Programiranje I

Beleške sa vežbi

Smer *Informatika*

Matematički fakultet, Beograd

Sana Stojanović

December 12, 2007

Sadržaj

1 Operator *sizeof*

1. ```
   /* Demonstracija sizeof operatora. sizeof operator izracunava velicinu
      tipa odnosno promenjive. */

   #include<stdio.h>

   main()
   {
       int i;
       float f;
       int n[10];


       printf("sizeof(int)=%d\n", sizeof(int));
       printf("sizeof(long)=%d\n", sizeof(long));
       printf("sizeof(short)=%d\n", sizeof(short));
       printf("sizeof(signed)=%d\n", sizeof(signed));
       printf("sizeof(unsigned)=%d\n", sizeof(unsigned));
       printf("sizeof(char)=%d\n", sizeof(char));
       printf("sizeof(float)=%d\n", sizeof(float));
       printf("sizeof(double)=%d\n", sizeof(double));

       printf("sizeof(i)=%d\n", sizeof(i));
       printf("sizeof(f)=%d\n", sizeof(f));
       printf("sizeof(n)=%d\n", sizeof(n));
       printf("Broj elemenata niza n : %d\n", sizeof(n)/sizeof(int));

   }


   Izlaz iz programa(u konkretnom slucaju): sizeof(int)=4
   sizeof(long)=4 sizeof(short)=2 sizeof(signed)=4 sizeof(unsigned)=4
   sizeof(char)=1 sizeof(float)=4 sizeof(double)=8 sizeof(i)=4
   sizeof(f)=4 sizeof(n)=40 Broj elemenata niza n : 10
   ```

2 switch naredba

1. ```
   /* Ilustracija switch konstrukcije */

   #include<stdio.h>

   main()
   {
       int n;
   ```

```
    printf("Unesi paran broj manji od 10\n");
    scanf("%d",&n);
    switch(n) {
        case 0:
             printf("Uneli ste nulu\n");
             break;
             /* Naredba break; je obavezna! Bez nje bismo dobili:
                Uneli ste nulu
                Uneli ste dvojku... */
        case 2:
              printf("Uneli ste dvojku\n");
              break;
        case 4:
              printf("Uneli ste cetvorku\n");
              break;
        case 6:
              printf("Uneli ste sesticu\n");
              break;
        case 8:
              printf("Uneli ste osmicu\n");
              break;
        default:
             printf("Uneli ste nesto sto nije paran broj\n");
    }
}

Ulaz: Unesi paran broj manji od 10
2
Izlaz: Uneli ste dvojku
```

3 Bitovski operatori

<<	šiftovanje ulevo
>>	šiftovanje udesno
&	bitska konjunkcija
~	komplement
|	disjunkcija
^	ekskluzivna disjunkcija

1. ```
   /* 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 udesno 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
```

2.
```
/* 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;
}
```

3.
```
/* 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 promenite:\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)));

}
```

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

4.
```
/* sum_of_bits - izracunava sumu bitova datog neoznacenog broja */

#include <stdio.h>

int sum_of_bits(unsigned x)
{
        /* Broj bitova tipa unsigned */
        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("%d\n",sum_of_bits(127));
        printf("%d\n",sum_of_bits(128));
        printf("%d\n",sum_of_bits(0x00FF00FF));
        printf("%d\n",sum_of_bits(0xFFFFFFFF));
}
```

5.
```
/* get_bits, set_bits, invert_bits -
   izdvajanje, postavljanje i invertovanje grupe 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:
00001010 10100000 10101111 10100000
00000000 00000000 00000000 10101111
00001010 10100000 11111111 10100000
00001010 10100000 01010000 10100000
```

6. ```
   /* 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:
11111010 11110000 11111010 11110000
00001111 01011111 00001111 01011111
00111110 10111100 00111110 10111100
```

7. **Za vežbu:** Napisati program kojim se za date n i m izračunava suma:

$$S = \frac{n}{b(n)} - \frac{n+1}{b(n+1)} + \ldots + (-1)^{m-n}\frac{m}{b(m)}$$

gde je rezultat funkcije b(n) broj jedinica u binarnom zapisu dekadnog broja n.

8. Operator dodeljivanja koji će broju x tipa unsigned sačuvati n krajnjih desnih bitova, a ostale postaviti na nulu.

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

9. Operator dodeljivanja koji će u x očistiti n bitova (postaviti nule) počev od pozicije p.

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

10. Operator dodeljivanja kojim se invertuje x (prevodi jedan u nula i nula u jedan) počev od pozicije p na dužini n.

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

4 Funkcije *getchar* i *putchar*, redirekcija ulaza i izlaza

1. Ilustracija funkcija *getchar* i *putchar*.

```
/* Program cita jedan karakter i ispisuje ga - demonstracija
   putchar i getchar. */

#include <stdio.h>

main()
{
    int c;          /* Karakter - obratiti paznju na int */
    c = getchar(); /* cita karakter sa standardnog ulaza */
    putchar(c);    /* pise karakter c na standardni izlaz */

    putchar('\n'); /* prelazak u novi red */
    putchar('a');  /* ispisuje malo a */
    putchar(97);   /* ekvivalentno prethodnom */
}


Ulaz: s

Izlaz iz programa: s s aa
```

2.
```
/* Program prepisuje standardni ulaz na standardni izlaz.
   Ilustracija redirekcije standardnog ulaza i izlaza.
   Pokrenuti program sa :

        ./a.out <primer.c
        ./a.out >tekst.txt
        ./a.out <primer.c >kopija.c
*/


#include <stdio.h>

main()
{
    int c;
    /* Obratiti paznju na raspored zagrada */
    while ((c = getchar()) != EOF)
        putchar(c);
}
```

3.
```
/* to_upper, to_lower - funkcije konverzije u velika, odnosno mala
   slova */

#include <stdio.h>

/* Funkcija konvertuje mala slova u velika,
   a ostala ne menja */
char to_upper(char c)
{
        if ('a'<=c && c<='z')
                return c-'a'+'A';
        return c;
}

/* Funkcija konvertuje velika slova u mala,
   a ostala ne menja */
char to_lower(char c)
{
        if ('A'<=c && c<='Z')
                return c-'A'+'a';
        return c;
}

main()
{
```

```
        int c;
        while ((c = getchar()) != EOF)
                putchar(to_upper(c));
}
```

4. `/* Program vrsi prebrojavanje unetih karaktera sa ulaza. */`

   I varijanta preko while petlje:

```
        #include<stdio.h>

        main()
        {
            long br;

            /* Pocetna inicijalizacija */
            br = 0;
            while (getchar() != EOF)
                br++;
            printf("%ld\n", br);
        }
```

   II varijanta preko for petlje:

```
        #include<stdio.h>

        main()
        {
            long br;

            for(br = 0; getchar() != EOF; br++)
                ;
            printf("%ld\n", br);
        }
```

5. `/* Program broji linije sa ulaza (broj pojavljivanja karaktera za novi red). */`

```
#include <stdio.h>
main()
{
   int c; /* Znak sa ulaza */
   int brl=0; /* Brojac linija */

   while((c=getchar()) != EOF)
```

```
     if (c=='\n')
        brl++;
   printf("Prelazaka u novi red ima: %d\n",brl);
}
```

6.
```
/* Program koji broji linije, reci i karaktere sa ulaza. */

#include<stdio.h>

#define IN 1   /* oznacava da smo unutar reci */
#defin OUT 0   /* oznacava da smo van reci */

main()
{
 int c, br_linija, br_reci, br_karaktera, stanje;

 /* U promenljivoj stanje cuvamo podatak da li se trenutno nalazimo
 unutar reci ili ne. Na pocetku se ne nalazimo u reci. */
 stanje = OUT;

 /* Inicijalizacija brojaca. */
 br_linija = br_reci = br_karaktera = 0;

 while((c = getchar()) != EOF) {
    br_karaktera++;
    if (c == '\n')
        br_linija++;
    if (c == ' ' || c == '\n' || c == '\t')
        stanje = OUT;
    else if (stanje == OUT) {
        stanje = IN;
        br_reci++;
        }
    }

 printf("Broj reci: %d\nBroj linija: %d\nBroj karaktera: %d\n", br_reci,
 br_linija, br_karaktera);
}
```

7.
```
/* Program broji cifre, praznine i broj ostalih karaktera. */

#include<stdio.h>

main()
{
```

```
    int c, br_cif, br_praz, br_ost;

    br_cif = br_praz = br_ost = 0;

    while ((c = getchar()) != EOF) {
       switch(c) {
           case '0': case '1': case '2': case '3': case '4':
           case '5': case '6': case '7': case '8': case '9':
               br_cif++;
               break;
           case ' ':
           case '\t':
           case '\n':
               br_praz++;
               break;
           default:
               br_ost++;
       }

    printf("Broj cifara: %d\nBroj praznina: %d\nBroj ostalih karaktera: %d\n",
       br_cir, br_praz, br_ost);
}
```

8. ```
   /* Prepisuje ulaz na izlaz cineci tabulatore, nove linije i
   backslash-ove vidljivim. */

   #include <stdio.h>
   main()
   {
       int znak;

       znak=getchar();
       while( znak!=EOF )
       {
           if( znak=='\t' ) /*uciniti tab vidljivim */
               {
                putchar('\\');
                putchar('t');
               }
           else
               if( znak=='\n' ) /*uciniti new line vidljiv */
                   {
                    putchar('\\');
                    putchar('n');
                    putchar('\n');
   ```
   ```

```
                }
            else
                if( znak=='\\' ) /*backslash udvojiti */
                    {
                     putchar('\\');
                     putchar('\\');
                    }
                else
                    putchar(znak);

        znak=getchar();
    } /* while( znak!=EOF ) */
} /*main() */
```

9. ```
   /* Program koji prepisuje ulaz na izlaz pri cemu vise blanko
   znakova zamenjuje jednim. */

   #include <stdio.h> #define GRANICA '0'

   main() {
       int znak; /*tekuci znak sa ulaza*/
       int preth; /*znak koji prethodi tekucem */

       preth=GRANICA;
       while ( (znak=getchar() ) !=EOF)
       {
        if (znak !=' ' || preth != ' ')
           putchar(znak);
        preth=znak;
       }
   }
   ```

10. ```
    /* Program proverava da li su zagrade ( i ) dobro uparene. */

    #include <stdio.h>
    #include <stdlib.h>

    main()
    {
        int c;
        int br_otv = 0;

        while((c=getchar()) != EOF)
        {
         switch(c){
            case '(': br_otv++;
                      break;
    ```

```
        case ')': br_otv--;
                  if (br_otv<0)
                    {
                        printf("Visak zatvorenih zagrada\n");
                        exit(1);
                    }
        }
    }

    if (br_otv == 0)
        printf("Zagrade su u redu\n");
    else
        printf("Visak otvorenih zagrada\n");
}
```

5 Domaći zadatak

1. Napisati program koji za uneti ceo broj n sa standardnog ulaza određuje prvi prost broj koji je veći (ili jednak) od njega. Na primer,

   ```
   Za uneti broj 8 program treba da vrati 11
   Za uneti broj 3 program treba da vrati 3
   ```

   Treba poslati kod programa ***domaci2.c*** *i u tex-u opis programa* ***domaci2.tex*** *u koji ćete ubaciti kod programa naredbom verbatim.*