Osnovi programiranja

Beleške sa vežbi

Smer *Računarstvo i informatika*
Matematički fakultet, Beograd

Jelena Tomašević

April 9, 2006

Sadržaj

1

Programski jezik C

[1]

1.1 Pokazivači na funkcije

Primer 1 *Program demonstrira upotrebu pokazivača na funkcije.*

```
#include <stdio.h>

int kvadrat(int n) { return n*n; }

int kub(int n) { return n*n*n; }

int parni_broj(int n) { return 2*n; }

/* Funkcija izracunava sumu od 1 do n f(i),
   gde je f data funkcija */
int sumiraj(int (*f) (int), int n) {
    int i, suma=0;
    for (i=1; i<=n; i++)
        suma += (*f)(i);

    return suma;
}

main() {
printf("Suma kvadrata brojeva od jedan do 3 je %d\n", sumiraj(kvadrat,3));
printf("Suma kubova brojeva od jedan do 3 je %d\n", sumiraj(kub,3));
printf("Suma prvih pet parnih brojeva je %d\n", sumiraj(parni_broj,5));
} /*Izlaz: Suma kvadrata brojeva od jedan do 3 je 14 Suma kubova
brojeva od jedan do 3 je 36 Suma prvih pet parnih brojeva je 30 */
```

1.2 Matrice - uvežbavanje

Primer 2 *Program ilustruje rad sa kvadratnim matricama i relacijama. Elementi i je u relaciji sa elementom j ako je m[i][j] = 1, a nisu u relaciji ako je m[i][j] = 0.*

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

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

/* Dinamicka matrica je odredjena adresom
   pocetka niza pokazivaca i dimenzijama tj.
   int** a;
   int m,n;
*/

/* Alokacija kvadratne matrice nxn */
int** alociraj(int n)
{
    int** m;
    int i;
    m=malloc(n*sizeof(int*));
    if (m == NULL)
    {
        printf("Greska prilikom alokacije memorije!\n");
        exit(1);
    }

    for (i=0; i<n; i++)
        {
            m[i]=malloc(n*sizeof(int));
            if (m[i] == NULL)
            {
            int k;
            printf("Greska prilikom alokacije memorije!\n");
            for(k=0;k<i;k++)
                    free(m[k]);
            exit(1);
            }
        }

    return m;
}

/* Dealokacija matrice dimenzije nxn */
void obrisi(int** m, int n)
{
    int i;
    for (i=0; i<n; i++)
       free(m[i]);
    free(m);
}

/* Ispis matrice /
void ispisi_matricu(int** m, int n)
{
     int i, j;
      for (i=0; i<n; i++)
      {
```

```
         for (j=0; j<n; j++)
             printf("%d ",m[i][j]);
         printf("\n");
      }
}

/* Provera da li je relacija predstavljena matricom refleksivna */
int refleksivna(int** m, int n)
{
    int i;
    for (i=0; i<n; i++)
       if (m[i][i]==0)
           return 0;

    return 1;
}

/* Provera da li je relacija predstavljena matricom simetricna */
int simetricna(int** m, int n)
{
    int i,j;
    for (i=0; i<n; i++)
      for (j=i+1; j<n; j++)
         if (m[i][j]!=m[j][i])
             return 0;
    return 1;
}

/* Provera da li je relacija predstavljena matricom tranzitivna*/
int tranzitivna(int** m, int n)
{
    int i,j,k;

    for (i=0; i<n; i++)
       for (j=0; j<n; j++)
           for (k=0; k<n; k++)
               if ((m[i][j]==1)
                   && (m[j][k]==1)
                   && (m[i][k]!=1))
                     return 0;
    return 1;
}

/* Pronalazi najmanju simetricnu relaciju koja sadrzi relaciju a
*/
void simetricno_zatvorenje(int** a, int n)
{
   int i,j;
   for (i=0; i<n; i++)
     for (j=0; j<n; j++)
     {
        if (a[i][j]==1 && a[j][i]==0)
```

```
                    a[j][i]=1;
          if (a[i][j]==0 && a[j][i]==1)
                    a[i][j]=1;
      }
}

main() {
       int **m;
       int n;
       int i,j;

       printf("Unesi dimenziju matrice : ");
       scanf("%d",&n);
       m=alociraj(n);

       for (i=0; i<n; i++)
          for (j=0; j<n; j++)
             scanf("%d",&m[i][j]);

       printf("Uneli ste matricu : \n");

       ispisi_matricu(m,n);

       if (refleksivna(m,n))
            printf("Relacija je refleksivna\n");
       if (simetricna(m,n))
            printf("Relacija je simetricna\n");
       if (tranzitivna(m,n))
            printf("Relacija je tranzitivna\n");

       simetricno_zatvorenje(m,n);

       ispisi_matricu(m,n);

       obrisi(m,n);
}
```

Primer 3 *Izračunati vrednost determinante matrice preko Laplasovog razvoja.*

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

/* Funkcija alocira matricu dimenzije nxn */
int** allocate(int n)
{
     int **m;
     int i;
     m=(int**)malloc(n*sizeof(int*));
     if (m == NULL) {
        printf("Greska prilikom alokacije memorije!\n");
        exit(1);
}
```

```
    for (i=0; i<n; i++)
        {
        m[i]=malloc(n*sizeof(int));
        if (m[i] == NULL)
         {
         int k;
         for(k=0;k<i;k++)
                free(m[k]);
         printf("Greska prilikom alokacije memorije!\n");
         exit(1);
         }
        }

     return m;
}

/* Funkcija vrsi dealociranje date matrice dimenzije n */ void
deallocate(int** m, int n)
{
     int i;
     for (i=0; i<n; i++)
        free(m[i]);
     free(m);
}

/* Funkcija ucitava datu alociranu matricu sa standardnog ulaza */
void ucitaj_matricu(int** matrica, int n)
{
     int i,j;
     for (i=0; i<n; i++)
        for (j=0; j<n; j++)
           scanf("%d",&matrica[i][j]);
}

/* Rekurzivna funkcija koja vrsi Laplasov razvoj */
int determinanta(int** matrica, int n)
{
    int i;
    int** podmatrica;
    int det=0,znak;

    /* Izlaz iz rekurzije je matrica 1x1 */
    if (n==1)
    return matrica[0][0];

    /* Podmatrica ce da sadrzi minore polazne matrice */
    podmatrica=allocate(n-1);
    znak=1;
    for (i=0; i<n; i++)
    {
    int vrsta,kolona;
    for (kolona=0; kolona<i; kolona++)
```

```
        for(vrsta=1; vrsta<n; vrsta++)
            podmatrica[vrsta-1][kolona] = matrica[vrsta][kolona];
    for (kolona=i+1; kolona<n; kolona++)
        for(vrsta=1; vrsta<n; vrsta++)
            podmatrica[vrsta-1][kolona-1] = matrica[vrsta][kolona];

    det+= znak*matrica[0][i]*determinanta(podmatrica,n-1);
    znak*=-1;
    }
    deallocate(podmatrica,n-1);
    return det;
}

main()
{
   int **matrica;
   int n;

   scanf("%d", &n);
   matrica = allocate(n);
   ucitaj_matricu(matrica, n);
   printf("Determinanta je : %d\n",determinanta(matrica,n));
   deallocate(matrica, n);
}
```

1.3 Dinamički niz

Primer 4 *Ilustracija dinamičkog niza.*

```
/* Program za svaku rec unetu sa standardnog
   ulaza ispisuje broj pojavljivanja.
   Verzija sa dinamickim nizom i realokacijom.
*/

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

/* Rec je opisana imenom i brojem
   pojavljivanja */
typedef struct _rec
{
    char ime[80];
    int br_pojavljivanja;
} rec;

/* Dinamicki niz reci je opisan pokazivacem na
   pocetak, tekucim brojem upisanih elemenata i
   tekucim brojem alociranih elemenata */
rec* niz_reci;
int duzina=0;
```

```
int alocirano=0;

/* Realokacija se vrsi sa datim korakom */
#define KORAK 10

/* Funkcija ucitava rec i vraca njenu duzinu ili
   -1 ukoliko smo dosli do znaka EOF*/
int getword(char word[],int max)
{
    int c, i=0;

    while (isspace(c=getchar()))
         ;

    while(!isspace(c) && c!=EOF && i<max-1)
    {
          word[i++]=c;
          c = getchar();
    }

    word[i]='\0';

    if (c==EOF) return -1;
        else return i;
}

main()
{
char procitana_rec[80];
int i;
while(getword(procitana_rec,80)!=-1)
{
/* Proveravamo da li rec vec postoji u nizu */

   for (i=0; i<duzina; i++)
    /* Ako bi smo uporedili
    procitana_rec == niz_reci[i].ime
    bili bi uporedjeni pokazivaci a ne
    odgovarajuci sadrzaji!!!
    Zato koristimo strcmp. */
    if (strcmp(procitana_rec,
                    niz_reci[i].ime)==0)
    {
    niz_reci[i].br_pojavljivanja++;
    break;
    }

/* Ukoliko rec ne postoji u nizu */
if (i==duzina) {
    rec nova_rec;
    /* Ako bi smo dodelili
        nova_rec.ime = procitana_rec
```

```
        izvrsila bi se dodela pokazivaca
        a ne kopiranje niske procitana_rec
        u nova_rec.ime.
        Zato koristimo strcpy!!! */
    strcpy(nova_rec.ime,procitana_rec);
    nova_rec.br_pojavljivanja=1;

    /* Ukoliko je niz "kompletno popunjen"
        vrsimo realokaciju */
    if (duzina==alocirano)
    {
    alocirano+=KORAK;

    /* Sledeca linija zamenjuje blok
        koji sledi i moze se
        koristiti alternativno. Blok je
        ostavljen samo da bi
        demonstrirao korisnu tehniku */
    /*
    niz_reci=realloc(niz_reci,
                    (alocirano)*sizeof(rec)); */

    {
    /* alociramo novi niz, veci
        nego sto je bio prethodni */
    rec* novi_niz=(rec *)malloc(alocirano*sizeof(rec));

    /* Kopiramo elemente starog niza u novi */
    for (i=0; i<duzina; i++)
            novi_niz[i]=niz_reci[i];
    /* Uklanjamo stari niz */
    free(niz_reci);
    /* Stari niz postaje novi */
    niz_reci=novi_niz;
    }

    if (niz_reci==NULL)
    {
    printf("Greska prilikom
                    alokacije memorije");
    exit(1);
    }
    }
    /* Upisujemo rec u niz */
    niz_reci[duzina]=nova_rec;
    duzina++;
} }

/* Ispisujemo elemente niza */ for(i=0; i<duzina; i++)
    printf("%s - %d\n",niz_reci[i].ime,
                niz_reci[i].br_pojavljivanja);
```

```
free(niz_reci); }
```

1.4 Zadaci za vežbu

Zadatak 1 *Ispisati na izlaz sumu celih brojeva koji se unose kao argumenti komandne linije.*

Zadatak 2 *Napisati funkciju koja omogućava računanje proizvoda dve kvadratne matrice dimenzija $n \times n$. Napisati program koji omogućava unošenje dve kvadratne matrice i štampanje proizvoda te dve matrice.*

Zadatak 3 ***Jun, 2004.*** *Napisati funkciju koja računa multiplikativnu otpornost datog pozitivnog broja. Multiplikativna otpornost se računa na sledeći način $n_0 = n$, n_k je jednak proizvodu cifara broja n_{k-1}, $k = 1, 2...$, multiplikativna otpornost je najmanje k za koje je n_k jednocifren broj. Napisati program koji iz datoteke čije se ime zadaje na ulazu čita brojeve, gde su brojevi zapisani po jedan u svakom redu i u drugu datoteku čije se ime zadaje takođe na ulazu upisuje red po red date brojeve i njihovu multiplikativnu otpornost.*