Programiranje I

Beleške sa vežbi

Smer *Informatika*
Matematički fakultet, Beograd

Jelena Tomašević i Sana Stojanović

November 1, 2005

Sadržaj

Čas 1

1.1 LaTeX

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

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 oprerativnim 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 numeriše u skladu sa svojim rednim brojem. Na sličan način dodajemo i podpoglavlja 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	$\backslash textbf\{tekst\}$
Italik	$\backslash emph\{tekst\}$
<u>Podvučen</u>	$\backslash underline\{tekst\}$
$\overline{Nadvučen}$	$\backslash overline\{tekst\}$

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 tebeli:

č	\v c
ć	\' c
đ	\d
ž	\v z

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

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

Ako imamo neko nabrajanje 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 Ω, 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 ω.

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:

$\frac{1}{2}$	\frac{1}{2}	$\vec{a}$	$\vec a$
a^x	a^{x}	log	$\log$
a_n	a_{n}	max	$\max$
ϵ	ϵ	ε	ε
$\forall$	$\forall$	$\exists$	$\exists$

Ako želimo da dodamo *fusnotu* [1] to radimo naredbom \footnote{tekst fusnote}

[1] ovo je fusnota

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.

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{*desctription*}	

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.

Čas 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 $x = b$. Medjutim, 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: $ax = xa$.*

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

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 $aa^n = a^{n+1} = a^n a$.

($\subseteq$): 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 $aaya = ayaa$ 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 $N = \{S\}$,*
$\Sigma = \{a, b\}$,
$P = \{S \to aS \ \ (1^\circ), S \to b \ \ (2^\circ)\}$.

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

Lema 1 *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

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

$a^n b$ ($n \in \mathbf{N}_0$). Reč w se netrivijalno (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, $L(G) \subseteq \{a^n b \mid n \in \mathbf{N}_0\}$.

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

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

$$S \underbrace{\overset{1^\circ}{\Rightarrow} aS \overset{1^\circ}{\Rightarrow} \ldots \overset{1^\circ}{\Rightarrow}}_{n} a^n S \overset{2^\circ}{\Rightarrow} a^n b$$

Zadatak 4 *Odrediti jezik generisan gramatikom $G = (N, \Sigma, P, S)$, gde je $N = \{S\}$,*
$\Sigma = \{a, b\}$,
$P = \{S \to aSb \ \ (1^\circ), S \to e \ \ (2^\circ)\}$.

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

Rešenje:
$G = (N, \Sigma, P, S)$, gde je $N = \{S\}$,
$\Sigma = \{a, b\}$,
$P = \{S \to aab \ \ (1^\circ), S \to aaSb \ \ (2^\circ)\}$
Dokažimo da važi $W = L(G)$.

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

w se može izvesti u gramatici G.

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

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

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

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

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* $N = \{S\}$,
$\Sigma = \{a, b\}$,
$P = \{S \to Sb \ \ (1°), S \to a \ \ (2°)\}$.

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

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

Čas 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 $0, 1, \ldots, 9$ odnosno od 0 do $10 - 1$. Znači u proizvoljnoj bazi B koristićemo 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

$$\delta_n = d_1 d_2 \ldots d_n$$

predstavlja broj

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

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

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

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

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

odatle takođe možemo da zaključimo da je:

$$\tilde{\delta}_{i-1} = \tilde{\delta}_i \text{ div } B, \qquad d_i = \tilde{\delta}_i \text{ mod } B \tag{3.1}$$

Znači, ako pretpostavimo da su cifre broja u sistemu B redom $d_1, \ldots, d_n$ 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 $d_1 \ldots d_n = \delta_n$ 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. $c_i = x$ mod B, $x = x$ 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 i ostatak 0
13 / 2 = 6 i ostatak 1
6 / 2 = 3 i ostatak 0
3 / 2 = 1 i ostatak 1
1 / 2 = 0 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 i ostatak 5
22 / 8 = 2 i ostatak 6
2 / 8 = 0 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 i ostatak 5
11 / 16 = 0 i ostatak 11(heksadekadna cifra B)
Dakle, rešenje će biti broj cije 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:

$$\tilde{\delta} = \sum_{i=1}^{n} d_i * B^{-i} = \frac{1}{B}(\tilde{d_1} + \frac{1}{B}(\tilde{d_2} + \ldots + \frac{1}{B}\tilde{d_n}) \ldots) \tag{3.2}$$

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:

$$\tilde{\delta_i} = \frac{1}{B}(\tilde{d_i} + \tilde{\delta}_{i+1}), \qquad 0 \leq \tilde{\delta_i} < 1$$

Odnosno

$$\tilde{d_i} = \text{trunc}(B * \tilde{d_i})$$

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

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. $x = B * x - d_i$, $i = i + 1$.

[2] pri 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

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.*

$0.867 * 2 = 1.734$, *ceo deo 1*
$0.734 * 2 = 1.468$, *ceo deo 1*
$0.468 * 2 = 0.936$, *ceo deo 0*
$0.936 * 2 = 1.872$, *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

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, pocev 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 $(11110111000011010100 00)_2 = (\ 0011\ 1101\ 1100\ 0011\ 0101\ 0000)_2 = (3DC350)_{16}$

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* $(11010100100)_2$

Čas 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 računaru

- 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

[1] Zasnovano na materijalu "Zapis tekstova u računaru" Filipa Marića

- Standardni karakterski kodovi: Sedamdesetih godina su se pojavile tabele standardnih karakterskih kodova dovoljne za zapis pomenutih karaktera

  Najpoznatiji su

  - 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 međunardonog 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 verijama 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, Ć, Č.

- 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 $\check{Z} < A$ što ne odgovara alfabetskom redosledu unutar kolacione sekvencije. Nakon slova Ž slede velika slova engleske abecede, zatim slova Š(0x5B), Đ(0x5C), Ć(0x5D), Č(0x5E), ž(Ox60), zatim slede mala slova abecede, i na kraju slova š(0x7B), đ(0x7C), ć(0x7D), č(0x7E).

Zadatak 19 *Poređati 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 najvece 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 (obicno 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 višejezični 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 dodadno definišu

- ISO definiše UCS-2 standard koji jednostavno svaki UNICODE karakter prevodi u odgovarajuca 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 agloritma 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*)
- Korespodencija izmedju karaktera i glifova ne mora biti jednoznacna
- Jedan glif može da predstavi više karaktera (ligature)
- Isti karakter može da se predstavlja razlicitim 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!

Čas 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.

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

  Drugo pravilo može da se napiše i kao:

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

- BNF za ceo broj.

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

- BNF za realne brojeve.

```
<RealanBroj>  ::= "-" <NeoznacenRealanBroj>
               | <NeoznacenRealanBroj>;
<NeoznacenRealanBroj> ::= <NeoznacenCeoBroj>
                        | <NeoznacenCeoBroj> "." <NeoznacenCeoBroj>;
<NeoznacenCeoBroj> ::=  <cifra> | <cifra> <NeoznacenCeoBroj>;
<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 rekurziju da bi se izrazile relacije a EBNF koristi iteraciju.

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

- Pravougaone 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 zagarade "{..}".
- 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.

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

- EBNF za ceo broj.

```
<CeoBroj> ::= [ + | - ] <NeoznacenCeoBroj>
<NeoznacenCeoBroj> ::= ( <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 aritmeticki 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 fiksnim skupom iskaznih slova)*

Zadatak 26 *napisati BNF/EBNF za klauzu duzine 5 (nad nekim fiksnim skupom iskaznih slova).*

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

5.1.3 Sintaksni dijagrami

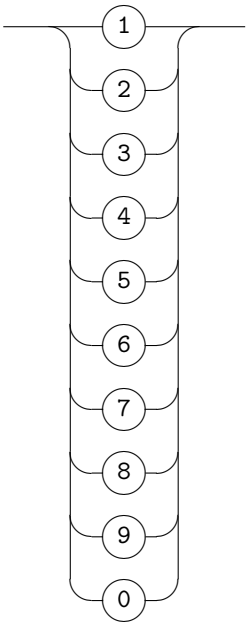
Sintaksni dijagrami predstavljaju grafičku notaciju za predstavljanje kontekstno-slobodnih gramatika. To su dijagrami slični dijagramima toka. Čitanje sintaksnog dijagrama znači kretanje od leve ka desnoj strani prateći strelice. Ono što je bitno to je da oni imaju istu izražajnu moć kao i BNF ili EBNF.

Pogledajmo kako izgledaju sintaksni dijagrami nekoliko već pomenutih gramatika:

- Sintaksni dijagram za `cifra`:

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

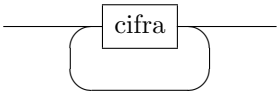
  cifra

- Sintaksni dijagram za `NeoznacenCeoBroj`:

  ```
  <NeoznacenCeoBroj> ::= ( cifra ) +
  ```

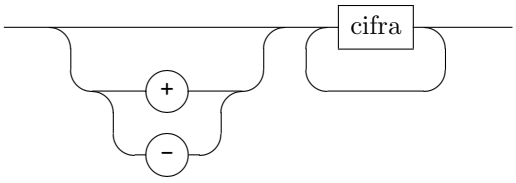
  broj

- Sintaksni dijagram za `CeoBroj`:

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

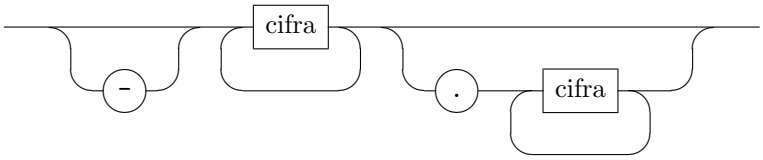
CeoBroj



- Sintaksni dijagram za `RealanBroj`:

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

RealanBroj



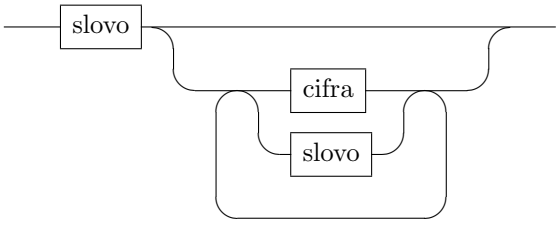
- Sintaksni dijagram za `identifikator`:

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

ili

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

identifikator



Čas 6

Programski jezik C

6.1 Tipovi, operatori i izrazi. Kontrola toka.

[1]

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 odstampa 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!
```

[1] Zasnovano 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`.

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
Progresan 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
Moze i : 5/2.0 = 2.500000 i 5.0/2 = 2.500000
Za promenljive mora kastovanje : 2.500000
```

Primer 15 *Ilustracija prefiksnog i postfiksnog 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\ny = %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
```

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

int main()
{
    int a;
    for  (a = POCETAK; 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;
defalut:
      printf("Uneli ste nesto sto nije paran broj\n");
}
return 0;
}

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

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 proizvode kvadrata svih brojeva od 5 do 35. Nakon svakog petog proizvoda odstampati znak za novi red(for, while).*

Zadatak 34 *Ilustracija korišćenja funkcije za izračunavanje faktorijela celog broja.*

(a) Napisati program koji izračunava faktorijel unetog broja.

(b) Napisati funkciju koja izračunava faktorijel celog broja.

(c) Napisati program koji izračunava faktorijel 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.*