Osnovi programiranja

Beleške sa vežbi

Smer *Računarstvo i informatika*
Matematički fakultet, Beograd

Jelena Tomašević

December 25, 2005

Sadržaj

1

[1]

1.1 Ugnježdena petlja

Primer 1 *Ilustracija dve ugnježdene petlje.*

```
#include<stdio.h>
int main()
{
int i,j;

for(i=1; i<=3; i++)
    {
    for(j=1; j<=3; j++)
        printf("%d * %d = %d\t", i, j, i*j);
    printf("\n");
    }
}

Izlaz:
1 * 1 = 1       1 * 2 = 2       1 * 3 = 3
2 * 1 = 2       2 * 2 = 4       2 * 3 = 6
3 * 1 = 3       3 * 2 = 6       3 * 3 = 9
```

Primer 2 *Program koji ispisuje tablicu množenja*

```
#include<stdio.h>

main()
{
 int n, m;  /* Dimenzije tablice */
 int i, j;  /* Brojaci */

 scanf("%d", &n);
 scanf("%d", &m);

 /* Petlja po redovima... */
 for(i = 0; i < n; i++) {
```

[1] Zasnovano na primerima sa sajtova http://www.matf.bg.ac.yu/∼filip, http://www.matf.bg.ac.yu/∼milena, http://www.matf.bg.ac.yu/∼jelenagr.

```
    /* unutrasnja petlja */
    for(j = 0; j < m; j++)
        printf("%d * %d = %d\t", i, j, i*j);
    /* na kraju prelazimo u sledeci red */
    printf("\n");
 }
}
```

Primer 3 *Program koji ispisuje prvih n prostih brojeva*

```
#include<stdio.h>

main()
{
 int i, n, br, delilac, ostatak;

 printf("Unesite koliko prostih brojeva zelite da dobijete: \n");
 scanf("%d", &n);

 /* Inicijalizujemo brojac i - koliko smo prostih brojeva nasli do sad */
 i = 0;

 /* Pocetni broj za koji proveravamo da li je prost */
 br = 2;

 /* Trazimo i-ti prost broj */
 while(i < n) {
    /* Ako je u pitanju 2 ili 3 prost je */
    if (br <= 3)
        p = 1;
    else if (br % 2 == 0)
        /* ako je broj paran i veci od 2 onda nije prost */
        p = 0;
    else {
        /* Ispitujemo samo neparne pa delioci mogu biti samo neparni
        brojevi */
        delilac = 3;
        ostatak = 1;

        while(ostatak != 0 && delilac * delilac <= n) {
            ostatak = n % delilac;
            delilac++;
        }
        p = (ostatak != 0);
    }

    /* Ako je broj prost... */
    if (p) {
        /* stampamo ga... */
        printf("Broj %d je prost.\n", n);
        /* i uvecavamo broj pronadjenih prostih brojeva. */
        i++;
     }
```

```
 /* U svakom slucaju prelazimo na proveru da li je sledeci broj prost */
 br++;
 }
}
```

1.2 Bit-operatori

!!!Ne mešati sa logičkim operatorima!!!

```
& bitsko AND
| bitsko OR
^ bitsko ekskluzivno OR
<< levo pomeranje
>> desno pomeranje
~ jedinicni komplement
```

Primer 4 *Demonstracija bitskih operatora*

```
#include <stdio.h>

main()
{  printf("%o %o\n",255,15);
   printf( "255 & 15 = %d\n", 255 & 15 );
   printf( "255 | 15 = %d\n", 255 | 15 );
   printf( "255 ^ 15 = %d\n", 255 ^ 15 );
   printf( "2 << 2   = %d\n", 2 << 2 );
   printf( "16 >> 2  = %d\n", 16 >> 2 );
}

Izlaz iz programa je:
377 17
255 & 15 = 15
255 | 15 = 255
255 ^ 15 = 240
2 << 2   = 8
16 >> 2  = 4
```

Primer 5 *print_bits - stampa bitove u zapisu datog celog broja x.*

```
#include <stdio.h>

/* Funkcija stampa bitove datog celog broja x.
   Vrednost bita na poziciji i je 0 ako i samo ako se pri konjunkciji broja x sa maskom
   000..010....000 - sve 0 osim 1 na poziciji i, dobija 0.
   Funkcija krece od pozicije najvece tezine kreirajuci masku pomeranjem jedinice u levo
   za duzina(x) - 1 mesto, i zatim pomerajuci ovu masku za jedno mesto u levo u svakoj
   sledecoj iteraciji sve dok maska ne postane 0.
*/

void print_bits(int x)
{
```

```
    /* Broj bitova tipa unsigned */
    int wl = sizeof(int)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');

    putchar('\n');
}


main()
{
    print_bits(127);
    print_bits(128);
    print_bits(0x00FF00FF);
    print_bits(0xFFFFFFFF);
}
```

```
Izlaz iz programa:
00000000000000000000000001111111
00000000000000000000000010000000
00000000111111110000000011111111
11111111111111111111111111111111
```

Primer 6 *Program proverava da li se na k-tom mestu nalazi 1.*

```
#include <stdio.h>
main(){
   int n,k;
   printf("Unesite broj i poziciju tog broja
    koju zelite da proverite:\n");
   scanf("%d %d",&n,&k);
   if ((n&(1 << (k-1)))!=0)
     printf("Bit je 1\n");
   else
     printf("Bit je 0\n");
   return 0;
}
```

Primer 7 *Program postavlja na k-to mesto 1*

```
#include <stdio.h>

void print_bits(int x);

main(){
  int n,k;
  printf("Unesite broj i poziciju tog broja koju zelite da proverite:\n");
  scanf("%d %d",&n,&k);
  printf("Binarno, une\v seni broj je\n");
  print_bits(n);
```

```
  printf("Novi broj je %d\n",(n |(1<<k)));
  printf("Binarno, novi broj je\n");
  print_bits((n |(1<<k)));
  return 0;
}
```

Izrazom a>>b vrši se pomeranje sadržaja operanda a predstavljenog u binarnom obliku za b mesta u desno. Popunjavanje upražnjenih mesta na levoj strani zavisi od tipa podataka i vrste računara. Ako se pomeranje primenjuje nad operandom tipa unsigned popunjavanje je nulama. Ako se radi o označenom operandu popunjavanje je jedinicama kada je u krajnjem levom bitu jedinica, a nulama kada je u krajnjem levom bitu nula.

Primer 8 *Funkcija koja broji bitove postavljene na 1 u broju*

```
int bitcount(unsigned x)
{
    int b;
    for(b=0; x!=0; x>>=1)
    if (x & 01) b++;
    return b;
}
```

Primer 9 *sum_of_bits - izračunava sumu bitova datog neoznačenog broja.*

```
#include <stdio.h>

/* Pomocna funkcija - stampa bitove neoznacenog broja */
void print_bits(unsigned x)
{
    int wl = sizeof(unsigned)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');

    putchar('\n');
}

/*
int sum_of_bits(unsigned x)
{

    int wl = sizeof(unsigned)*8;
    int br = 0;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask>>=1)
        if (x&mask)
            br++;

    return br;
}
```

```
*/

/* Efikasnija verzija */
int sum_of_bits(unsigned x)
{
    int br;
    for (br = 0; x; x>>=1)
        if (x&1)
            br++;

    return br;
}


main()
{
    printf("Binarni zapis broja 127 je\n");
    print_bits(127);
    printf("Suma bitova broja 127 je %d\n",sum_of_bits(127));
    printf("Binarni zapis broja 128 je\n");
    print_bits(128);
    printf("Suma bitova broja 128 je %d\n",sum_of_bits(128));
    printf("Binarni zapis broja 0x00FF00FF je\n");
    print_bits(0x00FF00FF);
    printf("Suma bitova broja 0x00FF00FF je %d\n",sum_of_bits(0x00FF00FF));
    printf("Binarni zapis broja 0xFFFFFFFF je\n");
    print_bits(0xFFFFFFFF);
    printf("Suma bitova broja 0xFFFFFFFF je %d\n",sum_of_bits(0xFFFFFFFF));
}
```

Primer 10 *get_bits, set_bits, invert_bits - izdvajanje, postavljanje i invertovanje pojedinacnih bitova*

```
#include <stdio.h>

/* Pomocna funkcija - stampa bitove neoznacenog broja */
void print_bits(unsigned x)
{
    int wl = sizeof(unsigned)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');

    putchar('\n');
}


/* Funkcija vraca n bitova broja x koji pocinju na poziciji p */
unsigned get_bits(unsigned x, int p, int n)
{
    /* Gradimo masku koja ima poslednjih n jedinica
        0000000...00011111
       tako sto sve jedinice ~0 pomerimo u levo za n mesta
```

```
            1111111...1100000
       a zatim komplementiramo
    */
    unsigned last_n_1 = ~(~0 << n);

    /* x pomerimo u desno za odgovarajuci broj mesta, a zatim
       konjunkcijom sa konstruisanom maskom obrisemo pocetne cifre */

    return (x >> p+1-n) & last_n_1;
}

/* Funkcija vraca modifikovano x tako sto mu je izmenjeno n bitova
   pocevsi od pozicije p i na ta mesta je upisano poslednjih n bitova
   broja y */
unsigned set_bits(unsigned x, int p, int n, unsigned y)
{
    /* Maska 000000...000111111  - poslednjih n jedinica */
    unsigned last_n_1 = ~(~0 << n);

    /* Maska 1111100..000111111 - n nula pocevsi od pozicije p */
    unsigned middle_n_0 = ~(last_n_1 << p+1-n);

    /* Brisemo n bitova pocevsi od pozicije p */
    x = x & middle_n_0;

    /* Izdvajamo poslednjih n bitova broja y i pomeramo ih na poziciju p */
    y = (y & last_n_1) << p+1-n;

    /* Upisujemo bitove broja y u broj x i vracamo rezultat */
    return x | y;

}

/* Invertuje n bitova broja x pocevsi od pozicije p */
unsigned invert_bits(unsigned x, int p, int n)
{
    /* Maska 000000111...1100000 - n jedinica pocevsi od pozicije p */
    unsigned middle_n_1 = ~(~0 << n) << p+1-n;

    /* Invertujemo koristeci ekskluzivnu disjunkciju */
    return x ^ middle_n_1;
}

main()
{
    unsigned x = 0x0AA0AFA0;
    print_bits(x);

    print_bits(get_bits(x, 15, 8));
    print_bits(set_bits(x, 15, 8, 0xFF));
    print_bits(invert_bits(x, 15, 8));
}
```

```
Izlaz iz programa:
00001010101000001010111110100000
00000000000000000000000010101111
00001010101000001111111110100000
00001010101000000010100010100000
```

Primer 11 *right_rotate_bits, mirror_bits - rotiranje i simetrija bitova.*

```
#include <stdio.h>

/* Pomocna funkcija - stampa bitove neoznacenog broja */
void print_bits(unsigned x)
{
    int wl = sizeof(unsigned)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');

    putchar('\n');
}


/* Funkcija vrsi rotaciju neoznacenog broja x za n pozicija u desno */
unsigned right_rotate(unsigned x, int n)
{
    int i;
    int wl = sizeof(unsigned)*8;

    /* Postupak se ponavlja n puta */
    for (i = 0; i < n; i++)
    {
        /* Poslednji bit broja x */
        unsigned last_bit = x & 1;

        /* x pomeramo za jedno mesto u desno */
        x >>= 1;

        /* Zapamceni poslednji bit stavljamo na pocetak broja x*/

        x |= last_bit<<wl-1;
    }

    return x;
}

/* Funkcija obrce binarni zapis neoznacenog broja x tako sto bitove cita unatrag */
unsigned mirror(unsigned x)
{
    int i;
    int wl = sizeof(unsigned)*8;
```

```
    /* Rezultat inicijalizujemo na poslednji bit broja x */
    unsigned y = x & 1;

    /* Postupak se ponavlja wl-1 puta */
    for (i = 1; i<wl; i++)
    {
        /* x se pomera u desno za jedno mesto */
        x >>= 1;
        /* rezultat se pomera u levo za jedno mesto */
        y <<= 1;

        /* Poslednji bit broja x upisujemo na poslednje mesto rezultata */
        y |= x & 1;
    }
    return y;
}

main()
{
    unsigned x = 0xFAF0FAF0;
    print_bits(x);
    print_bits(mirror(x));
    print_bits(right_rotate(x, 2));
}

Izlaz iz programa:
11111010111100001111101011110000
00001111010111110000111101011111
00111110101111000011111010111100
```

Zadaci za vežbu:

Zadatak 1 *Napisati program koji ispituje da li dva niza imaju barem jedan zajednički element.*

Zadatak 2 *Napisati operator dodeljivanja koji će broju x tipa unsigned sačuvati n krajnjih desnih bitova, a ostale postaviti na nulu.*

```
x=x&~(~0 << n); ili x&=~(~0 << n);
```

Zadatak 3 *Napisati operator dodeljivanja koji će u x očistiti n bitova (postaviti nule) počev od pozicije p.*

```
x&=~(~(~0<<n)<<(p-1))
```

Zadatak 4 *Napisati operator dodeljivanja kojim se invertuje x (prevodi jedan u nulu i nula u jedan) počev od pozicije p na dužini n.*

```
x^=(~(~0<<n)<<(p-1));
```

Zadatak 5 *Program koji sabira pozitivne brojeve niza cifara koji se završava nulom.*

```
#include<stdio.h>

main()
```

```
{
 int x, zbir;

 printf("Unesite niz cifara pri cemu je 0 oznaka za kraj\n");

 /* Beskonacna while petlja. */
 while( 1 ) {
    /* Citamo sledeci element... */
    scanf("%d", &x );
    /* ako smo procitali 0 znaci da smo stigli do kraja... */
    if( x == 0 )
        /* i izlazimo iz petlje */
        break;
    /* Ako je broj negativan preskacemo ga... */
    else if( x < 0 )
        /* i idemo na sledeci */
        continue;
    /* inace, broj je pozitivan i dodajemo ga u zbir. */
    else zbir = zbir + x;
 }

 printf("Suma pozitivnih je %d\n", zbir);
}
```

Primer 12 *Program koji računa zbir* $1 + x + \frac{x^2}{2} + \ldots + \frac{x^n}{n!}$

```
#include<stdio.h>

main()
{
 float f, suma; /* Faktor sume i suma */
 float x;       /* Promenljiva x iz izraza */
 int i;         /* Brojac u petljama */
 int n;         /* Broj sabiraka */

 scanf("%d", n);
 scanf("%d", x);

 /* Pocetne inicijalizacije */
 f = 1;
 suma = 1;

 /* U jednom prolazu petlje dodajemo tekuci sabirak */
 for(i = 1; i <=n; i++) {
    f = f * x / i;
    suma = suma + f;
 }

 printf("Suma prvih %d clanova je %f \n", n, suma);
}
```

Primer 13 *Napisati program koji računa sumu* $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^n * \frac{x^{2n-1}}{(2n-1)!}$

```
#include<stdio.h>

main()
{
 float f, suma, x;
 int i, n;

 scanf("%d", &n);
 scanf("%f", &x);

 /* Pocetne inicijalizacije */
 suma = x;
 f = x;

 for(i = 1; i <= n; i++) {
    f = -f * x * x / ((2*i+1)*2*i);
    suma = suma * f;
 }

 printf("Suma prvih %d clanova je %f \n", n, suma);
}
```

Primer 14 (DOMAĆI) *Napisati program koji računa sumu* $1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + (-1)^n \frac{x^{2n}}{(2n)!}$

Primer 15 *Program koji računa sumu* $x - \frac{x^3}{3*1!} + \frac{x^5}{5*2!} - \frac{x^7}{7*3!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)*n!}$

```
#include<stdio.h>

main()
{
 int i, n;
 float x, f, suma;

 scanf("%d", &n);
 scanf("%f", &x);

 /* Pocetne inicijalizacije */
 f = x;
 suma = x;

 for(i = 1; i < n; i++) {
    f = -f * x * x / i;
    suma = suma + f/(2*i+1);
 }

 printf("Suma prvih %d clanoca je %f\n", n, suma);
}
```