Programiranje 1
Programski jezik C

5. čas

FUNKCIJE

- Veliki računski zadaci mogu se razbiti u manje delove i time se omogućava ljudima da iskoriste ono što su neki drugi već uradili, umesto da počinju sve od početka. Odgovarajuće funkcije skrivaju detalje postupka od delova programa I time čine ceo program jasnijim i jednostavnijim za menjanje.
- Prilikom definicije funkcije navodi se:

  – tip povratne vrednosti funkcije (ako se ne navede podrazumeva se int)

  – ime funkcije

  – lista argumenata

  – telo funkcije.

FUNKCIJE

- U deklaraciji (prototipu) funkcije je sve isto kao i u definiciji, osim što izostaje telo funkcije. (Obratiti pažnju gde treba koristiti deklaraciju a gde definiciju funkcije)
- Primer definicije funkcije:

```
tip_rezultata ime_funkcije (tip param1, tip param2, ..., tip paramN)
{

/*telo funkcije*/

}
```

- Primer deklaracije funkcije:

```
tip_rezultata ime_funkcije (tip param1, tip param2, ..., tip paramN);
```

POZIVANJE FUNKCIJE

Poziv funkcije u programu se ostvaruje navođenjem:

– imena funkcije

– liste stvarnih argumenata funkcije

npr. faktorijel(5);

- Mehanizam za vraćanje vrednosti iz funkcije predstavlja naredba return

  return izraz;

- Tip izraza se evenutualno konvertuje u tip rezultata.

VOID

Iza return može i da se ne stavi ništa, ali u tom slučaju se ni jedna vrednost ne vraća pozivaocu.

• Funkcija koja nema povratnu vrednost deklariše se da ima povratni tip **void**.

• Slično, ako funkcija nema argumente, u deklaraciji se umesto liste argumenata navodi void.

FUNKCIJE - primer

- Napisati funkciju koja računa zbir dva cela broja I program koji testira rad ove funkcije.

```
#include <stdio.h>

/* Definicija funkcije */

int zbir (int a, int b) {return a+b;}

int main() {

/* Poziv funkcije */

printf("%d\n", zbir(3,5));

return 0;

}
```

FUNKCIJE – primer deklaracije i definicije funkcije

Uraditi isti primer, u kome se domonstrira razlika između definicije i deklaracije funkcije

- Deklaracija funkcije može da stoji nezavisno od definicije funkcije.
- Deklaracija je neophodna u situacijama kada se definicija funkcije navodi nakon upotrebe date funkcije u kodu.

```
#include <stdio.h>

/* Deklaracija funkcije zbir() */
int zbir(int, int);              /* int zbir(int a, int b); */

int main() {
/* Poziv funkcije */
printf("%d\n", zbir(3,5));
return 0;
}
/* Definicija funkcije */
int zbir(int a, int b) {return a+b;}
```

FUNKCIJE – zadaci za vežbu

- Napisati sledeće funkcije kao i programe koji ih testiraju

  – maksimum dva broja

  – faktorijel datog broja

  – n-ti stepen realnog broja x (omogu ćiti da funkcija radi i za negativne stepene)

  – ojlerova funkcija datog broja (za zadati broj računamo koliko ima brojeva koji su veći ili jednaki od 1, a manji od datog broja, a u isto vreme su uzajamno prosti sa datim brojem)

PRENOS PARAMETARA PO VREDNOSTI

Parametri se funkciji prenose po vrednosti, što znači da funkcija zapravo radi sa njihovim lokalnim kopijama.

- Sve promene koje funkcija načini nad parametrima, zapravo se odnose na lokalne kopije parametara, a ne na same parametre.
- Zaključak je da se preneti parametri NE MOGU promeniti u telu funkcije.

```
#include <stdio.h>
void f (int x) {
x *= 2;
x++;
}

int main() {
int x = 3;
f(x);
printf("%d\n", x);
return 0;
}
```

Analiza prethodnog primera

- U prethodnom primeru, biće ispisano 3,jer se promene na x-u, vrše nad njegovom lokalnom kopijom u funkciji f, a ne baš nad x-om.
- •Ako hoćemo da u funkciji promenimo x, jedan od načina je da iz funkcije vratimo to promenjeno x (na kraju funkcije da stavimo **return x;**),

  a u main funkciji da kažemo **x = f(x);**

Analiza prenosa parametara u primeru sa funkcijom zbir

Kada se u main funkciji pojavi poziv funkcije rezultat = zbir (5,3) realizuju se sledeće akcije:

– Rezerviše se memorijski prostor za promenljive deklarisane u funkciji zbir();

– Formalnim parametrima se dodeljuju vrednosti stvarnih parametara: a=3; b=5;

– Izvršava se telo funkcije;

– Rezultat izračunavanja u funkciji se postavlja na mesto obraćanja toj funkciji, odnosno dodeljuje se promenljivoj rezultat, i prelazi se na izvršenje sledeće naredbe u funkciji main().

SPOLJAŠNJE (GLOBALNE) PROMENLJIVE

C program sastoji se od skupa spoljašnjih objekata, koji mogu biti:

– promenljive

– funkcije

- Spoljašnje promenljive su definisane izvan svake funkcije i zato su potencijalno na raspolaganju raznim funkcijama.
- Funkcije su uvek spoljašnje, jer C ne dozvoljava definisanje funkcija unutar drugih funkcija

SPOLJAŠNJE (GLOBALNE) PROMENLJIVE

Prenošenje podataka između funkcija:

– preko argumenata

– preko povratnih vrednosti

– preko spoljašnjih promenljivih

- Ukoliko se veliki broj promenljivih mora deliti između funkcija, bolje je koristiti spoljašnje promenljive nego dugačke liste argumenata.
- Međutim, ovo treba pažljivo primenjivati, jer u suprotnom se dobija program sa previše veza između funkcija.

Spoljašnje promenljive - primer

```
#include <stdio.h>
/* Deklaracija funkcije prebroj */
void prebroj();

/* Globalni brojaci. Podrazumevano su inicijalizovani nulom. */
int br_malih, br_velikih, br_cifara, br_belina, br_redova;

int main() {
/* Pozivamo funkciju za prebrojavanje */
prebroj();
/* Prikazujemo vrednosti globalnih brojaca, koje je funkcija
prebroj() izmenila */
printf("Broj malih slova: %d\n", br_malih);
printf("Broj velikih slova: %d\n", br_velikih);
printf("Broj cifara: %d\n", br_cifara);
printf("Broj belina: %d\n", br_belina);
printf("Broj redova: %d\n", br_redova);
/* Povratna vrednost funkcije main() */
return 0;
}
```

Nastavak primera

```
/* Definicija funkcije prebroj() */
void prebroj() {
int c;
while((c = getchar()) != EOF) {
    if(c >= 'a' && c <= 'z')  ++br_malih;
    else if(c >= 'A' && c <= 'Z')   ++br_velikih;
    else if(c >= '0' && c <= '9')  br_cifara++;
   else if (c == '\n' || c == '\t' || c ==' ') {
            br_belina++;
            if(c == '\n') br_redova++;
      }

  }
}
```

Poređenje spoljašnjih i automatskih promenljivih

Automatske promenljive su unutrašnje za funkciju. One nastaju kada otpočinje izvršavanje funkcije i nestaju kada se ono završava.

- Spoljašnje promenljive su permanentne I stoga zadržavaju svoje vrednosti između poziva funkcija. Dakle, životni ciklus ovih promenljivih je duži.

STATIČKE PROMENLJIVE

Deklaracija static, koja se primenjuje na spoljašnju promenljivu ili funkciju, ograničava domet tog objekta na preostali deo izvorne datoteke.

- Deklaracija static se može primeniti i na unutrašnje promenljive. Unutrašnje statičke promenljive su lokalne za odreñenu funkciju baš kao i automatske promenljive, ali za razliku od njih, one nastavljaju da postoje i nakon završetka funkcije. Dakle, one ne nastaju i ne nestaju sa svakim pozivom funkcije.

INICIJALIZACIJA

Spoljašnje i statičke promenljive se uvek inicijalizuju na nulu ukoliko nisu eksplicitno inicijalizovane. Inicijalizator mora biti konstantan izraz. Inicijalizacija se obavlja samo jednom, pre početka izvršavanja programa.

- Automatske promenljive imaju nedefinisane početne vrednosti ukoliko nisu eksplicitno inicijalizovane. Inicijalizator može biti svaki izraz u kojem učestvuju prethodno definisane vrednosti, pa čak i pozivi funkcija. Inicijalizacija se obavlja svaki put prilikom izvršavanja funkcije ili bloka.

Životni vek i oblast važenja promenljivih - primer

```
#include <stdio.h>

/* Globalna promenljiva */
int a = 0;
/* Uvecava se globalna promenljiva */
void uvecaj() {
a++;  printf("uvecaj::a = %d\n", a);}

/* Umanjuje se lokalna promenljiva a. Globalna promenljiva sa istim
imenom zadrzava svoju vrednost. */
void umanji() {
/* Ova promenljiva a je nezavisna u odnosu na globalnu
promenljivu a */
int a = 0;
a--;
printf("umanji::a = %d\n", a);
}
```

Nastavak primera

```
void nestaticka_prom() {
/* Nestaticke promenljive ne cuvaju vrednosti kroz pozive funkcije */
int s = 0;
s++;
printf("nestaticka promenljiva::s = %d\n", s);
}

void staticka_prom() {
/* Staticke promenljive cuvaju vrednosti kroz pozive funkcije.
Inicijalizacija se odvija samo u okviru prvog poziva. */


static int s = 0;
s++;
printf("staticka promenljiva::s = %d\n", s);
}
```

Nastavak primera

```
int main() {
int i, x = 3; /* Ovo su promenljive lokalne za funkciju main */
printf("main::x = %d\n", x);
for(i=0; i<3; i++) {
/* Promenljiva u okviru bloka je nezavisna od spoljne promenljive. Ovde
se koristi promenljiva x lokalna za blok for 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);
uvecaj();     umanji();

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

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

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