Programiranje 1

Beleške sa vežbi

Školska 2008/2009 godina

Matematički fakultet, Beograd

Jelena Graovac (Tomašević)

January 15, 2009

Sadržaj

1

Programski jezik C

[1]

1.1 Funkcije koje vrše konverziju

Primer 1 *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 2 *Konvertovanje niske cifara u ceo broj.*

```
#include<stdio.h>

/* atoi:  konvertuje s u ceo broj */
int atoi(char s[])
```

[1] Zasnovano na primerima sa sajta `http://www.matf.bg.ac.yu/~filip`, `http://www.matf.bg.ac.yu/~milan`.

```
{
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 3 *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 4 *Program vrši 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");
}
```

1.2 Sortiranje niza

Niz može biti sortiran ili uređjen u opadajućem, rastućem, neopadajućem i nerastućem poretku. Dat je algoritam za sortiranje niza koji se unosi sa ulaza u nerastućem poretku odnosno tako da važi da je `niz[0] >= niz[1] >= ... niz[n]`. Jednostavnom modifikacijom ovog algoritma niz se može sortirati i u opadajućem, rastućem ili neopadajućem poretku.

Primer 5 *Selection sort*
U prvom prolazu se razmenjuju vrednosti a[0] sa onim članovima ostatka niza koji su veći od njega. Na taj način će se posle prvog prolaza kroz niz a[0] postaviti na najveći element niza.

```
#include<stdio.h>
#define MAXDUZ 100

int main()
{
   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

   /* Dimenzija niza, pomocna i brojacke promenljive */
   int n,pom,i,j;

   printf("Unsite dimenziju niza\n");
   scanf("%d",&n);

   if (n>MAXDUZ)
   {
        printf("Nedozvoljena vrednost za n\n");
        exit(1);
   }

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

   /*Sortiranje*/
   for(i=0; i<n-1; i++)
      for(j=i+1; j<n; j++)
         if(a[i]<a[j])
         {
            pom=a[i];
            a[i]=a[j];
            a[j]=pom;
```

```
        }

   /* Ispis niza */
   printf("Sortirani niz:\n");
   for(i=0; i<n; i++)
      printf("%d\t",a[i]);

   putchar('\n');

   return 0;

}
```

1.3 Linearna i binarna pretraga niza

Primer 6 *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;
    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 7 *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. U ovom primeru
niz je uredjen u rastucem ili neopadajucem poretku.
*/
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]
                 zato sto je niz uredjen u rastucem poretku*/
                 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);
}
```

1.4 Veza između nizova i pokazivača

U C-u postoji jaka veza između pokazivača i nizova. Deklaracija

```
int a[10];
```

definiše niz a veičine 10, odnosno blok od 10 susednih objekata u memoriji sa imenima `a[0]`, `a[1]`, `a[2]`, `a[3]`, `a[4]`, `a[5]`, `a[6]`, `a[7]`, `a[8]`, `a[9]`. Zapis `a[i]` ukazuje na `i`-ti element niza `a`. Ako je `pa` pokazivač na neki ceo broj deklarisan pomoću

```
int *pa;
```

tada dodeljivanje

```
pa = &a[0];
```

podešava `pa` da pokazuje na nulti element niza `a`, to jest, pa sadrži adresu od `a[0]`. Kako je ime niza sinonim za adresu početnog elementa niza, dodela

```
pa = &a[0];
```

se može napisati i kao

```
pa = a;
```

Ako `pa` pokazuje na određeni element nekog niza, tada `pa+1` pokazuje na naredni element (ne na naredni bajt) pa tako `pa+i` pokazuje `i` elemenata iza `pa`, a `pa-i` na `i` elemenata ispred `pa`. Dakle, ako `pa` pokazuje na `a[0]` onda je

```
*(pa+i)
```

zapravo sadržaj `i`-tog elementa niza to jest isto je što i `a[i]`. A `pa+i` je isto što i `&pa[i]`.

Primer 8 *Ilustracija veze između nizova i pokazivača.*

```
#include <stdio.h>

main()
{
int a[10]={0,1,2,3,4,5,6,7,8,9};
int *pa, i;
pa=a;
printf("Vrednost od a je %p\n", a);
printf("Vrednost od pa je %p\n", pa);
printf("Vrednost od a[0] je %d\n", a[0]);
printf("Vrednost od *a je %d\n", *a);
printf("Vrednost od pa[0] je %d\n", pa[0]);
printf("Vrednost od *pa je %d\n", *pa);
```

```
printf("Vrednost od *(pa+3) je %d\n", *(pa+3));
printf("Vrednost od pa[3] je %d\n", pa[3]);
printf("Vrednost od *(a+3) je %d\n", *(a+3));
printf("Vrednost od a[3] je %d\n", a[3]);


/*Vraca se kao rezultat *pa a nakon toga se pa uveca za jedan*/
/* Ovo moze...*/
printf("Vrednost od *pa++ je %d\n", *pa++);

/* ...a ovo bi prijavilo gresku
printf("Vrednost od *a++ je %d\n", *a++); */
printf("Vrednost od *pa je %d\n", *pa);

/*Ono na sta pokazuje pa (*pa) se uveca za jedan*/
printf("Vrednost od +++pa je %d\n", +++pa);

printf("Pocetni niz sada izgleda ovako:\n");
for(i=0;i<10;i++)
 printf("%d ", a[i]);
}
Izlaz:
Vrednost od a je 0022FF40
Vrednost od pa je 0022FF40
Vrednost od a[0] je 0
Vrednost od *a je 0
Vrednost od pa[0] je 0
Vrednost od *pa je 0
Vrednost od *(pa+3) je 3
Vrednost od pa[3] je 3
Vrednost od *(a+3) je 3
Vrednost od a[3] je 3
Vrednost od *pa++ je 0
Vrednost od *pa je 1
Vrednost od +++pa je 2
Pocetni niz sada izgleda ovako:
0 2 2 3 4 5 6 7 8 9
```

Dakle, niz `a` se ponaša kao pokazivač na početni element niza. Razlika između `a` i `pa` je u tome što `a` ne može da promeni svoju vrednost (`a` uvek pokazuje na prvi element niza). Tako izraz `pa++` jeste dozvoljen (pomera `p` za jedno mesto ispred u nizu) dok `a++` nije dozvoljeno, sto se i vidi iz prethodnog primera!!!

Ako `pa` i `qa` pokazuju na elemente istog niza, tada su relacije `==, !=, <, >` dozvoljene a takođe se dva pokazivača mogu i oduzimati. Rezultat će biti broj elemenata u nizu koji ih razdvaja. Na primer, ako u gornjem primeru stavimo da je

```
int *pa, *qa;
pa=a+3; qa=a+7;
```

Onda će vrednost `qa-pa` biti `4` a vrednost `pa-qa` će biti `-4`. Logička vrednost izraza `pa<qa` biće 1 (tačno).

Primer 9 *Funkcija za izračunavanje dužine niske (stringa).*

```
Indeksna aritmetika:
int duzina_niske(char[]s)
{
int i;
for(i=0; s[i] != '\0'; i++)
  ;
return i;
}

Pokazivacka aritmetika:
int duzina_niske(char *s)
{
int i;
for(i=0; *s != '\0'; i++)
  s++;
return i;
}
```

1.5 Vraćanje nizova iz funkcije

Sledeći primer je u C-u sintaksno neispravan zato što ne može funkcija kao rezultat svog rada da vrati podatak koji je tipa int[]:

Primer 10 `int[]funkcija()`

```
{
int a[10],i;
/*Napunimo niz nekim vrednostima*/
for(i=0; i<10; i++)
a[i]=i;
return a;
}
```

Sledeći primer je za razliku od prošlog primera sintaksno ispravan ali je semantički potpuno neispravan:

Primer 11 `int* funkcija()`

```
{
int a[10],i;
/*Napunimo niz nekim vrednostima*/
for(i=0; i<10; i++)
a[i]=i;
return a;
}
```

Naime, niz `a` je lokalni podatak pa samim tim prestaje da bude vidljiv nakon napuštanja funkcije. To znači, da bi ova funkcija vratila kao rezultat adresu podatka koji na tom mestu više ne postoji, što je ozbiljan propust.

Napomena: Nikada se kao povratna vrednost funkcije ne sme vratiti adresa lokalnog podatka. Ispravno je sledeće:

Primer 12 `void funkcija(int a[], int n)`

```
{
int i;
```

```
/*Napunimo niz nekim vrednostima*/
for(i=0; i<n; i++)
a[i]=i;
/*Kako se niz funkciji prenosi preko pokazivaca,
to ce sve izmene u okviru funkcije biti vidljive i po izlasku iz nje.*/
}
```

Zadaci za vežbu:

Zadatak 1 ***I kolokvijum, februar 2005.***

1. *Napisati funkciju koja ispituje da li dve niske (koje se prenose kao parametri funkcije) su anagrami. Anagrami su niske koje se sastoje od istih karaktera. Npr. vetar, trave, verat su anagrami.*

2. *Napisati program koji testira funkciju iz prvog dela.*

Zadatak 2 ***I kolokvijum, februar 2005.*** *Napisati program koji učitava sa standardnog ulaza dve niske sa ne više od 80 karaktera u svakoj i prirodan broj* `k` *i ispisuje na standardni izlaz poruku da li se prva niska dobila cikličnim pomeranjem druge niske za* `k` *mesta. Na primer za* `k=3`*, niska CDEAB"" se dobila cikličnim pomeranjem niske "ABCDE"*

Zadatak 3 ***Jun, 2004.*** *Napisati funkciju koja koja kao argumente prihvata dve niske i proverava da li se prva od zadatih niski može dobiti cikličnim pomeranjem karaktera druge niske.*

Zadaci za praktikum:

Napomena: Ove zadatke na praktikumu rade samo studenti kod prof. Vitasa. Studenti prof. Pavlović-Lažetić rade zadatke za vežbu. Kao tekst za testiranje ovih zadataka, uzeti svoj seminarski rad.

Zadatak 4 *Napisati funkciju koja izračunava broj rečenica u tekstu. Rečenice se završavaju karakterima* `'.'`, `'?'` *ili* `'!'` *iza kojih sledi ' ' a nakon toga veliko slovo.*

Zadatak 5 *Napisati funkciju koja izračunava broj vlastitih imena ili reči izvedenih iz nekog vlastitog imena i broj broj brojeva bez obzira da li su zapisani slovima ili ciframa. Broj vlastitih imena vratiti preko liste argumenata. Napisati program koji testira rad ove funkcije.*

Zadatak 6 *Napisati program koji ispisuje redne brojeve rečenica u kojima se pojavljuje tačno jedno vlastito ime ili reč izvedena iz nekog vlastitog imena. Rečenice se završavaju karakterima* `'.'`, `'?'` *ili* `'!'` *iza kojih sledi ' ' a nakon toga veliko slovo.*