj
a
rogrmirnj
*
```

Zadatak 72 Jun, 2002. Data je datoteka u kojoj se nalazi tekst čiji su naslovi obeleženi etiketom h. Maksimalna dubina naslova je n, gde se ceo broj n zadaje kao argument komandne linije. Svaka etiketa naslova je zatvorena. (Npr. <h3>Zadaci za pismeni</h3>). Jedini komentari u tekstu sadrže oznake broja strane i oblika su <!--xxxx--> gde je xxxx najviše četvorocifreni neoznačen broj. Tekst je kodiran bez ikakvih grešaka! Sastaviti program koji iz komandne linije uzima ime gore opisane datoteke i kreira na izlazu datoteku u kojoj se nalazi sadržaj ulaznog teksta. Sadržaj se formira kao niz redova koji sadrže niske obeležene h-etiketama i odgovarajući broj strane. Npr.

```
ulaz:          izlaz:
<h3>Zadaci za pismeni</h3> 2.2.3. Zadaci za pismeni.....228
<h4>Pitanja za usmeni</h4> 2.2.3.1. Pitanja za usmeni.....235
```

gde su navedeni naslovi uzastopni. Sadržaj ulazne datoteke se mora formirati pre nego što se u nju upiše.

Zadatak 73 Jun, 2002. U tekstualnoj datoteci nalaze se podaci o prijemnom ispitu učenika jedne osnovne škole tako što je u svakom redu navedeno: ime i prezime učenika (niz znakova ne duži od 50 znakova), broj poena na osnovu uspeha (decimalan broj), broj poena na prijemnom ispitu iz matematike (decimalan broj) i broj poena na prijemnom ispitu iz maternjeg jezika (decimalan broj). Za učenika koji osvoji manje od 10 poena ukupno na oba prijemna smatra se da nije položio prijemni. Napisati program na C-u koji na osnovu podataka iz ove datoteke formira i prikazuje rang listu učenika. Rang lista sadrži: redni broj, ime i prezime učenika, broj poena na osnovu uspeha, broj poena na prijemnom ispitu iz maternjeg jezika, broj poena na prijemnom ispitu iz matematike i ukupan broj poena i sortirana je opadajuće po ukupnom broju poena. U rang listi se navode prvo oni učenici koji su položili prijemni a potom učenici koji nisu položili prijemni. Između ove dve

grupe staviti horizontalnu liniju (—————). Ime datoteke navodi se kao argument komandne linije.

Zadatak 74 Jun, 2002. Napisati program u C-u koji sa standardnog ulaza učitava cifre n i k , a na standardnom izlazu prikazuje najmanji prirodan broj koji počinje cifrom n i ima svojstvo da se smanjuje k puta kada se cifra n premesti sa početka na kraj. Primer: za $n=3$ i $k=2$ traženi broj je 315789473684210526

Zadatak 75 Jun, 2002. Svaka linija datoteke čije se ime prosleđuje komandnom linijom sadrži po 6 celih brojeva: $x_1, y_1, x_2, y_2, x_3, y_3$ koji predstavljaju redom koordinate temena jednog trougla. Linija u datoteci nema više od 100. Napisati program koji uzimajući u obzir samo trouglove koji su jednakostranični, ispituje da li se oni svi mogu "upisati" jedan u drugi (ako je jedan trougao upisan u drugi njegova temena mogu i ne moraju pripadati stranicama ovog drugog). Odgovarajuću poruku štampati na ekran.

Zadatak 76 Jun, 2002. Data je datoteka u kojoj se nalazi tekst u kom se nazivi institucija koji se satoje od slova engleske abecede i blanka obeležavaju etiketom name i atributom type.

Npr. `<name type='institution'>Palata pravde</name>` maksimalna dužina naziva institucije je n , gde se ceo broj n zadaje kao argument komandne linije. Jedini komentari u tekstu sadrže oznake broja strane i oblika su `<!-- -xxxx- -->` gde je `xxxx` najviše četvorocifreni neoznačen broj. Tekst je kodiran bez ikakvih greški. Sastaviti program koji iz komandne linije uzima ime gore opisane datoteke i kreira na izlazu datoteku `index.dat` u kojoj se nalazi indeks ulaza koji se formira kao niz redova koji sadrže naziv institucije i broj prve stranice na kojoj se taj naziv pojavio. nazive institucija koji se javljaju često (više od m puta, gde se m zadaje kao argument komandne linije) ne unositi u indeks. Program ne trab da pravi razliku između malih i velikih slova.

Zadatak 77 Septembar, 2002. Svaki red datoteke čije se ime zadaje komandnom linijom, sadrži po 3 cela broja: A, B, C (A i B nisu istovremeno jednaki nuli), koji predstavljaju koeficijente prave u ravni $Ax+By+C=0$. Broj redova u datoteci nije veći od 100. Napisati program koji pronalazi i na standardnom izlazu ispisuje sve parove paralelnih pravih, kao i sve trojke pravih koje se seku u jednoj tački. Način prikaza traženih podataka je proizvoljan, ali treba voditi računa o njihovoj preglednosti.

Zadatak 78 Septembar, 2002. Grupa od n plesača (na čijim kostimima su redom brojevi od 1 do n) uvežbava svoju plesnu tačku tako što formiraju krug iz kog će redom izlaziti plesači na sledeći način:

1. počev od plesača označenog brojem 1, a brojeći udesno (ka plesačima sa većim rednim brojevima), izlazi m -ti plesač
2. nakon isključenja, brojanje otpočinje od sledećeg plesača i to u suprotnom smeru, tj. ako se brojalo udesno, počinje se od desnog suseda isključenog plesača i broji se ulevo
3. izlasci iz kruga se nastavljaju sve dok svi plesači ne budu isključeni

Celi brojevi m, n se zadaju kao argumenti komandne linije. Napisati C program koji ispisuje redne brojeve plesača u redosledu napuštanja kruga.

Zadatak 79 Septembar, 2002. N osoba obeleženih brojevima 1, 2, . . . N stoji u krugu. Počev od osobe sa rednim brojem 1 broji se K osoba i K -ta osoba izlazi iz kruga, a potom se nastavlja brojanje preostalih osoba na isti način, počev od prve osobe koja je izašla. Ovo se nastavlja sve dok u krugu ne ostane samo jedna osoba. Napisati program koji sa standardnog ulaza učitava vrednosti za N i K , a na standardnom izlazu prikazuje redosled izlaska ljudi iz kruga i redni broj osobe koja poslednja ostaje. Primer: za $N=4$ i $K=3$ redosled izlazaka je 3, 2, 4 i na kraju ostaje 1.

Zadatak 80 Septembar, 2002. Parametar komandne linije je ime datoteke čiji svaki red (izuzev prvog) je oblika `ime_deteta:ime_roditelja`. Prvi red sadrži samo ime jednog roditelja čija su sva deca navedena u narednim redovima u već opisanom obliku. Nije obavezno da se sva deca istog roditelja pojavljuju u uzastopnim redovima i nije unapred poznat ukupan broj roditelja. Jednostavnosti radi, može se smatrati: da sve osobe imaju imena sastavljena od slova engleske abecede, da su sva imena međusobno različita (ignorirajući razliku malih i velikih slova), da svaki roditelj nema više od četvero dece i da redovi datoteke nemaju više od 40 karaktera. Napisati program koji za svaku osobu *X* formira datoteku (čiji je naziv ime osobe) i koja u svakom redu sadrži imena najbližih stričeva, tetki, ujaka osobe *X* (misli se na rođenu braću i sestre roditelja osobe *X*).

Zadatak 81 Septembar, 2002. Slika je opisana u kvadratnoj matrici tako da elementi koji određuju sliku popunjeni su cifrom 1, odnosno cifrom 0. Kao parametar komandne linije zadaje se ime datoteke u čijoj prvoj liniji se nalazi dimenzija matrice koja opisuje sliku, a zatim se u svakoj liniji nalaze vrste matrice. Elementi unutar vrste su razdvojeni blankom. Napisati *C* program koji, ne koristeći pomoćne matrice, premešta podsluku (čije koordinate gornjeg levog ugla, dužina i širina se zadaju kao argumenti komandne linije) na novu poziciju čiji položaj gornjeg levog ugla se zadaje sa standardnog ulaza. Original i kopija moraju ostati u okvirima polazne matrice. Poruke o eventualnim greškama štampati na standardni izlaz za poruke o grešci.

Zadatak 82 Septembar, 2002. Napisati program koji sa standardnog ulaza učitava cifre pozitivnog celog broja (kojih nema više od 100, a na ulazu su jedna pored druge tj. između cifara nema praznih mesta) a na standardnom izlazu ispisuje najmanji pozitivan ceo broj zapisan istim ciframa. Rezultat ne sme počinjati cifrom nula.

Zadatak 83 Januar, 2002. Neka su u tekstualnoj datoteci *LAVIRINT* dati podaci o matrici-lavirintu. Prvi red tekstualne datoteke sadrži broj kolona (80) i broj vrsta (25) a u svakom sledećem redu se nalaze podaci o jednoj vrsti matrice: karakier 'Z' označava da odgovara polje matrice predstavlja zid, a karakrer 'P' označava prazan prostor. Napisati program koji na standardnom izlazu prikazuje lavirint učitani iz datoteke ali tako da polja zida prikazuje karakterom 'X' a prazna polja blanko karakterom. Program potom učitava koordinate dve pozicije u lavirintu i utvrđuje da li postoji put kroz lavirint od jedne do druge pozicije (kretanje je moguće samo kroz prazna polja i to u četiri pravca - gore, dole, levo i desno). Ako put postoji program ponovo prikazuje lavirint ali tako da na početnoj poziciji umesto blanko karaktera stoji karakter 'A', na krajnjoj karakter 'B', a na svim ostalim poljima na putu karakrer 'O'. Ako put ne postoji dati odgovarajuću poruku.

Zadatak 84 Nepoznati rok Sa standardnog ulaza se unosi ime datoteke čiji prvi red sadrži dimenziju celobrojne kvadratne matrice *n* ($n > 100$), a ostali redovi elemente matrice (vrstu po vrstu). Formirati niz *b* dimenzije *n* čiji je prvi član suma elemenata glavne dijagonale, drugi suma elemenata na prvoj donjoj dijagonalnoj paraleli (nju čine elementi odmah ispod glavne dijagonale), treći element suma druge donje dijagonalne paralele, itd. Ispisati niz na standardni izlaz. Sve greške štampati na standardni izlaz za greške.

Zadatak 85 Nepoznati rok Sa standardnog ulaza se unose veliki, celi, neoznačeni brojevi sa najviše 100 cifara. Ovih brojeva ima manje od 100 ali njihov broj nije unapred poznat. Napisati program koji sabira ovako unete brojeve i na standardni izlaz ispisuje njihov zbir. Napomena: Svaki broj se unosi u posebnom redu a potrebno je voditi računa o korektnosti ulaznih podataka.