Programiranje 1

Beleške sa vežbi

Školska 2006/2007 godina

Matematički fakultet, Beograd

Jelena Tomašević

January 16, 2007

Sadržaj

1

Programski jezik C

[1]

1.1 Linearna i binarna pretraga niza

Primer 1 *Linearno pretraživanje*

```
#include <stdio.h>
/* Funkcija proverava da li se dati element x nalazi
u datom nizu celih brojeva.
Funkcija vraca poziciju u nizu na
kojoj je x pronadjen
odnosno -1 ukoliko elementa nema.
*/
int linearna_pretraga(int niz[], int br_elem, int x)
{
    int i;
    for (i = 0; i<br_elem; i++)
        if (niz[i] == x)
            return i;
    /* nikako else */
    return -1;
}

main()
{
    /* Inicijalizacija niza moguca je
    na ovaj nacin*/
    int a[] = {4, 3, 2, 6, 7, 9, 11};
    /* Da bi smo odredili koliko clanova
    ima niz mozemo koristiti operator
    sizeof*/
    int br_elem = sizeof(a)/sizeof(int);
    int x;
    int i;
```

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

```
    printf("Unesite broj koji trazimo : ");
    scanf("%d",&x);
    i = linearna_pretraga(a, br_elem, x);
    if (i == -1)
        printf("Element %d nije nadjen\n",x);
    else
        printf("Element %d je nadjen na poziciji %d\n",x, i);
}
```

Primer 2 *Binarna pretraga niza*

```
/* Binarna pretraga niza celih brojeva - iterativna verzija*/

#include <stdio.h>

/* Funkcija proverava da li se element x javlja unutar niza
celih brojeva a.
Funkcija vraca poziciju na kojoj je element nadjen odnosno
-1 ako ga nema.
!!!!! VAZNO !!!!!
Pretpostavka je da je niz a uredjen po velicini
*/
int binarna_pretraga(int a[], int n, int x)
{
    /* Pretrazujemo interval [l, d] */
    int l = 0;
    int d = n-1;
    /* Sve dok interval [l, d] nije prazan */
    while (l <= d)
    {
        /* Srednja pozicija intervala [l, d] */
        int s = (l+d)/2;
        /* Ispitujemo odnos x i a[srednjeg elementa] */
        if (x == a[s])
            /* Element je pronadjen */
            return s;
        else if (x < a[s])
                {
                 /* Pretrazujemo interval [l, s-1] */
                 d = s-1;
                }
            else
            {
                /* Pretrazujemo interval [s+1, d] */
                l = s+1;
            }
    }
    /* Element nije nadjen */
    return -1;
}

main()
```

```
{
    int a[] = {3, 5, 7, 9, 11, 13, 15};
    int x;
    int i;
    printf("Unesi element koji trazimo : ");
    scanf("%d",&x);
    i = binarna_pretraga(a, sizeof(a)/sizeof(int), x);
    if (i==-1)
        printf("Elementa %d nema\n", x);
    else
        printf("Pronadjen na poziciji %d\n", i);
}
```

1.2 Životni vek i oblast važenja promenjivih. Statičke promenljive

Primer 3 *Demonstracija zivotnog veka i oblasti vazenja 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 */
    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
```

Primer 4 *Primer ilustruje vidljivost imena promenljivih.*

```
#include <stdio.h> main() {
    int pom=1;
    printf("Pre ulaska u unutrasnji blok pom=%d\n",pom);
    {
        int pom=50;
        printf("Pre izlaska iz unutrasnjeg bloka  pom=%d\n",pom);
    }
    printf("Nakon izlaska iz  unutrasnjeg bloka  pom=%d\n",pom);
}

Izlaz: Pre ulaska u unutrasnji blok pom=1
Pre izlaska iz unutrasnjeg bloka  pom=50
Nakon izlaska iz  unutrasnjeg bloka pom=1
```

1.3 Konverzija

1.3.1 Automatska konverzija

Ako je jedan od operanada različličit vrši se konverzija, uvek u smeru manjeg ka većem tipu

Naredba dodele:

```
int i=5;
float f=2.3;
f=i; /* f ce imati vrednost 5.0*/

obrnuto:

int i=5;
float f=2.3;
i=f; /* i ce imati vrednost 2*/
```

1.3.2 Eksplicitna konverzija

```
(tip)<izraz>

float x;
x=2.3+4.2;          /* x ce imati vrednost 6.5 */
x=(int)2.3+(int)4.2; /* x ce imati vrednost 6 */
x=(int)2.3*4.5;     /* x ce imati vrednost 9.0 jer zbog prioriteta
                    operatora konverzije prvo ce biti izvrsena
                    konverzija broja 2.3 u 2 pa tek onda izvrseno
                    mnozenje. */
x=(int)(2.3*4.5)    /* x ce imati vrednost 10.0 */
```

Primer 5 *Kako izbeći celobrojno deljenje*

```
int a,b;
float c;
a = 5;
b = 2;
c = a/b; /* Celobrojno deljenje, c=2*/
c = (1.0*a)/b;  /* Implicitna konverzija: 1.0*a je realan
                broj pa priliko deljenja sa b dobija se
                realan rezultat c=2.5*/
c = (0.0+a)/b;  /* Implicitna konverzija: (0.0+a) je realan
                broj pa priliko deljenja sa b dobija se
                realan rezultat c=2.5*/
c = (float)a/(float)b; /* Eksplicitna konverzija*/
```

1.3.3 Funkcije koje vrše konverziju

Primer 6

```
#include <stdio.h>
main()
{
int vrednost;
vrednost='A';
printf("Veliko slovo\n karakter=%3c\nvrednost=%3d\n",vrednost,vrednost);
vrednost='a';
printf("Malo\n karakter=%3c\nvrednost=%3d\n",vrednost,vrednost);
}

Izlaz (u slucaju ASCII):
Veliko slovo
karakter= A
vrednost= 65
Malo
karakter= a
vrednost= 97
```

Primer 7 *Funkcija koja konvertuje velika slova u mala slova.*

```
#include<stdio.h>

/* Konvertuje karakter iz velikog u malo slovo */
char lower(char c)
{
if (c >= 'A' && c <= 'Z')
    return c - 'A' + 'a' ;
else
    return c;
}

main()
{
char c;
printf("Unesi neko veliko slovo:\n");
scanf("%c", &c);
printf("Odgovarajuce malo slovo je %c\n", lower(c));
```

```
}

Izlaz:
Unesi neko veliko slovo:
J
Odgovarajuce malo slovo je j
```

Primer 8 *Konvertovanje niske cifara u ceo broj.*

```
#include<stdio.h>

/* atoi:  konvertuje s u ceo broj */
int atoi(char s[])
{
int i, n;
n = 0;
for (i = 0; (s[i] >= '0') && (s[i] <= '9'); ++i)
    n = 10 * n + (s[i] - '0');
return n;
}

main()
{
int n;
n = atoi("234");
printf("\nN je : %d\n",n);
}

Izlaz:

N je : 234
```

Primer 9 *btoi - konverzija iz datog brojnog sistema u dekadni.*

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

/* Pomocna funkcija koja izracunava vrednost koju predstavlja karakter u datoj osnovi
   Funkcija vraca -1 ukoliko cifra nije validna.

   Npr.
   cifra 'B' u osnovi 16 ima vrednost 11
   cifra '8' nije validna u osnovi 6

*/

int digit_value(char c, int base)
{
    /* Proveravamo obicne cifre */
    if (isdigit(c) && c < '0'+base)
        return c-'0';
```

```
    /* Proveravamo slovne cifre za mala slova */
    if ('a'<=c && c < 'a'+base-10)
        return c-'a'+10;

    /* Proveravamo slovne cifre za velika slova */
    if ('A'<=c && c < 'A'+base-10)
        return c-'A'+10;

    return -1;
}

/* Funkcija izracunava vrednost celog broja koji je zapisan u datom
   nizu karaktera u datoj osnovi. Za izracunavanje se koristi Hornerova shema.
*/
int btoi(char s[], int base)
{
    int sum = 0;

    /* Obradjuju se karakteri sve dok su cifre */
    int i, vr;
    for (i = 0; (vr = digit_value(s[i], base)) != -1; i++)
        sum = base*sum + vr;

    return sum;

}

main()
{
    char bin[] = "11110000";
    char hex[] = "FF";

    printf("Dekadna vrednost binarnog broja %s je %d\n", bin, btoi(bin, 2));
    printf("Dekadna vrednost heksadekadnog broja %s je %d\n", hex, btoi(hex, 16));

}

Izlaz:
Dekadna vrednost binarnog broja 11110000 je 240
Dekadna vrednost heksadekadnog broja FF je 255
```

Primer 10 *Program vrsi konverziju iz dekadnog brojnog sistema u datu osnovu*

```
#include<stdio.h>
#define OSNOVA 16
main()
{
    int x;  /* Broj cija se konverzija vrsi */
    int ostaci[32]; /* Niz ostataka pri deljenju sa osnovom */
    int i = 0;

        /* Unosi se dekadni broj */
    scanf("%d",&x);
```

```
    /* Srz algoritma konverzije */
    while(x>0)
    {
        /* novi ostatak se dodaje u pomocni niz */
        ostaci[i++] = x%OSNOVA;
        x/=OSNOVA;
    }

    /* Niz se ispisuje unatrag */
        for (i--; i>=0; i--)
        if (ostaci[i]<=10)  /* Slucaj kada je cifra dekadna */
            printf("%d",ostaci[i]);
        else    /* Slucaj kada cifra prevazilazi dekadni opseg */
            printf("%c",'A'+ostaci[i]-10);

    printf("\n");
}
```

1.4 #define sa argumentima

Primer 11 *Demonstracija pretprocesorske direktive* `#define` *sa argumentima*

```
#include<stdio.h>
/* Racuna sumu dva broja */
#define sum(a,b) ((a)+(b))

/* Racuna kvadrat broja - pogresna verzija */
#define square_w(a) a*a

/* Racuna kvadrat broja */
#define square(a) ((a)*(a))

/* Racuna minimum tri broja */
#define min(a, b, c) (a)<(b) ? ((a)<(c) ? (a) : (c)) : ((b)<(c) ? (b) : (c))

main()
{
    printf("sum(3,5) = %d\n", sum(3,5));
    printf("square_w(5) = %d\n", square_w(5));
    printf("square_w(3+2) = %d\n", square_w(3+2));
    printf("square(3+2) = %d\n", square(3+2));
    printf("min(1,2,3) = %d\n", min(1,2,3));
    printf("min(1,3,2) = %d\n", min(1,3,2));
    printf("min(2,1,3) = %d\n", min(2,1,3));
    printf("min(2,3,1) = %d\n", min(2,3,1));
    printf("min(3,1,2) = %d\n", min(3,1,2));
    printf("min(3,2,1) = %d\n", min(3,2,1));
}
```

Izlaz iz programa:

```
sum(3,5) = 8
square_w(5) = 25
square_w(3+2) = 11
square(3+2) = 25
min(1,2,3) = 1
min(1,3,2) = 1
min(2,1,3) = 1
min(2,3,1) = 1
min(3,1,2) = 1
min(3,2,1) = 1
```

Primer 12 *Demonstracija pretprocesorske direktive* `#define` *sa argumentima*

```
#include <stdio.h>
#define KUBW(a) (a * a * a)
#define KUB(a)  ( (a) * (a) * (a))

main()

{

int b=1;

printf("KUB(%d) = %d\n", 2*b+4, KUBW(2*b+4) );
printf("KUB(%d) = %d\n", 2*b+4, KUB(2*b+4) );

}

Izlaz:
KUB(6) = 22
KUB(6) = 216
```

Primer 13 *Ilustacija beskonačne petlje:*

```
#define forever for(;;);
```

Moguće je definisati makroe sa argumentima tako da tekst zamene bude različit za različita pojavljivanja makroa.

Primer 14

```
#define max(A, B) ((A)>(B) ? (A) : (B))
```

na osnovu ovoga će linija

```
x=max(p+q, r+s)
```

biti zamenjena linijom

```
x=((p+q) > (r+s) ? (p+q) : (r+s));
```

Treba voditi računa o sporednim efektima. Sledeća linija koda pruzrokovaće uvećanje vrednosti i i j za dva.

```
max(i++, j++)
```

Takođe treba voditi računa o zagradama. Sledeći makro prouzrokovaće neočekivane rezultate za poziv square(a+1)

```
#define square(x) x*x
```

Primer 15

```
#include  <stdio.h>
#define max1(x,y) (x>y?x:y)
#define max2(x,y) ((x)>(y)?(x):(y))
#define swapint(x,y) { int z; z=x;  x=y; y=z; }
#define swap(t,x,y) { \
    t z; \
    z=x;  \
    x=y; \
    y=z; }

main()
{

int x=2,y=3;

printf( "max1(x,y) = %d\n", max1(x,y)  );
/* max1(x,y) = 3 */

/* Zamena makroom se ne vrsi
unutar niski pod navodnicima*/
printf( "max1(x=5,y) = %d\n", max1(x,y)  );
/* max1(x=5,y) = 3 */

printf( "max1(x++,y++) = %d\n", max1(x++,y++)  );
/* max1(x++,y++) = 4 */


printf( "x = %d, y = %d\n", x, y  );
/* x = 3, y = 5 */

swapint(x,y);

printf( "x = %d, y = %d\n", x, y  );
/* x = 5, y = 3 */

swap(int,x,y);
printf( "x = %d, y = %d\n", x, y  );
/* x = 3, y = 5 */
}

Izlaz:
max1(x,y) = 3
max1(x=5,y) = 3
max1(x++,y++) = 4
x = 3, y = 5
x = 5, y = 3
x = 3, y = 5
```