Programiranje II

Beleške sa vežbi

Smer *Informatika*

Matematički fakultet, Beograd

Sana Stojanović

06.03.08.

Sadržaj

1 Prenos matrica u funkcije

1. Napisati funkcije za učitavanje elemenata matrice i ispisivanje elemenata matrice dimenzije 3×3. Nakon toga napisati funkciju koja računa zbir elemenata na sporednoj dijagonali matrice.

   Npr. za matricu:

   ```
   1 2 3
   4 5 6
   7 8 9
   ```

   funkcija treba da vrati zbir 15.

   ```
   /* Program koristi samo staticke matrice */
   #include <stdio.h>

   /* Funkcija koja ucitava elemente matrice. */
   void unesi(int a[][3])
   {
       int i, j;
       for (i = 0; i<3; i++)
           for(j = 0; j<3; j++)
               scanf("%d", &a[i][j]);
   }

   /* Funkcija ispisuje elemente matrice dimenzije 3x3.  */
   void ispisi(int a[][3])
   {
       int i, j;
       for (i = 0; i<3; i++)
       {
           for(j = 0; j<3; j++)
               printf("%d\t", a[i][j]);

           putchar('\n');
       }
   }

   int suma(int a[][3])
   {
       int i, j;
       int s=0;

       /* kako za elemente na sporednoj dijagonali vazi da je
          zbir indeksa jednak n-1, u jednoj for petlji mozemo
   ```

```
       da prodjemo kroz elemente na sporednoj dijagonali i
       da izracunamo njihov zbir. */
    for(i=0, j=n-1; i<n; i++, j--)
        s += a[i][j];

    return s;
}

/* Funkcija koja ispisuje elemente matrice proizvoljne dimenzije */
void ispisi1(int **a, int n)
{
    int i, j;
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
            printf("%d\t", *(a+i*n+j));
        putchar('\n');
    }
}

main()
{
    /* Matrica, odnosno dvodimenzionalni niz dimenzije 3x3 */
    int a[3][3];

    /* Unosimo elemente matrice */
    unesi(a);

    /* Ispisujemo matricu */
    ispisi(a);

    /* Izracunavamo i ispisujemo sumu elemenata na sporednoj dijagonali */
    printf("%d\n", suma(a));

    /* Ispisivanje matrice na drugi nacin. Koristimo to sto su
       elementi matrice 3x3 sekvencijalno rasporedjeni u memoriji.
    /* Sada predajemo i dimenziju matrice kao argument funkciji */
    ispisi1(a, 3);

}
```

2. Napisati program koji matricu dimenzije 3×3 učitava sa standardnog ulaza i zatim ispisuje indekse onih elemenata matrice koji su jednaki sumi svojih susednih elemenata. Pod susednim elementima podrazumevamo

sve okolne elemente koji pripadaju matrici, npr:

```
za matricu:

1 2 3
4 5 6
7 8 9
```

elementu 5 susedni su elementi: 1, 2, 3, 4, 6, 7, 8 i 9
elementu 1 susedni su elementi: 2, 4 i 5
elementu 2 susedni su elementi: 1, 4, 5, 6 i 3
Slično za ostale elemente matrice.

```
#include <stdio.h>

/* Funkcija koja ucitava elemente matrice. */
void unesi(int a[][3])
{
    int i, j;
    for (i = 0; i<3; i++)
        for(j = 0; j<3; j++)
            scanf("%d", &a[i][j]);
}

/* Funkcija ispisuje elemente matrice dimenzije 3x3.  */
void ispisi(int a[][3])
{
    int i, j;
    for (i = 0; i<3; i++)
    {
        for(j = 0; j<3; j++)
            printf("%d\t", a[i][j]);

        putchar('\n');
    }
}

/* Funkcija koja proverava da li se zadati indeksi nalaze u okvirima
   matrice i ako se nalaze vraca element koji se nalazi na zadatoj
   poziciji odnosno nulu ako su indeksi van okvira matrice */
int f(int a[][3], int i, int j)
{
    if (i>=0 && i<n && j>=0 && j<n)
        return a[i][j];
    else
        return 0;
```

```
}

main()
{
    int a[3][3];
    int i, j, suma, n;

    /* Unosimo elemente matrice dimenzije 3x3 */
    unesi(a);

    n=3;

    /* Za sve elemente matrice... */
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
        {
            /* ...odredjujemo sumu okolnih elemenata */
            suma = f(a, i-1, j-1) + f(a, i-1, j) + f(a, i-1, j+1) +
                   f(a, i,   j-1) +                 f(a, i,   j+1) +
                   f(a, i+1, j-1) + f(a, i+1, j) + f(a, i+1, j+1);

            /* Ako je tekuci element jednak sumi okolnih elemenata
               stampamo indekse tog elementa */
            if (a[i][j] == suma)
                printf("Element a[%d][%d] = %d je jednak sumi svojih okolnih
                        elemenata\n", i, j, a[i][j]);
        }
}
```

2 Pokazivači - veza sa nizovima

1. Osnovna pokazivačka aritmetika.

```
#include <stdio.h>

main()
{
    char s[] = "abcde";
    int t[] = {1, 2, 3, 4, 5};

    /* Inicijalizujmo pokazivace ps i pt na pocetke nizova s i t */
    char* ps = &s[0];
    int* pt =  &t[0];

    /* Pokazivace je moguce sabirati sa celim brojevima i
```

```
        od njih je moguce oduzimati cele brojeve*/
    /* Ispisimo vrednosti pokazivaca */
    printf("ps = %p\n", ps);
    printf("ps+1 = %p\n", ps+1);
    printf("ps+2 = %p\n", ps+2);
    printf("ps-1 = %p\n", ps-1);
    printf("ps-2 = %p\n", ps-2);


    /* Prilikom sabiranja pokazivaca i celih brojeva,
       dodaje se velicina odgovarajuceg tipa. */
    printf("pt = %p\n", pt);
    printf("pt+1 = %p\n", pt+1);
    printf("pt+2 = %p\n", pt+2);
    printf("pt-1 = %p\n", pt-1);
    printf("pt-2 = %p\n", pt-2);

    /* Na pokazivace je moguce primenjivati i operatore ++ i -- */
    for (ps = s; *ps; ps++)
        putchar(*ps);
    putchar('\n');

    /* Slicno, dva pokazivaca istog tipa se mogu oduzimati. Prilikom
       izracunavanja rezultata, uzima se u obzir velicina tipa. */
    ps = &s[3];
    printf("s = %p\n", s);
    printf("ps = %p\n", ps);
    printf("ps - s = %d\n", ps - s);
    pt = &t[3];
    printf("t = %p\n", t);
    printf("pt = %p\n", pt);
    printf("pt - t = %d\n", pt - t);

}
```

2. Ilustracija veze između pokazivača i nizova.

```
#include <stdio.h>

void print_array(int* pa, int n);

main()
{
    int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

```
int num_of_elements = sizeof(a)/sizeof(int);
int* pa;

/* Niz je isto sto i adresa prvog elementa */
printf("Niz a : %p\n", a);
printf("Adresa prvog elementa niza a (&a[0]) : %p\n", &a[0]);

/* Moguce je dodeliti niz pokazivacu odgovarajuceg tipa */
pa = a;

printf("Pokazivac pa ukazuje na adresu : %p\n", pa);

/* Nizu nije moguce dodeliti pokazivacku promenljivu
  (nizove mozemo smatrati KONSTANTNIM pokazivacima na prvi element) */
/* a = pa; */

/* Niz je moguce koristiti kao pokazivac tj. vaze pravila pokazivacke
   aritmetike */
printf("a + 3 = %p\n", a + 3);

/* Vazi da je
        a + i =  &a[i]
   odnosno
                *(a + i) = a[i]
*/
printf("&a[3] = %p\n", &a[3]);

/* Identiteti
        a + i =  &a[i]
   odnosno
                *(a + i) = a[i]
   vazi i za pokazivace i za nizove */

/* Pokazivace je na osnovu prethodnog moguce indeksirati kao nizove */
printf("pa[5] = %d\n", pa[5]);
printf("*(pa + 5) = %d\n", *(pa+5));


/* Medjutim, sizeof(pa) je samo velicina pokazivaca, a ne niza */
printf("sizeof(a) = %d\n", sizeof(a));
printf("sizeof(pa) = %d\n", sizeof(pa));

/* Pozivamo funkciju za stampanje niza i saljemo joj niz */
print_array(a, num_of_elements);
/* Pozivamo funkciju za stampanje niza i saljemo joj pokazivac na
   pocetak niza */
```

```
    print_array(pa, num_of_elements);
}

/* Proslednjivanje niza u funkciju
    void print_array(int pa[], int n);
   je ekvivalentno prosledjivanju pokazivaca u funkciju
    void print_array(int* pa, int n);
   Izmedju ovih konstrukcija nema nikakve razlike.
*/
void print_array(int* pa, int n)
{
    int i;
    for (i = 0; i<n; i++)
        printf("%d ", pa[i]);
    putchar('\n');
}
```

3. Napisti funkcije za rad sa niskama korišćenjem pokazivačke aritmetike.

```
/* strlen, strcpy, strcat, strcmp, strchr, strstr, ... -
   verzije sa pokazivacima */
/* Vezbe radi, implementirane su funkcije biblioteke string.h */
#include <stdio.h>
#include <stdlib.h> /* Zbog NULL */

/* Izracunava duzinu stringa */
int string_length(char *s)
{
    char* t;
    for (t = s; *t; t++)
        ;
    return t - s;
}


/* Kopira string src u string dest. Pretpostavlja da u dest ima
   dovoljno prostora. */
void string_copy(char *dest, char *src)
{
    /* Kopira karakter po karakter, sve dok nije iskopiran karakter '\0' */
    while(*dest++ = *src++)
        ;
}
```

```
/* Nadovezuje string t na kraj stringa s. Pretpostavlja da u s ima
   dovoljno prostora. */
void string_concatenate(char *s, char *t)
{
    /* Pronalazimo kraj stringa s */
    while (*s)
        s++;

    /* Vrsi se kopiranje, slicno funkciji string_copy */
    while (*s++ = *t++)
        ;
}

/* Vrsi leksikografsko poredjenje dva stringa.
    Vraca :
        0 - ukoliko su stringovi jednaki
        <0 - ukoliko je s leksikografski ispred t
        >0 - ukoliko je s leksikografski iza t
*/
int string_compare(char *s, char *t)
{
    /* Petlja tece sve dok ne naidjemo na prvi razliciti karakter */
    for (; *s == *t; s++, t++)
        if (*s == '\0') /* Naisli smo na kraj oba stringa, a nismo nasli
                           razliku */
            return 0;


    /* *s i *t su prvi karakteri u kojima se niske razlikuju.
       Na osnovu njihovog odnosa, odredjuje se odnos stringova */
    return *s - *t;
}

/* Pronalazi prvu poziciju karaktera c u stringu s, i vraca
   pokazivac na nju, odnosno NULL ukoliko s ne sadrzi c */
char* string_char(char *s, char c)
{
    for (; *s; s++)
        if (*s == c)
            return s;

    /* Nije nadjeno */
    return NULL;
}

/* Pronalazi poslednju poziciju karaktera c u stringu s, i vraca
```

```
   pokazivac na nju, odnosno NULL ukoliko s ne sadrzi c */
char* string_last_char(char *s, char c)
{
    char *t = s;
    /* Pronalazimo kraj stringa s */
    while (*t++)
        ;

    /* Krecemo od kraja i trazimo c unazad */
    for (t--; t >= s; t--)
        if (*t == c)
            return t;

    /* Nije nadjeno */
    return NULL;
}

main()
{
    char s[100];
    char t[] = "Zdravo";
    char u[] = " svima";
    char r[] = "racunari";

    string_copy(s, t);
    printf("%s\n", s);

    string_concatenate(s, u);
    printf("%s\n", s);

    printf("%d\n",string_char(r, 'n') - r);
    printf("%d\n",string_last_char(r, 'a') - r);
}
```