Programiranje 2

Beleške sa vežbi

Školska 2006/2007 godina

Matematički fakultet, Beograd

Jelena Tomašević

April 9, 2007

Sadržaj

1

Programski jezik C

[1]

1.1 Alokacija memorije

Funkcija **void* malloc(unsigned n)** vraća pokazivač na blok od n susednih bajtova neinicijalizovane memorije ili `NULL` ukoliko zahtev ne može da se ispuni.

Funkcija **void* calloc(unsigned n, unsigned velicina)** vraća pokazivač na memorijski prostor dovoljno velik za n objekata navedene veličine, ili NULL ako zahtev ne može biti ispunjen. Rezervisani memorijski prostor se inicijalizuje nulama.

Funkcija **void* realloc(void *p, unsigned velicina)** menja veličinu objekta na koga pokazuje p tako što ga postavlja na vrednost velicina. Njegov sadržaj će ostati nepromenjen do manje od dve veličine – stare ili nove. Ako je nova veličina veća, novi prostor se ne inicijalizuje. Funkcija realloc vraća pokazivač na novi prostor, ili NULL ako zahtev ne može biti ispunjen i u tom slučaju *p ostaje nepromenjeno.

Funkcija **void free(void *p)** oslobađa memorijski prostor na koji pokazuje p. Na ne radi ništa ako je p jednako NULL. p mora biti pokazivač na memorijski prostor koji je prethodno alociran funkcijama calloc, malloc ili realloc.

Za korišćenje ovih funkcija neophodno je uključiti zaglavlje `stdlib.h`.

Primer 1 *Demonstracija funkcije* `malloc`

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

main()
{
int n;
int i;
int *a;

printf("Unesi broj clanova niza : ");
scanf("%d", &n);

/* Kao da je moglo da se uradi
    int a[n];
```

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

```
*/
a = (int*)malloc(n*sizeof(int));

/* Kad god se vrsi alokacija memorije mora se proveriti da li je ona
   uspesno izvrsena!!! */
if (a == NULL)
{
    printf("Nema slobodne memorije\n");
    exit(1);
}

/* Od ovog trenutka a koristimo kao obican niz */
for (i = 0; i<n; i++)
    scanf("%d",&a[i]);

/* Stampamo niz u obrnutom redosledu */
for(i = n-1; i>=0; i--)
    printf("%d",a[i]);

/* Oslobadjamo memoriju*/
free(a);
}
```

Primer 2 *Demonstracija funkcije* `calloc` *- funkcija inicijalizuje sadrzaj memorije na 0.*

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

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

printf("Unesi broj clanova niza : ");
scanf("%d", &n);

/* Niz m NE MORA garantovano da ima sve nule */
m = malloc(n*sizeof(int));
if (m == NULL) {
                printf("Greska prilikom alokacije memorije!\n");
                exit(1);
               }

/* Niz c MORA garantovano da ima sve nule */
c = calloc(n, sizeof(int));
if (c == NULL) {
                printf("Greska prilikom alokacije memorije!\n");
                free(m);
                exit(1);
                }
for (i = 0; i<n; i++)
    printf("m[%d] = %d\n", i, m[i]);

for (i = 0; i<n; i++)
```

```
    printf("c[%d] = %d\n", i, c[i]);

free(m);
free(c);
}
```

1.2 Dinamički niz

Primer 3 *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 bismo 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 bismo 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;
    niz_reci=realloc(niz_reci, (alocirano)*sizeof(rec));

    /* Ovo je ekvivalentno kao da smo napisali sledeci blok naredbi:
    {
    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;
    }*/

    /* Nastavljamo dalje: */

    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); }
```

1.3 Niz pokazivača

Primer 4

```
#include <stdio.h>
#include <stdlib.h>
main()
{
/* Niz od tri elemenata tipa int*/
int nizi[3];

/* Niz od tri elemenata tipa int*, dakle
  niz od tri pokazivaca na int*/
int* nizip[3];

/* Alociramo memoriju za prvi element niza*/
nizip[0] = (int*) malloc(sizeof(int));
if (nizip[0] == NULL)
{
    printf("Nema slobodne memorije\n");
    exit(1);
}
/* Upisujemo u prvi element niza broj 5*/
*nizip[0] = 5;
printf("%d", *nizip[0]);
```

```
/* Alociramo memoriju za drugi element niza.
   Drugi element niza pokazuje na niz od dva
   elementa*/
nizip[1] = (int*) malloc(2*sizeof(int));
if (nizip[1] == NULL) {
    printf("Nema slobodne memorije\n");
    free(nizip[0]);
    exit(1);
}

/* Pristupamo prvom elementu na koji pokazuje
   pokazivac nizip[1]*/
*(nizip[1]) = 1;

/* Pristupamo sledecem elementu u nizu na koji pokazuje
   nizip[1].
*/
*(nizip[1] + 1) = 2;

printf("%d", nizip[1][1]);

/* Alociramo memoriju za treci element niza nizip. */
nizip[2] = (int*) malloc(sizeof(int));
if (nizip[2] == NULL) {
    printf("Nema slobodne memorije\n");
    free(nizip[0]);
    free(nizip[1]);
    exit(1);
}

*(nizip[2]) = 2;

printf("%d", *(nizip[2]));

free(nizip[0]);
free(nizip[1]);
free(nizip[2]);
}
```

Primer 5

```
#include <stdio.h>
#include <stdlib.h>
main()
{
/* Niz karaktera*/
char nizc[5];

/* Niz karaktera od cetiri elementa
  ('A', 'n', 'a', '\0')*/
char nizcc[]="Ana";
printf("%s", nizcc);
```

```
/* Niz od tri pokazivaca. Prvi pokazuje na
   nisku karaktera Kruska, drugi na nisku karaktera
   Sljiva a treci na Ananas. */
char* nizcp[]={"Kruska", "Sljiva", "Ananas"};

printf("%s", nizcp[0]);
printf("%s", nizcp[1]);
printf("%s", nizcp[2]);
}
```

1.4 Matrice

Primer 6 *Statička alokacija prostora za matricu.*

```
#include <stdio.h>

main()
{
int a[3][3] = {{0, 1, 2}, {10, 11, 12}, {20, 21, 22}};
int i, j;

/* Alternativni unos elemenata matrice
for(i=0; i<3; i++)
    for(j=0; j<3; j++)
        {
        printf("a[%d][%d] = ", i, j);
        scanf("%d", &a[i][j]);
        }
*/

a[1][1] = a[0][0] + a[2][2];
/* a[1][1] = 0 + 22 = 22 */

printf("%d\n", a[1][1]);   /* 22 */

/* Stampanje elemenata matrice*/
for(i=0; i<3; i++)
    {
    for(j=0; j<3; j++)
        printf("%d\t", a[i][j]);
    printf("\n");
    }
}
```

Nama je potrebno da imamo veću fleksibilnost, tj da se dimenzije matrice mogu uneti kao parametri našeg programa. Zbog toga je neophodno koristiti dinamicku alokaciju memorije.

Primer 7 *Implementacija matrice preko niza.*

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

```
/* Makro pristupa clanu na poziciji i, j matrice koja ima
   m vrsta i n kolona */
#define a(i,j) a[(i)*n+(j)]

main()
{
   /* Dimenzije matrice */
   int m, n;

   /* Matrica */
   int *a;

   int i,j;

   /* Suma elemenata matrice */
   int s=0;

   /* Unos i alokacija */
   printf("Unesi broj vrsta matrice : ");
   scanf("%d",&m);

   printf("Unesi broj kolona matrice : ");
   scanf("%d",&n);

   a=malloc(m*n*sizeof(int));
   if (a == NULL) {
                   printf("Greska prilikom alokacije memorije!\n");
                   exit(1);
                   }

   for (i=0; i<m; i++)
      for (j=0; j<n; j++)
      {
      printf("Unesi element na poziciji (%d,%d) : ",i,j);
      scanf("%d",&a(i,j));
      }

   /* Racunamo sumu elemenata matrice */
   for (i=0; i<m; i++)
      for (j=0; j<n; j++)
            s+=a(i,j);

   /* Ispis unete matrice */
   printf("Uneli ste matricu : \n");
   for (i=0; i<m; i++)
   {   for (j=0; j<n; j++)
          printf("%d ",a(i,j));
       printf("\n");
   }

   printf("Suma elemenata matrice je %d\n", s);
```

```
   /* Oslobadjamo memoriju */
   free(a);
}
```

Primer 8 *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.*

```
#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 9 *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);
}
```

1.5 Zadaci za vežbu

Zadatak 1 *(a) Napisati program koji omogućava unos dimenzije kvadratne matrice a zatim i unos elemenata te matrice sa standardnog ulaza.*

(b) Napisati funkciju koja računa zbir elemenata kvadratne matrice dimenzije $n \times n$.

(c) Napisati funkciju koja računa proizvod elemenata ispod glavne dijagonale matrice dimenzija $n \times n$.

(d) Napisati funkciju koja omogućava računanje proizvoda dve kvadratne matrice dimenzija $n \times n$.

(e) Napisati program koji omogućava unošenje dve kvadratne matrice i štampanje zbira, proizvoda elemenata te dve matrice kao i proizvoda elemenata ispod glavne dijagonale svake od tih matrica.