Osnovi programiranja

Beleške sa vežbi

Smer *Računarstvo i informatika*
Matematički fakultet, Beograd

Jelena Tomašević

April 3, 2006

Sadržaj

1

Programski jezik C

[1]

1.1 Argumenti komandne linije

Primer 1 *Ilustracija rada sa argumentima komandne linije.*

```
/* Program pozivati sa npr.:
        ./a.out
        ./a.out prvi
        ./a.out prvi drugi treci
        ./a.out -a -bc ime.txt
*/

#include <stdio.h>

/* Imena ovih promenljivih mogu biti proizvoljna. Npr.

        main (int br_argumenata, char* argumenti[]);

   ipak, uobicajeno je da se koriste sledeca imena:
*/

main(int argc, char* argv[])
{
        int i;

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

Primer 2 *Program ispisuje opcije navedene u komandnoj liniji.* K&R *rešenje.*

[1] Preuzeto sa sajta http://www.matf.bg.ac.yu/∼milena

```
/* Opcije se navode koriscenjem znaka -,  pri cemu je moguce da iza jednog -
   sledi i nekoliko opcija.
   Npr. za -abc -d -fg su prisutne opcije a b c d f g  */

/* Resnje se intenzivno zasniva na pokazivackoj aritmetici i prioritetu operatora */

#include <stdio.h>

int main(int argc, char* argv[])
{
        char c;
        /* Dok jos ima argumenata i dok je karakter na poziciji 0 upravo crtica */
        while(--argc>0 && (*++argv)[0]=='-')
                /* Dok god ne dodjemo do kraja tekuceg stringa */
                while (c=*++argv[0])
                        printf("Prisutna opcija : %c\n",c);

}

Izlaz:
Prisutna opcija : a
Prisutna opcija : b
Prisutna opcija : c
Prisutna opcija : d
Prisutna opcija : f
Prisutna opcija : g
```

Primer 3 *Program ispisuje opcije navedene u komandnoj liniji - jednostavnija verzija.*

```
#include <stdio.h>

main(int argc, char* argv[])
{
        /* Za svaki argument komande linije, pocevsi od argv[1]
           (preskacemo ime programa) */
        int i;
        for (i = 1; i < argc; i++)
        {
                /* Ukoliko i-ti argument pocinje crticom */
                if (argv[i][0] == '-')
                {       /* Ispisujemo sva njegova slova pocevsi od pozicije 1 */
                        int j;
                        for (j = 1; argv[i][j] != '\0'; j++)
                                printf("Prisutna je opcija : %c\n", argv[i][j]);
                }
                /* Ukoliko ne pocinje crticom, prekidamo */
                else
                        break;
        }
}
```

Primer 4 *Napisati program koji sa standardnog ulaza učitava pozitivan ceo broj, a na standardni izlaz ispisuje vrednost tog broja sa razmenjenim vrednostima bitova na poziciji* `i`, `j`. *Pozicije* `i`, `j`

se učitavaju kao parametri komandne linije. Smatrati da krajnji desni bit binarne reprezentacije je 0-ti bit. Pri rešavanju nije dozvoljeno koristiti pomoćni niz niti aritmetičke operatore `+,-,/,*,%`.

```
#include <stdio.h>
unsigned Trampa(unsigned n, int i, int j);

main(int argc, char **argv)
{
    unsigned x; /*broj sa standardnog ulaza ciji se bitovi razmenjuju*/
     int i,j;   /*pozicije bitova za trampu*/

     /*ocitavanje parametara komandne linije i broja sa standarnog ulaza*/
   sscanf(argv[1], "%d", &i);
   sscanf(argv[2], "%d", &j);
   scanf("%u", &x);

   printf("\nNakon trampe vrednost unetog broja je %u\n", Trampa(x,i,j));

}
unsigned Trampa(unsigned n, int i, int j)
{
   //ako se bit na poziciji i razlikuje od bita na poziciji j, treba ih invertovati
   if (  ((n>>i)&1)  !=   ((n>>j)&1)  )   n^= (1<<i) | (1<<j);
   return n;
}
```

Primer 5 *Iz datoteke čije se ime zadaje kao argrument komandne linije, učitati cele brojeve sve dok se ne učita nula, i njihov zbir ispisati u datoteku čije se ime takođe zadaje kao argument komandne linije.*

```
#include<stdio.h>
main(int argc, char* argv[])
{
    int n, S=0;
    FILE* ulaz, *izlaz;
    /* Ukoliko su imena datoteka navedena kao argumenti...*/
    if (argc>=3)
    {
        /* ...otvaramo datoteku i proveravamo da li smo uspeli */
         if ( (ulaz = fopen(argv[1], "r")) == NULL)
             printf("Greska : datoteka %s ne moze biti otvorena\n", argv[1]);
         if ( (izlaz = fopen(argv[2], "w")) == NULL)
            printf("Greska : datoteka %s ne moze biti otvorena\n", argv[2]);
    }
    else
    {
        char ime_datoteke_ulaz[256], ime_datoteke_izlaz[256];
        /* Ucitavamo ime datoteke */
        printf("U kojoj datoteci se nalaze brojevi: ");
        scanf("%s", ime_datoteke_ulaz);
        /* Otvaramo datoteku i proveravamo da li smo uspeli */
        if ( (ulaz = fopen(ime_datoteke_ulaz, "r")) == NULL)
            printf("Greska : datoteka %s ne moze biti otvorena\n", ime_datoteke_ulaz);
```

```
        printf("U kojoj datoteci treba ispisati rezultat: ");
        scanf("%s", ime_datoteke_izlaz);
        /* Otvaramo datoteku i proveravamo da li smo uspeli */
        if ( (izlaz = fopen(ime_datoteke_izlaz, "w")) == NULL)
            printf("Greska : datoteka %s ne moze biti otvorena\n", ime_datoteke_izlaz);
    }

    fscanf(ulaz, "%d", &n);
    while(n!=0)
    {
        S+=n;
        fscanf(ulaz, "%d", &n);
    }
    fprintf(izlaz,"Suma brojeva ucitanih iz datoteke je %d.", S);
    return 0;
}
```

1.2 Alokacija memorije

`void* malloc(size_t n)` vraća pokazivač na n bajtova neinicijalizovane memorije ili `NULL` ukoliko zahtev ne može da se ispuni.

Za njeno korišćenje neophodno je uključiti zaglavlje `stdlib.h`. Oslobađanje memorije - funkcija `free`.

Ne sme se koristiti nešto što je već oslobođeno, ne sme se dva puta oslobađati ista memorija.

Primer 6

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

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

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

/* Kao da ste mogli da uradite
    int a[n];
*/
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 koristim 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 7 *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);
}
```

1.3 Niz pokazivača

Primer 8

```
#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 9

```
#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 Pokazivači na funkcije

Primer 10 *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
*/
```

1.5 Matrice

Primer 11 *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 12 *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 13 *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 14 *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);
}
```

1.6 Zadaci za vežbu

Zadatak 1 *Napisati program koji omogućava unos dimenzije kvadratne matrice i unos elemenata matrice sa standardnog ulaza.*

1. *Napisati funkciju koja računa zbir elemenata matrice dimenzija $n \times m$.*

2. *Napisati funkciju koja računa proizvod elemenata ispod glavne dijagonale matrice dimenzija $n \times n$.*

Program treba da odštampa zbir elemenata matrice i proizvod elemenata ispod glavne dijagonale.

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.*