Programiranje i programski jezici

Rad sa karakterima

Stringovi

Karakter

- Tip podataka **`char`** predstavlja karakter.
- Pomoću tipa `char` predstavljamo slova, cifre, specijalne karaktere (`,.;:-+*/!“#$%&` ...), praktično sve što možemo da otkucamo sa tastature.
- Promenljivoj tipa `char` možemo dodeliti karakter koji se navodi unutar apostrofa:

```
char a = '%';
```

- Karakteri se u memoriji zapisuju kao celi brojevi koji zauzimaju 1 bajt. Broj koji odgovara karakteru je zapravo njegova pozicija u kodnoj tabeli (npr. ASCII tabeli).

ASCII tabela kodova karaktera

Kod	Taster ili simbol
32	(razmaknica)
33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<

Kod	Taster ili simbol
61	=
62	>
63	?
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y

Kod	Taster ili simbol
90	Z
91	[
92	\
93	]
94	^
95	_
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v

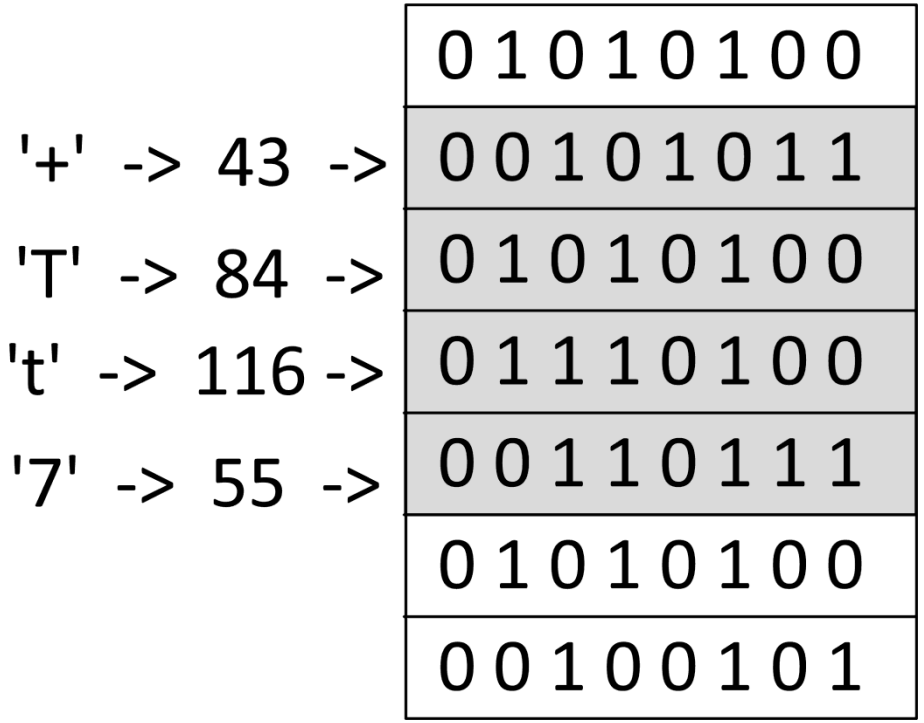
Kod	Taster ili simbol
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~

Kodiranje karaktera

- `char` podaci su celi brojevi u intervalu **–128** do **127**, a nama će biti interesantni samo pozitivni brojevi.
- Numerička vrednost karaktera odgovara njegovoj poziciji u ASCII tabeli. Tako, na primer, karakteru `'+'` odgovara broj 43, slovu `'T'` odgovara broj 84, slovu `'t'` odgovara broj 116, cifri `'7'` broj 55.
- Nas neće interesovati konkretna pozicija pojedinih karaktera. Jedino što treba da uočimo i zapamtimo iz ASCII tabele je sledeće:
  - Slova (mala i velika) su poređana u prirodno rastućem redosledu engleskog alfabeta (A, B, C, D, ..., a, b, c, d, ...),
  - Cifre su poređane u rastući redosled (0, 1, 2, 3, ...),
  - Skupovi malih i velikih slova se ne nadovezuju jedan na drugi (prvo dolaze velika slova, pa nekoliko specijalnih karaktera, pa mala slova).

Nekoliko karaktera u memoriji

Karakteru `'+'` odgovara broj 43
Slovu `'T'` odgovara broj 84
Slovu `'t'` odgovara broj 116
Cifri `'7'` odgovara broj 55.



Šetnja po karakterima

```
#include <stdio.h>
int main()
{
    char a = 'c', b = 'D', c = '5';
    printf("ASCII kod karaktera %c je %d \n\
       ASCII kod karaktera %c je %d \n\
       ASCII kod karaktera %c je %d \n", a, a, b, b, c, c);
    puts("\nSusedni karakteri su:");
    printf("%c %c %c\n", a-1, a, a+1);
    printf("%c %c %c\n", b-1, b, b+1);
    printf("%c %c %c", c-1, c, c+1);
}
```

Štampa

```
ASCII kod karaktera c je 99
ASCII kod karaktera D je 68
ASCII kod karaktera 5 je 53

Susedni karakteri su:
b c d
C D E
4 5 6
```

Karakter kao ceo broj

Prelom stringa na više redova

```
char a = 'c', b = 'D', c = '5';
printf("ASCII kod karaktera %c je %d \n\
ASCII kod karaktera %c je %d \n\
ASCII kod karaktera %c je %d \n", a, a, b, b, c, c);
```

- Specifikator formata za učitavanje i štampanje karaktera je **%c**.
- Kad se karakter odštampa koristeći **%d** specifikator, biće odštampan ASCII kod karaktera.
- Ako želimo da string prelomimo na više redova (obično radi preglednosti), prelom se vrši pomoću kose crte unazad ****.

Funkcija puts

- Funkcija **`puts`** služi za štampanje fiksnog stringa (tzv. string literal), koji je argument funkcije (navodi se unutar zagrada):

```
puts("\nSusedni karakteri su:");
```

- Nakon štampanja teksta, funkcija `puts` prelazi u novi red.
- Funkcija `puts` nema dodatnih argumenata kao `printf`.
- Ako je **`a`** tekući karakter, susednim karakterima možemo pristupiti jednostavno sa **`a+1`**, **`a+2`**, **`a-1`**, **`a-2`** itd.
- U prethodnom primeru, karakter `a` ima vrednost `'c'`, pa će naredba

```
printf("%c %c %c\n", a-1, a, a+1);
```

odštampati karaktere `'b'`, `'c'`, i `'d'`.

Primer 1

Napisati C program koji štampa sva mala slova. U jednom redu štampati 6 slova.

```
#include <stdio.h>

int main()
{
    int i, brojSlova = 0;
    for(i = 'a'; i <= 'z'; i++){
        printf("%c ", i);
        brojSlova++;
        if(brojSlova == 6){
            brojSlova = 0;
            printf("\n");
        }
    }
}
```

Štampa

```
a b c d e f
g h i j k l
m n o p q r
s t u v w x
y z
```

Primer 1

```
#include <stdio.h>

int main()
{
    int i, brojSlova = 0;
    for(i = 'a'; i <= 'z'; i++){
        printf("%c ", i);
        brojSlova++;
        if(brojSlova == 6){
            brojSlova = 0;
            printf("\n");
        }
    }
}
```

Kad se karakter koristi u nekom izrazu, uzima se vrednost njegovog ASCII koda.

Kad brojSlova dostigne vrednost 6, štampa se karakter za prelazak u novi red i resetuje se brojSlova.

Primer 2

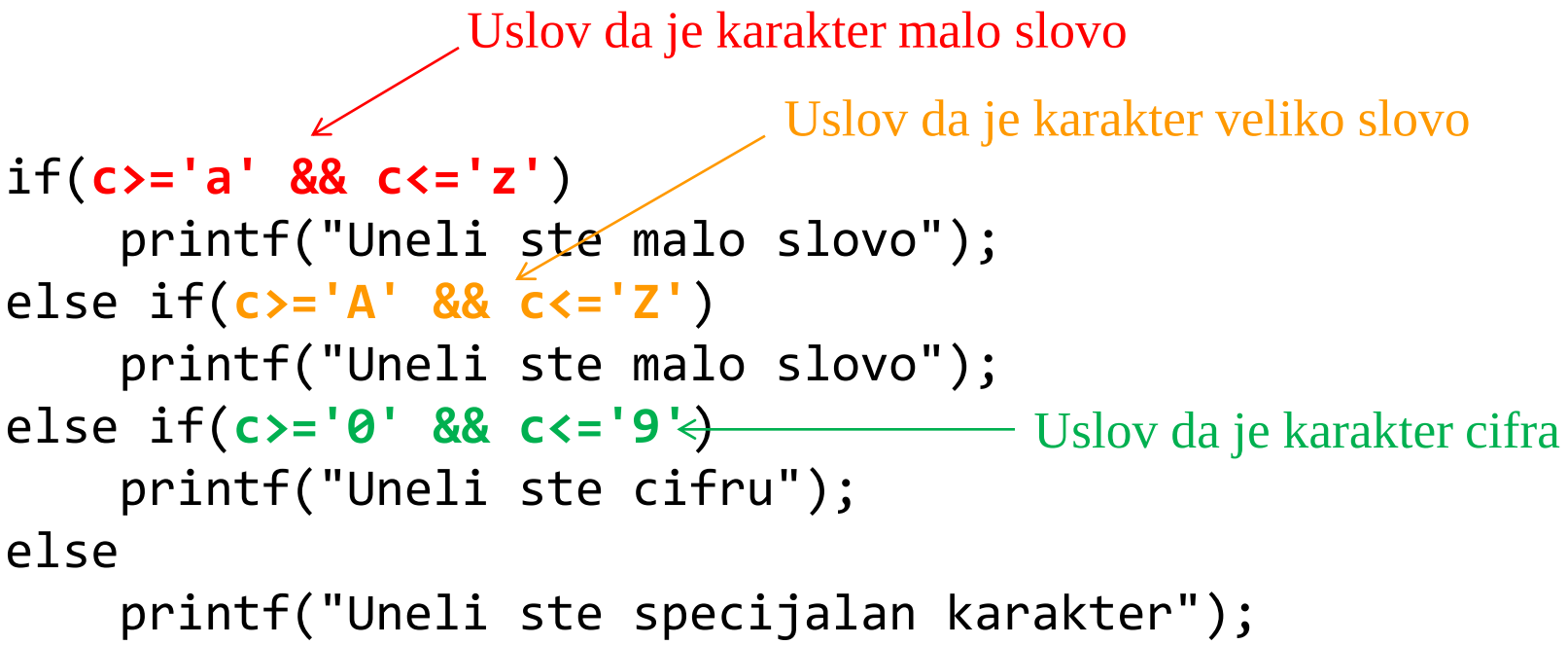
Napisati C program kod koga unosimo jedan karakter. Za uneti karakter, program ispisuje da li je to malo slovo, veliko slovo, cifra ili specijalan karakter.

```c
#include <stdio.h>

int main()
{
    char c;
    printf("Uneti karakter: ");
    scanf("%c", &c);
    if(c>='a' && c<='z')
        printf("Uneli ste malo slovo");
    else if(c>='A' && c<='Z')
        printf("Uneli ste veliko slovo");
    else if(c>='0' && c<='9')
        printf("Uneli ste cifru");
    else
        printf("Uneli ste specijalan karakter");
}
```

Štampa (3 izvršenja)

```
Uneti karakter: 3
Uneli ste cifru

Uneti karakter: %
Uneli ste specijalan karakter

Uneti karakter: Q
Uneli ste veliko slovo
```

Primer 2




```
if(c>='a' && c<='z')
    printf("Uneli ste malo slovo");
else if(c>='A' && c<='Z')
    printf("Uneli ste malo slovo");
else if(c>='0' && c<='9')
    printf("Uneli ste cifru");
else
    printf("Uneli ste specijalan karakter");
```

Primer 3

Napisati C program kod koga unosimo jedan karakter. Program treba da odštampa kvadrat 8x8 koristeći uneti karakter.

```
#include <stdio.h>

int main()
{
    char kar;
    int i, j;
    printf("Uneti karakter: ");
    scanf("%c", &kar);
    for(i = 1; i <= 8; i++){
        for(j = 1; j <= 8; j++)
            printf("%c",kar);
        printf("\n");
    }
}
```

Štampa

```
Uneti karakter: #
########
########
########
########
########
########
########
########
```

Za štampanje jednog karaktera, umesto `printf("%c",kar)` može poslužiti i funkcija **`putchar(kar)`**.

Niz karaktera – string

- Niz karaktera (podataka tipa `char`) se naziva **string**.
- Sa stringovima smo se već upoznali kroz funkcije `printf`:

```
printf("Suma je %d", suma);
```

- String je niz karaktera pod znacima navoda - `"Trebinje 2013"`.
- Za razliku od stringa, pojedinačni karakteri se navode koristeći apostrofe - `'T'`, `'r'`, `'e'`, … , `'1'`, `'3'`.
- String se deklariše kao svaki drugi niz, koristeći uglaste zagrade `[]` i broj elemenata unutar zagrada:

```
char s[30];
```

Inicijalizacija stringa. Terminacioni karakter

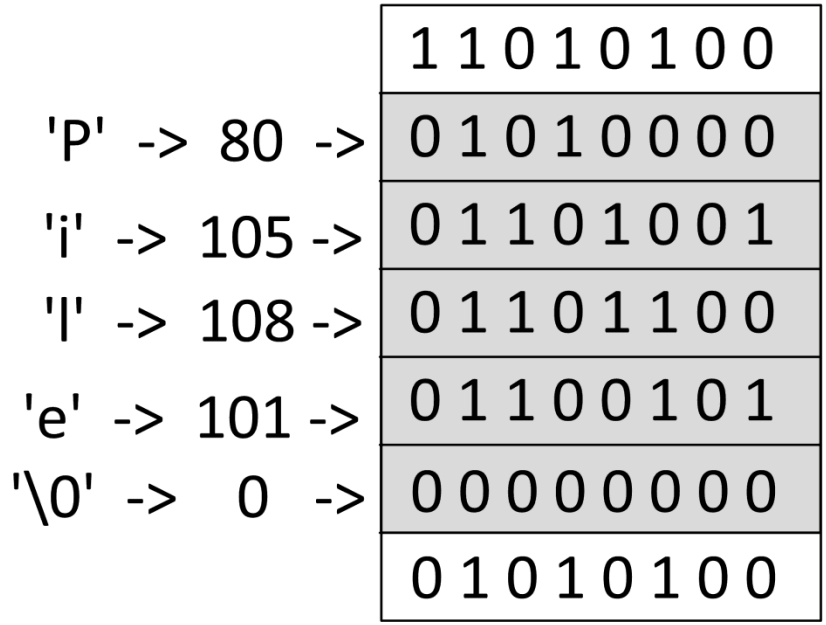
- Najčešći načini inicijalizacije stringa su:

```
char s[30] = "Trebinje 2013";
char s[] = "Trebinje 2013";
```

- Pored pojedinačnih karaktera, stringovi u C-u sadrže dodatni karakter koji se označava kao `'\0'`. Ovaj karakter se nalazi na kraju stringa (nakon svih karaktera) i naziva se **terminacioni karakter**.
- Na osnovu terminacionog karaktera, kompajler zna gde se završava string u memoriji.
- Terminacioni karakter se automatski dodaje na kraj stringa prilikom njegovog učitavanja, bez uticaja programera.
- Terminacioni karakter ima ASCII kod 0, ali to je manje važno.

String u memoriji

Posmatrajmo string `"Pile"`. Ovaj string se u memoriji predstavlja na sledeći način:



Učitavanje i štampanje stringa

- Za razliku od ostalih nizova, koji se učitavaju i štampaju element po element, string se može učitati i odštampati u jednoj naredbi:

```
scanf("%s",s);                  <= učitavanje stringa
printf("String je %s",s);       <= štampanje stringa
```

- Pri učitavanju i štampanju, navodi se samo ime stringa (bez &)!
- Alternativno, string se može učitati pomoću funkcije **`gets`**, i odštampati pomoću funkcije **`puts`**, što je već rečeno.
- Ove dve funkcije se koriste na sledeći način:

```
gets(s);
puts(s);
```

- Funkcije `gets` i `puts` su definisane u zaglavlju stdio.h.

scanf versus gets

- `scanf` učitava karaktere od prve beline (spejs, tab, Enter), dok `gets` učitava do prvog Enter-a. To dalje znači da ako želimo da učitani string sadrži spejs (što je često slučaj), taj string ne treba da učitavamo sa `scanf`.

```
#include <stdio.h>

int main()
{
    char s[30];
    printf("Uneti string: ");
    gets(s);
    printf("Ucitani string je %s\n",s);
    printf("Uneti string: ");
    scanf("%s",s);
    printf("Ucitani string je %s\n",s);
}
```

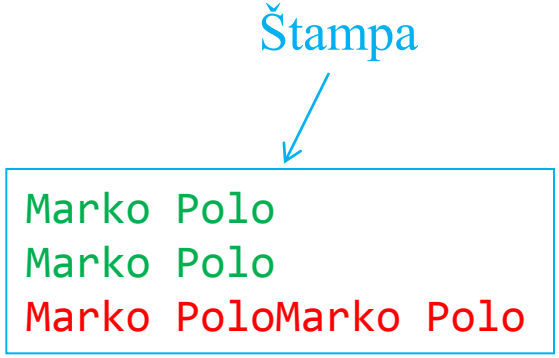
Štampa

```
Uneti string: Marko Polo
Ucitani string je Marko Polo
Uneti string: Marko Polo
Ucitani string je Marko
```

printf versus puts

- `printf` štampa string i ne prelazi u drugi red (osim ako se to ne naznači sa `\n`). String koji se štampa može da sadrži spejsove i ceo string će biti odštampan.
- `puts` štampa string i automatski prelazi u novi red.

```
#include <stdio.h>

int main()
{
    char s[30] = "Marko Polo";
    puts(s);
    puts(s);
    printf("%s",s);
    printf("%s",s);
}
```



Funkcije za rad sa stringovima

- Biblioteka **`string.h`** sadrži korisne funkcije za rad sa stringovima.
- Jedna on najčešće korišćenih je **`strlen`**, koja vraća dužinu stringa, tj. broj karaktera stringa bez terminacionog karaktera.
- **`strcpy(str1,str2)`** kopira string **`str2`** u string **`str1`**.
- **`strcat(str1,str2)`** nadovezuje **`str2`** na string **`str1`**.
- **`strcmp(str1,str2)`** poredi stringove **`str1`** i **`str2`**.
  - ako je **`str1 > str2`**, **`strcmp`** vraća pozitivan broj;
  - ako je **`str1 < str2`**, **`strcmp`** vraća negativan broj;
  - ako je **`str1 = str2`**, **`strcmp`** vraća 0.
- **`strcmp`** poredi stringove leksikografski – manji su oni stringovi koji u leksikonu dolaze ranije.
- Na primer, string `"baba"` je manji od stringova `"deda"` i `"babaroga"`, ali je veći od stringova `"ali-baba"` i `"bab"`.

Obilazak stringa

- String možemo obići kao i svaki drugi niz ako mu znamo dužinu.
- Drugi način je da se krećemo kroz string sve dok ne naiđemo na terminacioni karakter.

Prvi način

```
duzina = strlen(s);
for(i=0; i < duzina; i++){
    naredbe
}
```

Drugi način

```
for(i=0; s[i] != '\0'; i++){
    naredbe
}
```

Primer 4

Napisati C program koji učitava string S i određuje i štampa koliko u stringu ima slova, cifara i specijalnih karaktera.

```
#include <stdio.h>

int main()
{
    char s[30];
    int i, brSlova=0, brCif=0, brSpec=0;
    printf("Uneti string: ");
    scanf("%s", s);
    for(i=0; s[i]!='\0'; i++){
        if((s[i]>='a' && s[i]<='z') || (s[i]>='A' && s[i]<='Z'))
            brSlova++;
        else if(s[i]>='0' && s[i]<='9')
            brCif++;
        else
            brSpec++;
    }
    printf("Slova %d, cifara %d i specijalnih %d\n", brSlova, brCif, brSpec);
}
```

Štampa

```
Uneti string: Abcd1234^&*(
Slova 4, cifara 4 i specijalnih 4
```

Uslov za slova

Primer 5

Napisati C program koji učitava karakter `K` i prirodan broj `N`, i koji formira i štampa string dobijen nadovezivanjem karaktera `K` `N` puta. Pretpostaviti da je N<30.

```
#include <stdio.h>

int main()
{
    char s[30], K;
    int i, N = 0;
    printf("Uneti karakter K i broj N: ");
    scanf("%c%d", &K, &N);
    for(i=0; i<N; i++)
        s[i] = K;

    s[i] = '\0';
    printf("String je %s\n", s);
}
```

Ručno „zatvaranje“ stringa

Štampa

```
Uneti karakter K i broj N: $
10
String je $$$$$$$$$$
```

Primer 6

Napisati C program koji učitava dva stringa S i T, i koji na kraj stringa S nadovezuje samo cifre iz stringa T. Štampati izmenjeni string S.

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

int main()
{
    char S[50], T[30];
    int i, j;
    printf("Uneti stringove: ");
    scanf("%s%s", S, T);
    i = strlen(S);
    for(j = 0; T[j] != '\0'; j++){
        if(T[j]>='0' && T[j]<='9'){
            S[i] = T[j];
            i++;
        }
    }
    S[i] = '\0';
    printf("Novi string je %s", S);
}
```

Štampa

```
Uneti stringove: Trema
Odiseja2001
Novi string je Trema2001
```

Primer 6

Krećemo se od poslednjeg karaktera stringa, tj. prva pozicija u stringu `S` na kojoj upisujemo cifru iz stringa `T` je pozicija terminacionog karaktera.

```
i = strlen(S);
for(j = 0; T[j] != '\0'; j++){
    if(T[j]>='0' && T[j]<='9'){
        S[i] = T[j];
        i++;
    }
}
S[i] = '\0';
```

Kad upišemo karakter `T[j]` na kraj stringa `S`, pomeramo se u stringu `S` sa `i++`, radi upisa narednog karaktera.

Ručno „zatvaranje“ stringa

Primer 7

Napisati C program koji učitava string S i određuje i štampa sumu cifara tog stringa. Na primer, ako se unese `S="12abc45%#"`, program treba da odštampa broj 12 = 1+2+4+5.

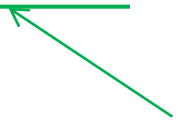
```
#include <stdio.h>

int main()
{
    char s[30];
    int i, sumCif = 0;
    printf("Uneti string: ");
    scanf("%s", s);
    for(i=0; s[i]!='\0'; i++){
        if(s[i]>='0' && s[i]<='9')
            sumCif += s[i] - '0';
    }
    printf("Suma cifara je %d\n", sumCif);
}
```

Štampa

```
Uneti string: 123RTFG^%9
Suma cifara je 15
```

Primer 7

```
for(i=0; s[i]!='\0'; i++){
    if(s[i]>='0' && s[i]<='9')
        sumCif += s[i] - '0';
}
```



Dobijanje cifre od odgovarajućeg karaktera:

Ako je `s[i]='0'` onda je `s[i]-'0' = 0`
Ako je `s[i]='1'` onda je `s[i]-'0' = 1`
...
Ako je `s[i]='9'` onda je `s[i]-'0' = 9`

Cifre su u ASCII tabeli poređane u rastućem redosledu, jedna za drugom.

Primer 8

Napisati C program koji učitava string S i proverava da li taj string može predstavljati binarni zapis broja. Štampati odgovarajuću poruku.

```
#include <stdio.h>

int main()
{
    char s[30];
    int i, jesteBin = 1;
    printf("Uneti string: ");
    scanf("%s", s);
    for(i=0; s[i]!='\0'; i++){
        if(s[i]!='0' && s[i]!='1')
            jesteBin = 0;
    }
    if(jesteBin == 1)
        printf("Jeste binaran broj");
    else
        printf("Nije binaran broj");
}
```

Štampa (2 izvršenja)

```
Uneti string: 110011101
Jeste binaran broj

Uneti string: 1110101 2
Nije binaran broj
```

Primer 8

```
#include <stdio.h>

int main()
{
    char s[30];
    int i, jesteBin = 1;
    printf("Uneti string: ");
    scanf("%s", s);
    for(i=0; s[i]!='\0'; i++){
        if(s[i]!='0' && s[i]!='1')
            jesteBin = 0;
    }
    if(jesteBin == 1)
        printf("Jeste binaran broj");
    else
        printf("Nije binaran broj");
}
```

Na početku pretpostavimo da jeste binarni broj. Promenljiva `jesteBin` je indikator binarnosti.

Ako bar jedna cifra nije ni `0` ni `1`, onda broj nije binaran i menjamo `jesteBin` u `0`. Nakon toga se više `jesteBin` ne vraća na `1`.

Zadaci za vežbu

1. Napisati C program koji učitava dva cela broja, **A** i **B**, i karakter **K** koji predstavlja aritmetičku operaciju, tj. može biti '+', '-', '*', '/' i '%'. Program treba da na brojevima A i B izvrši operaciju definisanu karakterom K i da odštampa rezultat. Na primer, ako smo uneli A=5, B=7 i K='*', program treba da odštampa broj 35.
2. Napisati C program koji učitava string S i proverava da li taj string može predstavljati oktalni zapis broja (jedini dozvoljeni karakteri su cifre 0,1,2...,7). Štampati odgovarajuću poruku.
3. Napisati C program koji učitava string S i proverava da li taj string može predstavljati ime. String predstavlja ime ako mu je prvi karakter veliko slovo, a svi ostali karakteri mala slova (npr. "Janko", "Pegla"). Štampati odgovarajuću poruku.

Zadaci za vežbu

4. Napisati C programe koji štampaju oblike prikazane ispod. Programe modifikovati tako da rade za proizvoljnu dužinu stranica kvadrata i trougla.