

Programiranje (C)

4. predavanje

Saša Singer

`singer@math.hr`

`web.math.hr/~singer`

`www.math.hr/~singer`

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Osnovni elementi jezika kroz primjere (kraj):
 - Funkcije i nizovi znakova.
 - Znakovni nizovi — stringovi.

Početak sistematskog prolaza kroz pravila jezika C.

- Konstante i varijable:
 - Skup znakova jezika.
 - Komentari.
 - Ključne riječi.
 - Osnovni tipovi podataka.
 - Deklaracije varijabli.
 - Konstante.
 - Enumeracije — pobrojani tipovi.

Primjer 8 — nizovi znakova, argumenti funkcije

Primjer 8. Treba napisati funkciju koja uzima niz znakova i u njemu nalazi prvi samoglasnik. Funkcija treba vratiti nađeni samoglasnik i mjesto u nizu gdje je nađen. Ako u danom nizu nema samoglasnika, umjesto mjesta u nizu vraća se -1 .

- Funkcija treba vratiti dvije vrijednosti.
- Kroz `return` naredbu može se vratiti samo jedna vrijednost, a druga se mora vratiti kroz argument funkcije.
 - Argument funkcije tada mora biti pokazivač na izlaznu vrijednost.
- Funkcija ne može vratiti polje kao vrijednost funkcije (kroz `return`).

Primjer 8 (nastavak)

Izlazna vrijednost funkcije je `mjesto` tipa `int`, radi lakšeg testa na `-1`, ako ne nađemo samoglasnik.

Nađeni `samoglasnik` onda moramo vratiti kroz argument, kao `pokazivač` na taj znak.

Zaglavlje (signatura) funkcije je:

```
int trazi(char linija[], int n, char *ps);
```

gdje je

- `linija[]` = niz znakova među kojima se traži samoglasnik,
- `n` = broj znakova u nizu `linija[]`,
- `ps` = pokazivač na nađeni samoglasnik.

Primjer 8 (nastavak)

Za test na samoglasnike trebamo **znakovne konstante** za znakove **a, e, i o, u**.

- **Znakovne konstante** dobivaju se stavljanjem **znaka** u **jednostruke** navodnike.

```
'a', 'e', 'i', 'o', 'u', '0', '9', '\n'
```

Početak programa i cijela definicija funkcije **trazi**:

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

```
int trazi(char linija[], int n, char *ps)
{
    int i;
```

Primjer 8 (nastavak)

```
char c;

for (i = 0; i < n; i++)
{
    c = linija[i];
    if (c == 'a' || c == 'e' || c == 'i'
        || c == 'o' || c == 'u')
    {
        *ps = c;
        return i;
    }
}
return -1;
}
```

Prijenos argumenata

U raznim programskim jezicima **argumenti** (parametri) se prenose u potprograme na dva načina:

- **po vrijednosti** (“call by value”), ili
- **po adresi** (“call by reference”).

Potprogrami u **C**-u su **funkcije** i vrijede sljedeća **pravila** za prijenos argumenata.

Argumenti (osim **polja**) se funkciji prenose **po vrijednosti**. To znači da funkcija dobiva **kopije** stvarnih argumenata u lokalne varijable. Posljedica:

- **Nema promjene vrijednosti** stvarnog argumenta, čak i kad je taj argument **varijabla**!

Sve promjene su u lokalnoj varijabli, a ne “vani”.

Prijenos argumenata (nastavak)

Iznimka su polja koja se prenose po adresi.

- Pri pozivu funkcije, argument tipa polje konvertira se u pokazivač na prvi element polja.
- Ime polja je taj pokazivač i on se prenosi po vrijednosti.

Posljedica:

- Funkcija može dohvatiti i promijeniti svaki element polja (tu nema kopije!).

Napomena: Već smo rekli da deklaracija argumenata tipa polje ne treba dimenziju polja (nema kopiranja i rezervacije memorije u funkciji).

```
char linija[];
```

Prijenos argumenata (nastavak)

Funkcija koja treba **promijeniti** vrijednost **varijable** u glavnom programu (koja nije cijelo polje)

- mora dobiti **pokazivač** na tu **varijablu** i
- koristiti operator **dereferenciranja** za **promjenu** sadržaja (vrijednosti varijable).

Pokazivač se prenosi **po vrijednosti** i **ne može** mu se promijeniti vrijednost (osim lokalno, u funkciji).

- Operatorom ***** **možemo** promijeniti **vrijednost** varijable na koju pokazivač **pokazuje**,
- tj. **sadržaj** na adresi koja je spremljena u pokazivaču.

Prijenos argumenata (nastavak)

Zato operator **dereferenciranja** može stajati na **lijevoj strani** izraza pridruživanja.

```
*ps = c;
```

Ovdje je:

- **ps** = **pokazivač** na traženi samoglasnik (ta adresa će stići iz glavnog programa),
- ***ps** = **sadržaj** (tipa **char**) na toj adresi, kojeg postavljamo u funkciji na nađeni samoglasnik **c** (ako ga nađemo).

Logički operatori

Logički operatori u jeziku C:

Operator	Značenje
&&	logičko i
	logičko ili
!	logička negacija (unarna)

U C-u ne postoji poseban tip za logičke vrijednosti, već se one prikazuju kao cijeli brojevi.

Standardna značenja:

● $0 \iff$ laž (FALSE),

● $\neq 0 \iff$ istina (TRUE). Najčešće se koristi vrijednost 1.

Logički operatori daju rezultat 0 ili 1.

Glavni program za Primjer 8

```
int main(void) // Glavni program
{
    /* Inicijalizacija polja
       u viticastim zagradama */

    char ime[] = { 'P', 'r', 'o', 'g', 'r',
                  'a', 'm', 's', 'k', 'i',
                  ' ', 'j', 'e', 'z', 'i',
                  'k', ' ', 'C', '.' };

    char znak;
    int no;
```

Glavni program za Primjer 8 (nastavak)

```
no = trazi(ime, 19, &znak);
if (no != -1) {
    printf("Prvi samoglasnik = %c\n", znak);
    printf("Nalazi se na mjestu = %d\n", no);
}
else
    printf("Nema samoglasnika.\n");
return 0;
}
```

Ako u deklaraciji polja odmah **inicijaliziramo** cijelo polje, **ne treba** navesti dimenziju. Prevoditelj će ju sam **izračunati**.

Primjer: deklaracija za polje **ime** (19 znakova).

Nizovi znakova i znakovni nizovi (stringovi)

Niz znakova je bilo koji niz (polje) znakova.

```
char poruka[128]; // poruka je niz od 128 zn.
```

Znakovni niz ili `string` je niz (polje) znakova koji završava nul-znakom `'\0'`. Uloga tog znaka je da označava kraj niza. Svaki niz znakova napisan unutar dvostrukih navodnika je konstantan znakovni niz (string). Primjer:

```
"Dobar dan.\n"
```

sastoji se od 12 znakova

```
D o b a r   d a n . \n \0
```

Primjer 9 — stringovi

Primjer 9. Treba napisati program koji učitava niz znakova omeđen bjelinama, računa broj numeričkih znakova u nizu i ispisuje učitani niz u obrnutom poretku.

- Znak konverzije `%s` služi za čitanje *stringa* omeđenog bjelinama i za pisanje *stringa*.
- Numerički znakovi su `'0'`, ..., `'9'` (znakovi za dekadске znamenke).

```
#include <stdio.h>
#include <string.h>
#define MAX 128

void inv(char[]);    /* Prototip */
```

Zaglavlja i direktive

Ovdje koristimo dvije standardne datoteke zaglavlja.

- Glavni program učitava datoteku zaglavlja `string.h` zbog funkcije `strlen` (**duljina stringa**).
- Funkcija `strlen` daje broj znakova u stringu, **ne računajući** zadnji nul-znak (`'\0'`).

Naredba

```
#define MAX 128
```

je tzv. **preprocesorska** direktiva kojom se uvodi **simboličko ime MAX**.

- **Preprocesor** svako pojavljivanje simbola `MAX` zamjenjuje sa `128` (i to doslovno, “riječ” za “riječ”).

Primjer 9 (nastavak)

```
int main(void)          /* Glavni program */
{
    char niz_znakova[MAX], c;
    int i, brojac = 0;

    scanf("%s", niz_znakova);

    i = 0;
    while ((c = niz_znakova[i]) != '\0')
    {
        if (c >= '0' && c <= '9') ++brojac;
        i++;
    }
}
```

Primjer 9 (nastavak)

```
printf("Broj numerickih znakova = %d\n",  
      brojac);  
inv(niz_znakova);  
printf("%s\n", niz_znakova);  
return 0;  
}
```

Izraz pridruživanja

Komad kôda:

```
i = 0;
while ((c = niz_znakova[i]) != '\0')
{
    if (c >= '0' && c <= '9') ++brojac;
    i++;
}
```

koristi tzv. *izraz pridruživanja* u `while` petlji.

- **Naredba pridruživanja** (ili dodjeljivanja vrijednosti) `c = niz_znakova[i]` je ujedno i *izraz*.
- **Vrijednost** tog *izraza* je ona vrijednost koja se **pridružuje** lijevoj strani, **nakon** izvršenja te naredbe.

Izraz pridruživanja (nastavak)

Zagrade oko tog izraza trebaju zbog prioriteta operatora.

- Operator pridruživanja = ima niži prioritet od !=.

Ekvivalentni kôd, bez izraza pridruživanja je:

```
i = 0;
c = niz_znakova[i];
while (c != '\0')
{
    if (c >= '0' && c <= '9') ++brojac;
    i++;
    c = niz_znakova[i];
}
```

Primjer 9 (nastavak)

```
/* INVERTIRANJE NIZA */

void inv(char s[])
{
    int c, i, j;

    for (i = 0, j = strlen(s)-1; i < j; i++, j--)
    {
        c = s[i];
        s[i] = s[j];
        s[j] = c;
    }
}
```

Prošireni oblik for petlje

Napomena: Prvi dio `for` petlje (inicijalizacija) i zadnji dio (pomak brojača) mogu imati **više naredbi**.

- Višestruke inicijalizacije i višestruka povećanja (smanjenja) brojača u `for` petlji odvajaju se **zarezom**.

Zadatak. Preuredite funkciju `trazi` i program iz Primjera 8. tako da rade sa stringovima.

Zadatak. Provjeru da li je znak `c` numerički možemo napraviti funkcijom `isdigit(c)` iz standardne biblioteke funkcija za provjeru znakova. Treba uključiti `<ctype.h>`. Isprobajte i ostale funkcije za provjeru znakova!

Dodatni primjeri

Domaća zadaća. Progleđajte pažljivo primjere u prvom poglavlju knjige KR2.

● Vrlo su **instruktivni** u smislu **tehnika programiranja**.

Posebno se isplati pogledati **odjeljak 1.5** (i nadalje), gdje je

● niz primjera o **obradi “teksta”** (ulaz i izlaz znakova).

Konstante i varijable

Skup znakova

Programski jezik **C** koristi sljedeći skup znakova:

- velika i mala slova engleske abecede **A-Z** i **a-z**,
- znamenke **0-9**,
- specijalne znakove:

+	-	*	/	=	%	&	#
!	?	^	"	'	~	\	
<	>	()	[]	{	}
:	;	.	,	-	(bjelina)		

Pod **bjelinom** se podrazumijeva, osim same **bjeline** (blanka), horizontalni i vertikalni tabulator te znak za prijelaz u novi red.

Komentari

Program treba **komentirati** radi lakšeg razumijevanja njegovog funkcioniranja.

- Prevoditelj **ignorira** komentare pri prevođenju!

Pravila za pisanje **komentara**:

- Komentar započinje s **/*** i završava s ***/**.
- Komentar može sadržavati više linija teksta.

```
/*      Ovo je
      komentar.  */
```

Komentari (nastavak)

Tipične **greške** u pisanju **komentara**.

- Ako je **ispušten** jedan graničnik, može se dogoditi **gubitak kôda**.

```
/*      Ovo je prvi komentar.           Nezatvoren!!!  
x = 72.0;  
/*      Ovo je drugi komentar. */
```

Prevoditelj **ignorira** tekst do prvog znaka ***/**.

Posljedica: Dodjeljivanje

```
x = 72.0;
```

je **dio komentara**.

Komentari (nastavak)

- Nije dozvoljeno pisati komentar **unutar** komentara — greška.

```
/*  
x = 72.0;    /*  Inicijalizacija  */  
y = 31.0;  
*/
```

Drugi komentar **nema** početak **/***.

Komentari (nastavak)

- C++ tip komentara (standard C99) nedozvoljen je u starijim prevoditeljima.
- Takav komentar počinje s // i završava krajem linije.

```
x = 2.17;    // Inicijalizacija
```

Identifikatori

- Identifikatori su imena koja se pridružuju različitim elementima programa, na primjer,
 - varijablama, poljima i funkcijama.
- Sastoje se od slova i brojeva (znamenki), s tim da prvi znak mora biti slovo.
- Velika i mala slova se razlikuju.
- Znak _ (donja crta) smatra se slovom.
- Duljina identifikatora je proizvoljna, ali prevodilac nije dužan razlikovati identifikatore koji su isti na prvih 6–63 mjesta (ovisno o standardu i vrsti varijable).

Primjeri identifikatora

- **Ispravno** napisani identifikatori:

```
x,      y13,      sum_1,      _temp,  
names,  Pov1,      table,      TABLE
```

- **Neispravno** napisani identifikatori:

```
3dan,      /* prvi znak je broj */  
"x",      /* nedozvoljeni znak " */  
ac-dc     /* nedozvoljeni znak - */
```

Ključne riječi

- Ključne riječi imaju posebno značenje u jeziku i ne mogu se koristiti kao identifikatori.

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

Osnovni tipovi podataka

- **int**: cjelobrojni podatak. Tipično zauzima 4 bajta.
- **char**: znakovni podatak. Sadržava jedan znak. Tipično zauzima 1 bajt.
- **float**: broj s pomičnim zarezom (floating-point) u jednostrukoj preciznosti. Tipično zauzima 4 bajta. (IEEE single).
- **double**: broj s pomičnim zarezom (floating-point) u dvostrukoj preciznosti. Tipično zauzima 8 bajta. (IEEE double).

Kratki i dugi tip `int`

- Cjelobrojni tip `int` može se modificirati pomoću kvalifikatora `short` i `long`. Tako dobivamo nove cjelobrojne tipove.
- Cjelobrojni tipovi za brojeve s predznakom:
 - `short int` ili kraće `short`: “kratki” cjelobrojni podatak. U memoriji zauzima manje mjesta od `int`, pa mu je manji raspon prikazivih cijelih brojeva.
 - `long int` ili kraće `long`: “dugi” cjelobrojni podatak. U memoriji zauzima više mjesta od `int`, pa mu je veći raspon prikazivih cijelih brojeva.

Tipovi za brojeve bez predznaka

- Cjelobrojne tipove za brojeve **bez predznaka** dobivamo upotrebom **kvalifikatora unsigned**.
- Oni zauzimaju **isti** memorijski prostor kao osnovni tipovi podataka (**int**, **short**, **long**), a mogu reprezentirati **samo nenegativne** cijele brojeve.
- Oni pokrivaju približno **dvostruko veći** raspon **pozitivnih** cijelih brojeva od osnovnih tipova. (Ponoviti prikaz iz UuR!)
- Cjelobrojni tipovi za brojeve **bez predznaka**:
 - **unsigned int** ili kraće **unsigned**,
 - **unsigned short int** ili kraće **unsigned short**,
 - **unsigned long int** ili kraće **unsigned long**.

Još o cjelobrojnim tipovima

- C propisuje samo **minimalnu preciznost** (tj. **duljinu**) pojedinih cjelobrojnih tipova:
 - tip **int** mora imati najmanje **16** bitova, a
 - tip **long** mora imati najmanje **32** bita.
- Operator **sizeof** (piše se kao funkcija) daje broj **bajtova** rezerviranih za prikaz vrijednosti odgovarajućeg **tipa**. Vrijedi:

```
sizeof(short) <= sizeof(int) <= sizeof(long)
```

- Datoteka zaglavlja **<limits.h>** sadrži simboličke konstante za **minimalne** i **maksimalne** dozvoljene vrijednosti pojedinih **cjelobrojnih** tipova.

Realni tipovi

- `long` se može primijeniti i na **realne** tipove podataka.
 - `long float` je isto što i `double`, a
 - `long double` ima **četverostruku** preciznost (ako postoji).
- Datoteka zaglavlja `<float.h>` sadrži **simboličke konstante** koje daju različite **informacije** o **realnim** tipovima podataka.

Deklaracija varijable

Deklaracija određuje **ime** i **tip varijable**. Ima oblik:

```
tip ime;
```

gdje je **tip** **tip** varijable, a **ime** njezino **ime**.

Primjer:

```
int a, b;  
unsigned c;  
char d;
```

Deklaracija varijable (nastavak)

Varijable **istog tipa** moguće je deklarirati u **istoj** deklaraciji **tipa**, a varijable se odvajaju **zarezom**.

Primjer:

```
short a, b, c;
```

Svaku od tih varijabli možemo deklarirati i u **zasebnoj** deklaraciji **tipa**:

```
short a;  
short b;  
short c;
```

Deklaracija polja

Polje je niz varijabli istog tipa indeksiranih cjelobrojnim indeksom u rasponu od 0 do $n - 1$, gdje je n broj elemenata polja.

Deklaracija polja ima oblik:

```
tip ime[dimenzija];
```

gdje je:

- `tip` podataka svakog elementa polja,
- `ime` je ime polja (zajedničko ime svih elemenata), a
- `dimenzija` je broj elemenata polja.

Pojedini elementi polja razlikuju se po indeksu koji se piše unutar uglatih zagrada.

Deklaracija polja (nastavak)

Primjer:

```
float vektor[10];
```

Elementi polja:

```
vektor[0], vektor[1], ..., vektor[9].
```

Svaki element je varijabla tipa `float`.

Deklaracija pokazivača

Pokazivači su **varijable** koje sadrže **adrese** drugih varijabli (nekog tipa).

Deklaracija pokazivača:

```
tip_p *ime;
```

gdje je:

- **ime** ime pokazivača (varijable), a
- ***** označava da identifikator **ime** **nije** varijabla tipa **tip_p**, nego **pokazivač** na varijablu tipa **tip_p** (tj. sadrži **adresu** varijable tipa **tip_p**).

Za lakše čitanje: **tip_p *ime** — kad **dereferenciramo** **ime** dobijemo objekt tipa **tip_p**.

Deklaracija pokazivača (nastavak)

Deklaracije **varijabli** nekog **tipa** i **pokazivača** na **isti tip** mogu se pisati u jednom retku:

```
float u, *pu;
```

ili u više redaka:

```
float u;  
float *pu;
```

Cjelobrojne konstante

Cjelobrojne konstante mogu biti zapisane u **tri** brojevnih sustava: **decimalnom** (baza 10), **oktalnom** (baza 8) i **heksadecimalnom** (baza 16).

Decimalne (dekadske) konstante:

- znamenke 0-9, dozvoljen predznak - ili +,
- ako konstanta ima više od jedne znamenke (bar **dvije**), **prva** znamenka **nije 0** (služi za ostale baze).
- Decimalna točka **nije dozvoljena**.

Primjer:

0, 1, 234, -456, 99999

Cjelobrojne konstante (nastavak)

Oktalne konstante:

- znamenke 0-7, dozvoljen predznak - ili +,
- prva znamenka uvijek 0.

Primjer:

0, 01, -0651, 077777

Cjelobrojne konstante (nastavak)

Heksadecimalne konstante:

- znamenke 0-9, dozvoljen predznak - ili +,
- mala slova a-f ili velika slova A-F,
- značenje slova: a = 10, b = 11, ..., f = 15,
- uvijek počinju s 0x ili 0X.

Primjer:

0x0, 0x1, -0x7FFF, 0xabcd, 0XABCD

Cjelobrojne konstante (nastavak)

Ako specijalno **ne navedemo** drugačije, onda konstanta ima tip **int** (svi prošli primjeri).

Konstante **ostalih** cjelobrojnih tipova definiraju se **dodavanjem sufiksa** (na kraju konstante).

Konstanta tipa:

- **long** formira se tako da se na kraj cjelobrojne konstante doda slovo **L** (veliko ili malo),
- **unsigned** formira se dodavanjem slova **U** (veliko ili malo), **nema** prednaka,
- **unsigned long** formira se dodavanjem slova **U** i **L** (veliko ili malo, u bilo kojem poretku), **nema** prednaka.

Cjelobrojne konstante (nastavak)

Primjeri:

```
500000U      /* unsigned (decimalna) */
123456789L   /* long (decimalna) */
123456789ul  /* unsigned long (decimalna) */
123456789LU /* unsigned long (decimalna) */
01234561     /* long (oktalna) */
0X50000U     /* unsigned (heksadecimalna) */
```
