Programiranje 1

Beleške sa vežbi

Školska 2008/2009 godina

Matematički fakultet, Beograd

Jelena Graovac (Tomašević)

December 18, 2008

Sadržaj

1

Programski jezik C

[1]

1.1 Statičke promenljive

Statičke promenljive se definišu pisanjem reči `static` ispred uobičajene definicije. Za statičke promenljive inicijalizacija se obavlja jedanput, pre početka izvršavanja programa dok se za ostale ona obavlja svaki put prilikom izvršavanja bloka ili funkcije.

Primer 1 *Demonstracija životnog veka i oblasti važenja promenjivih (scope).*

```
#include <stdio.h>

/* Globalna promenjiva */
int a = 0;

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

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

}

void nonstatic_var()
{
    /* Nestaticke promenjive ne cuvaju vrednosti kroz pozive funkcije */
```

[1] Zasnovano na primerima sa sajta `http://www.matf.bg.ac.yu/~filip` i knjige Milana Čabarkape "C - osnovi programiranja".

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

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

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

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


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

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

    increase();
    decrease();

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

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

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

Izlaz iz programa:
main::x = 3
```

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

1.2 Lenjo izračunavanje

Lenjo izračunavanje se odnosi na izračunavanje logičkih izraza. Tako se prilikom izračunavanja izraza `A && B`, ukoliko je `A` netačno, izraz `B` ne izračunava jer bez obzira na njegovu vrednost, vrednost celog izraza će svakako biti netačno. Slično tome, prilikom izračunavanja izraza `A || B` , ukoliko je `A` tačno, izraz `B` se ne izračunava jer bez obzira na njegovu vrednost, vrednost celog izraza će biti tačno.

Primer 2 *Ilustracija lenjog izračunavanja logičkih operatora.*

```
#include <stdio.h>

int b = 0;

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

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

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

    /* Funkcija izracunaj() ce se pozivati samo za neparne vrednosti a */
    b = 0;
```

```
    for (a = 0; a < 10; a++)
        if (a%2 == 0 || izracunaj())
            printf("Uslov ispunjen : a = %d, b = %d\n", a, b);
        else
            printf("Uslov nije ispunjen : a = %d, b = %d\n", a, b);
}

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

1.3 Pokazivači

Pokazivač (pokazivačka promenljiva) je promenljiva koja "pokazuje" na drugu promenljivu odnosno sadrži adresu memorijske lokacije u kojoj se čuva ta promenljiva.

Pokazivači se deklarišu na sledeći način: `tip *p1, *p2, ..., *pn;`

Specifikacija tipa zadaje tip promenljivih na koje pokazuju pokazivačke promenljive `p1, p2,...,pn`. Simbol * ispred imena promeljive ukazuje da je ona pokazivačka promenljiva. Na primer, pokazivači se mogu deklarisati na sledeći način:

```
int *ip; /* pokazivac na celobrojnu promenljivu */
char *cp; /* pokazivac na znakovnu promenljivu */
float *fp1, *fp2; /* pokazivac na float promenljivu */
```

Operatorima dodeljivanja:

```
i=5;
ip=&i;
```

promenljivoj `i` je dodeljena vrednost 5, a pokazivačkoj promenljivoj `ip` adresa od `i` (tako da pokazuje na promenljivu `i`).

Ako treba da pristupimo vrednosti promenljive na koju pokazuje pokazivačka promenljiva tada ispred imena pokazivačke promenljive stavljamo znak *. Na taj način vrednost na koju pokazuje `ip` dobijamo sa `*ip`.

```
int x=1, y=1, z[10];
int *ip;    /* ip je pokazivac na int,
               odnosno *ip je tipa int */

ip = &x;    /* ip sada pokazuje na x. &x znaci adresa od x. */
y=*ip;      /* y je sada 1 */
*ip = 0;    /* x je sada 0 */

*ip+=10;    /* x je sada 10*/
++*ip;      /* x je sada 11*/
(*ip)++;    /* x je sada 12,
            zagrada neophodna zbog prioriteta
            operatora*/

ip = &z[0]; /* ip sada pokazuje na z[0]*/
```

Primer 3 *Ilustracija rada sa pokazivačkim promenljivim.*

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

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

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


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

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

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

```
}

Izlaz (u konkretnom slucaju):
Adresa promenljive x je : 0012FF88
Vrednost promenljive x je : 3
Vrednost promenljive px je (tj. px) : 0012FF88
Vrednost promenljive na koju ukazuje px (tj. *px) je : 3
Vrednost promenljive na koju ukazuje px (tj. *px) je : 6
Vrednost promenljive x je : 6
```

Pored pokazivača na osnovne tipove, postoji i pokazivač na prazan tip (void).

```
void *pp;
```

Njemu može da se dodeli da pokazuje na int, ili na char ili na proizvoljan tip ali je to neophodno eksplicitno naglasiti svaki put kada želimo da koristimo ono na šta on pokazuje.

Primer 4 *Upotreba pokazivača na prazan tip.*

```
#include<stdio.h>

main()
{
void *pp;
int x=2;
char c='a';

pp = &x;
*(int *)pp = 17;    /* x postaje 17*/
printf("\n adresa od x je %p", &x);
printf("\n%d i %p",*(int*)pp,(int * )pp);

pp = &c;
printf("\n adresa od c je %p", &c);
printf("\n%c i %p",*(char*)pp,(char * )pp);

}

/*
 adresa od x je 0012FF78
17 i 0012FF78
 adresa od c je 0012FF74
a i 0012FF74

*/
```

Posebna konstanta koja se koristi da se označi da pokazivač ne pokazuje na neko mesto u memoriji je NULL.

1.4 Prenos parametara po vrednosti i preko pokazivača

Primer 5 *Demonstracija prenosa parametara po vrednosti - preneti parametri se ne mogu menjati!*

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

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

Ovaj problem se može prevazići na više načina.

Prvi način je da se promenljiva x ne prenese u funkciju f po vrednosti već preko pokazivača, odnosno da se prenese njena adresa.

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

main()
{
int x=3;
f(&x);
printf("%d\n", x);
}
Izlaz:
4
```

Drugi način je da se promenjena vrednost privremene promenljive x u okviru funkcije f vrati kao rezultat rada funkcije:

```
#include <stdio.h>
int f(int x)
{
return x++;
}

main()
{
int x=3;
printf("%d\n", f(x));
}
Izlaz:
4
```

Treći način je preko globalne promenljive:

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

```
int x=3;

void f()
{
x++;
}

main()
{
f();
printf("%d\n", x);
}

Izlaz:
4
```

Dakle, u programskom jeziku C argumenti se funkciji prosleđuju po vrednosti. To znači da sledeća funkcija neće uraditi ono što želimo:

```
void razmeni(int x, int y) /* POGRESNO!!!!!!!!*/
{
int pom;
pom = x;
x = y;
y = pom;
}
```

Zbog prenosa parametara po vrednosti, funkcija `razmeni` ne može da utiče na argumente `a` i `b` u funkciji koja je pozvala ovu funkciju `razmeni`. Funkcija `razmeni` samo zamenjuje kopije od `a` i `b`.

Da bi se dobio željeni efekat, potrebno je da se proslede adrese promenljivih `a` i `b`:

```
/* Zameni *px i *py */
void razmeni(int *px, int *py)
{
int pom;
pom = *px;
*px = *py;
*py = pom;
}
```

a poziv funkcije `razmeni` izlgeda sada ovako

```
razmeni(&a, &b);
```

Primer 6 *Demonstracija više povratnih vrednosti funkcije koristeći prenos preko pokazivača.*

```
    /* Funkcija istovremeno vraca dve vrednosti - kolicnik i ostatak dva data broja.
   Ovo se postize tako sto se funkciji predaju vrednosti dva broja (x i y) koji se dele
   i adrese dve promenljive na koje ce se smestiti rezultati */

void div_and_mod(int x, int y, int* div, int* mod)
{
    printf("Kolicnik postavljam na adresu : %p\n", div);
```

```
    printf("Ostatak postavljam na adresu : %p\n", mod);
    *div = x / y;
    *mod = x % y;
}

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

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

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

}

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

1.5 Zadaci za vežbu

Zadatak 1 *Broj je Armstrongov ako je jednak sumi n-tih stepena svojih cifara. Ispitati da li je broj koji se unosi sa standardnog ulaza Armstrongov.*

Zadatak 2 *Za dati broj može se formirati niz tako da je svaki sledeći član niza dobijen kao suma cifara prethodnog člana niza. Broj je srećan ako se dati niz završava sa jedinicom. Napisati program koji za uneti broj određuje da li je srećan.*

Zadatak 3 *Sa ulaza se unosi broj u osnovi deset i osnova* $<= 10$. *Odštampati vrednost datog broja u datoj osnovi.*

Zadatak 4 *Sa ulaza se unosi osnova* $<= 10$ *i broj. Proveriti da li je taj broj ispravan broj za datu osnovu i ako jeste izračunati njegovu vrednost u osnovi 10.*

Zadatak 5 *Broj je* **Nivenov** *ako je deljiv sumom svojih cifara.*

1. *Napsati funkciju koja računa sumu cifara broja* ***a****. Na primer, za broj* `121` *funkcija treba da vrati* `4`.
2. *Napisati funkciju koja proverava da li je broj* Nivenov *i vraća* `1` *ako jeste a* `0` *ako nije.*
3. *Napisati program koji za uneto* `n` *ispisuje prvih* `n` Nivenovih *brojeva.*
4. *Napisati program koji za uneto* `n` *ispisuje sve* Nivenove *brojeve manje od* `n`.

Zadatak 6 *Napisati program koji izračunava vrednost polinoma u tački x:*

1. *Napisati funkciju koja računa k-ti stepen prirodnog broja n.*
2. *Napisati program koji za uneti niz koeficijenata $a[i]$ i uneti broj x računa vrednost polinoma $a_n * x^n + a_{n-1} * x^{n-1} + ... + a_1 * x + a_0$*

Zadaci za praktikum

Zadatak 7 *Napisati funkciju koja izračunava broj grafičkih karaktera u tekstu i program koji vrši njeno testiranje.*

Zadatak 8 *Napisati funkciju koja izračunava zbir svih cifara koje su se pojavile u tekstu i program koji vrši njeno testiranje.*

Zadatak 9 *Napisati funkciju koja izračunava broj reči koje su se pojavile u tekstu i program koji vrši njeno testiranje. Smatrati da se reči sastoje iz proizvoljnog niza karaktera i da su razdvojene prazninama, tabulatorima ili prelascima u novi red.*

Zadatak 10 *Napisati funkciju koja izračunava broj reči u tekstu dužine n i program koji vrši njeno testiranje. Broj n se učitava sa ulaza.*

Zadatak 11 *Napisati funkciju koja izračunava zbir i razliku dva cela broja i program koji testira rad ove funkcije. Zbir vratiti preko liste argumenata.*