Programiranje I

Beleške sa vežbi

Smer *Informatika*

Matematički fakultet, Beograd

Sana Stojanović

January 19, 2008

Sadržaj

1 Funkcije sa nizovima

1. Program izračunava skalarni proizvod dva niza celih brojeva.

```
#include <stdio.h>
/* Funkcija koja izracunava skalarni proizvod dva niza kao argumente
   ocekuje nizove x i y i dimenziju niza n (potrebna je samo dimenzija
   jednog niza posto nizovi moraju biti iste dimenzije da bi mogli
   da izracunamo skalarni proizvod */
long skalar(int x[],int y[],int n);
main()
{
    int a[]={1,2,3,4,5,6}, b[]={8,7,6,5,4,3};
    printf("Skalarno a*b= %ld\n",skalar(a,b,6));
}

long mnozi(int x[],int y[],int n)
{
    int i;
    long suma=0;
    for(i=0;i<n;i++)
        suma=suma+x[i]*y[i];
    return suma;
}
```

```
Izlaz: Skalarno a*b= 98
```

2. Napisati program koji množi elemente niza (dimenzije manje ili jednake 10) koji se zadaje sa standardnog ulaza (tako što se prvo navede dimenzija niza pa zatim njegovi članovi) nekim celim brojem koji se takođe zadaje preko standardnog ulaza.

```
#include <stdio.h>
void mnozi(int a[], int n, int c)
{
    int i;
    for(i=0;i<n;i++)
        a[i] = a[i]*c;
}
main()
{
    int a[10];
    int n, c;

    printf("Unesite dimenziju niza (<=10): ");
    scanf("%d", &n);
```

```
    printf("Unesite koeficijente niza: ");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);

    printf("Unesite ceo broj kojim zelite da pomnozite niz: ");
    scanf("%d", &c);

    mnozi(a, n, c);

    printf("Elementi niza nakon mnozenja su: );
    for(i=0; i<n; i++)
        printf("%d ", a[i]);
}
```

2 Polinomi

1. Napisati program koji sa standardnog ulaza unosi prvo stepen (ceo broj manji od 10) pa zatim i celobrojne koeficijente polinoma (počevši od koeficijenta uz najniži stepen) a zatim štampa taj polinom u obliku: $P_n(x) = a_0 + a_1 * x^1 + \ldots + a_n * x^n$ pri čemu ne štampa članove sume koji su jednaki 0. Na primer, za unete vrednosti:

   ```
   2
   2 1 3
   ```

   Program treba da ispiše:

   ```
   P2(x) = 2 + 1*x^1 + 3*x^2
   ```

   A za unete vrednosti:

   ```
   2
   2 0 3
   ```

   Program treba da ispiše:

   ```
   P2(x) = 2 + 3*x^2
   ```

   ```
   #include <stdio.h>

   main()
   {
       int a[10];
       int n;
   ```

```
    printf("Unesite stepen polinoma:\n");
    scanf("%d", &n);

    printf("Unesite koeficijente polinoma pocevsi od koeficijenta uz
            najnizi stepen:\n);
    for(i=0; i<=n; i++)
        scanf("%d", &a[i]);

    /* Ispisivanje polinoma */
    printf("P%d(x)= ", n);
    printf("%d", a[0]);
    for(i=1; i<=n; i++)
        if (a[i])
           printf(" + %d*x^%d", a[i], i);
    printf("\n");
}
```

2. Napisati program koji sa standardnog ulaza unosi prvo stepen (ceo broj manji od 10) pa zatim i celobrojne koeficijente polinoma (počevši od koeficijenta uz najniži stepen) a zatim unosi i vrednost celobrojne promenljive x i štampa na standardni izlaz vrednost tog polinoma u toj tački.

   Na primer, za unete vrednosti

   ```
   2
   2 1 3
   1
   ```

   Program treba da izračuna vrednost polinoma:

   `P2(x) = 2 + 1*x^1 + 3*x^2`

   u tački $x = 1$ i da ispiše na standardni izlaz vrednost 6.

```
//Program izra\v cunava vrednost polinoma (prvi na\v cin).
#include <stdio.h>

int stepen(int, int);

main()
{
    int x,P=0,a[100];
    int n,i;
    printf("unesite stepen\n");
    scanf("%d",&n);
    for (i=0;i<=n;i++)
```

```
    {
        printf("unesite a[%d]\n",i);
        scanf("%d",&a[i]);
    }
    printf("unesite x\n");
    scanf("%d",&x);
    for (i=0;i<=n;i++)
        P=P+a[i]*stepen(x,i);
    printf("%d",P);
}

int stepen(int a ,int n)
{
    int i;
    int p=1;
    for (i=0;i < n;i++)
        p*=a;
    return p;
}

Ulaz:
unesite stepen
2
unesite a[0]
1
unesite a[1]
2
unesite a[2]
3
unesite x
4

Izlaz:
57.000000
```

3. Program izračunava vrednost polinoma (drugi način, efikasniji).

```
#include <stdio.h>
main()
{
    int n,i;
    int a[100],x,P;
    printf("Unesite najveci stepen: ");
    scanf("%d",&n);
    for (i=0;i<=n;i++)
    {
```

```
        printf("Unesite a[%d]: ",i);
        scanf("%d",&a[i]);
    }
    printf("Unesite x: ");
    scanf("%d",&x);
    P=0;
    for (i=n;i>=0;i--)
        P=P*x+a[i];

    printf("Vrednost polinoma je: %d\n",P);
}

Ulaz:
Unesite najveci stepen:
2
Unesite a[0]:
1
Unesite a[1]:
2
Unesite a[2]:
3
Unesite X: 4

Izlaz:
Vrednost polinoma je: 57.000000
```

3 Sume

1. Napisati program koji za uneti ceo broj n i realni broj x sa standardnog ulaza računa zbir
$$1 + x + \frac{x^2}{2} + \ldots + \frac{x^n}{n!}$$
Ispisati i sve članove sume.

   Napomena: program pozivati prvo za $x > 1$ pa onda za $x < 1$ da bi videli kako izgledaju faktori u jednom i u drugom slučaju.

```
#include<stdio.h>

main()
{
 float f, suma; /* Faktor sume i suma */
 float x;       /* Promenljiva x iz izraza */
 int i;         /* Brojac u petljama */
 int n;         /* Broj sabiraka */
```

```
 scanf("%d", n);
 scanf("%d", x);

 /* Pocetne inicijalizacije */
 f = 1;
 suma = 1;

 /* U jednom prolazu petlje dodajemo tekuci sabirak */
 for(i = 1; i <=n; i++)
 {
    f = f * x / i;
    suma = suma + f;
 }

 printf("Suma prvih %d clanova je %f \n", n, suma);
}
```

2. Napisati program koji računa zbir

$$1 + x + \frac{x^2}{2} + \ldots + \frac{x^n}{n!}$$

sa tačnošću $\varepsilon < 10^{-3}$, tj. računa zbir do prvog člana sume koji je manji od ε.

3. Napisati program koji za uneto n i x sa standardnog ulaza računa sumu

$$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^n * \frac{x^{2n+1}}{(2n+1)!}$$

Ako unesemo $x = 3.1415926$ dobićemo, za $n = 5$ vrednost 0.006925, a za $n = 10$ vrednost 0.00.

```
#include<stdio.h>

main()
{
 float f, suma, x;
 int i, n;

 scanf("%d", &n);
 scanf("%f", &x);

 /* Pocetne inicijalizacije */
 suma = x;
 f = x;
```

```
 for(i = 1; i <= n; i++) {
    f = -f * x * x / ((2*i+1)*2*i);
    suma = suma * f;
 }

 printf("Suma prvih %d clanova je %f \n", n, suma);
}
```

4. Napisati proram koji za uneto n i x sa standardnog ulaza računa sumu

$$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + (-1)^n \frac{x^{2n}}{(2n)!}$$

5. Program koji za uneto n i x sa standardnog ulaza računa sumu

$$x - \frac{x^3}{3 * 1!} + \frac{x^5}{5 * 2!} - \frac{x^7}{7 * 3!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1) * n!}$$

```
#include<stdio.h>

main()
{
 int i, n;
 float x, f, suma;

 scanf("%d", &n);
 scanf("%f", &x);

 /* Pocetne inicijalizacije */
 f = x;
 suma = x;

 for(i = 1; i <= n; i++) {
    f = -f * x * x / i;
    suma = suma + f/(2*i+1);
 }

 printf("Suma prvih %d clanoca je %f\n", n, suma);
}
```

4 Rekurzije

1. Napisati rekurzivnu funkciju za računanje faktorijela.

   Pre nego što krenete da pišete rekurzivnu funkciju najbolje bi bilo da pogledate kako izgleda rekurzivna definicija (matematički posmatrano) funkcije koju pišete.

Konktretno za faktorijal važi sledeća rekurzivna definicija:

$$f(n) = n! = \begin{cases} n * f(n-1) & \text{, za } n > 0 \\ 1 & \text{, za n=0} \end{cases}$$

Kada imate ovako zapisanu rekurzivnu definiciju lako dobijate rekurzivnu funkciju za izračunavanje faktorijala (tako što bukvalno prepišete definiciju u funkciju, naime, leve strane definicije idu u *return* naredbu, uslov da li je $n > 0$ ili $n = 0$ postavimo u *if* naredbi i dobijamo sledeću funkciju:

```
#include <stdio.h>

/* Rekurzivna verzija funkcije za racunanje faktorijala */
long f(int n)
{
      if (n == 0)
         return 1;
      else
         return n * f(n-1);
}

/* Iterativna verzija funkcije za racunjanje faktorijala. Zasniva se
   na iterativnoj definiciji faktorijala n! = 1*2*...*n */
long f_it(int n)
{
    long f = 1;
    int i;
    for (i = 1; i<=n; i++)
        f *= i;
    return f;
}

main()
{
      int n;

      printf("Unesite broj\n");
      scanf("%d", &n);
      printf("Faktorijal broja %d je %d\n", n, f(n));
}
```

2. Napisati rekurziju za računanje proizvoljnog stepena celog broja x. Naime, sa standardnog ulaza unosimo ceo broj x i pozitivan ceo broj n. Pozivom funkcije `stepen(x,n)` izračunati stepen broja x na broj n.

   Opet, prvo zapišemo funkciju koja nam treba na rekurzivan način. Neka je `stepen(x,n)`=x^n. Onda dobijamo sledeću definiciju:

$$stepen(x,n) = \begin{cases} x * stepen(x, n-1) & \text{, za } n > 0 \\ 1 & \text{, za n=0} \end{cases}$$

Na osnovu ove definicije, opet jednostavno dobijamo rekurzivnu funkciju za računanje stepena broja.

```
#include <stdio.h>

/* Rekurzivna funkcija */
int stepen(int x, int n)
{
    if (n == 0)
        return 1;
    else
        return x * stepen(x, n-1);
}

/* Iterativna funkcija koju racunamo po formuli da je
   x^n = x*x*...x (n puta) */
int stepen2(int x, int n)
{
     int i;
     int st = 1;
     for (i = 0; i< n; i++)
        st *= n;
     return st;
}

main()
{
      int x;
      int n;

      printf("Unesite broj i zeljeni stepen: \n");
      scanf("%d %d", &x, &n);
      printf("%d\n", stepen(x, n));
}
```

3. Ako zapišemo definiciju za računanje stepena broja na malo drugačiji način možemo dobiti malo efikasniju funkciju.

   Naime,

$$stepen(x,n) = \begin{cases} stepen(x, n/2) * stepen(x, n/2) & \text{, za parne } n > 0 \\ x * stepen(x, n/2) * stepen(x, n/2) & \text{, za neparne } n > 0 \\ 1 & \text{, za } n = 0 \end{cases}$$

I dobijamo sledeću funkciju (takođe rekurzivnu):

```
/* Rekurzivna funkcija po drugoj definiciji, efikasnija od prve */
int stepen1(int x, int n)
{
      if (n == 0)
        return 1;
      else
        {
            int st = stepen(x, n/2);
            if (n%2 == 0)
                return st*st;
            else
                return x*st*st;
        }
}
```

4. Napisati rekurzivnu funkciju za sumiranje niza brojeva `int suma(int a[], int n)` koja računa zbir prvih n elemenata niza a.

   Rekurzivna definicija ove funkcije može zapisati kao suma prvih $n-1$ članova na koju dodamo n-ti član (obratiti pažnju da je n-ti član niza član sa indeksom $n-1$).

$$suma(a,n) = \begin{cases} suma(a,n-1) + a_{n-1} & , n > 0 \\ 0 & , n = 0 \end{cases}$$

```
/* Rekurzivna verzija */
int suma(int a[], int n)
{
    if (n == 0)
        return 0;
    else
        return suma(a, n-1)+a[n-1];
}


/* Iterativna verzija, zasnovana na formuli
   suma_it(a, n) = a_0 + a_1 + ... a_{n-1} */
int suma_it(int a[], int n)
{
    int sum = 0;
    int i;
    for (i = 0; i<n;  i++)
        sum += a[i];
    return sum;
```

```
}

main()
{
    int a[] = {1, 2, 3, 4, 5, 6, 7};
    printf("sum = %d\n",  suma(a, sizeof(a)/sizeof(int)));
    printf("sum = %d\n",  suma_it(a, sizeof(a)/sizeof(int)));

}
```

5. Napisati funkciju koja računa n-ti Fibonačijev broj po formuli:

$$fib(n) = \begin{cases} 0 & , \text{za } n = 0 \\ 1 & , \text{za } n = 1 \\ fib(n-1) + fib(n-2) & , \text{za } n > 1 \end{cases}$$

```
#inlcude <stdio.h>

/* Rekurzivnu funkciju dobijamo direktno prepisivanjem rekurzivne
   definicije. Ako dodamo ispis broja za koji se funkcija poziva vidimo
   da je ovo neefikasno resenje */
int fib(int n)
{
printf("Pozivamo fib(%d)\n", n);
if (n == 0)
    return 0;
else if (n == 1)
    return 1;
else
    return fib(n-1) + fib(n-2);
}

/* Iterativna funkcija, efikasnija zato sto u svakom trenutku cuvamo samo
   tri promenljive i vrednost svakog fibonacijevog broja racunamo tacno
   jednom */
int fib_it(int n)
{
    int f0, f1, fn;

    //Pravimo pocetne inicijalizacije
    f0 = 0;
    f1 = 1;

    //U petlji racunamo sledeci fibonacijev broj
    for(int i = 0; i<= n-2; i++)
```

```
    {
       fn = f0 + f1;
       f0 = f1;
       f1 = fn;
    }

    return fn;
}

main()
{
      int n;

      printf("Unesite broj\n");
      scanf("%d", &n);

      printf("Rekurzivno: %d \n", fib(n));
      printf("Iterativno: %d \n", fib_it(n));
}
```

5 Argumenti komandne linije

1. Napisati program koji simulira korišćenje argumenata komandne linije.

```
/* Argumenti komandne linije programa */

/* 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("Broj argumenata komandne linije je: argc = %d\n", argc);

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

2. Napisati program koji od argumenata komandne linije ispisuje samo opcije koje su navedene u komandnoj liniji. Opcije prepoznajemo tako što su navedene iza znaka - (moguće je da iza jednog znaka - sledi više opcija).

```
/* Program ispisuje opcije navedene u komandnoj liniji. K&R resenje. */

/* 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 e f g  */

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

#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;
        }
}
```

6 Rad sa datotekama

1. Napisati program koji u datoteku sa imenom "podaci.txt" upisuje prvih 10 prirodnih brojeva i zatim iz te iste datoteke prepisuje brojeve na standardni izlaz.

```
/* Program demonstrira otvaranje datoteka ("r" - read i "w" - write mod)
   i osnovne tehnike rada sa datotekama */

#include <stdio.h>

/* Zbog funkcije exit */
#include <stdlib.h>

main()
{
        int i;

        /* Otvaramo datoteku sa imenom podaci.txt za pisanje */
        FILE* f = fopen("podaci.txt", "w");

        /* Ukoliko otvaranje nije uspelo, fopen vraca NULL. U tom slucaju,
           prijavljujemo gresku i zavrsavamo program */
        if (f == NULL)
        {
         printf("Greska prilikom otvaranja datoteke podaci.txt za pisanje\n");
         exit(1);
        }

        /* Upisujemo u datoteku prvih 10 prirodnih brojeva
           (svaki u posebnom redu) */
        for (i = 0; i<10; i++)
                fprintf(f, "%d\n", i);

        /* Zatvaramo datoteku */
        fclose(f);

        /* Otvaramo datoteku sa imenom podaci.txt za citanje */
        f = fopen("podaci.txt", "r");

        /* Ukoliko otvaranje nije uspelo, fopen vraca NULL. U tom slucaju,
           prijavljujemo gresku i zavrsavamo program */
        if (f == NULL)
        {
         printf("Greska prilikom otvaranja datoteke podaci.txt za citanje\n");
         exit(1);
```

```
        }

        /* Citamo brojeve iz datoteke dok ne stignemo do kraja i
           ispisujemo ih na standardni izlaz */
        while(1)
        {
                int br;
                /* Pokusavamo da procitamo broj */
                fscanf(f, "%d", &br);

                /* Ukoliko smo dosli do kraja datoteke, prekidamo */
                if (feof(f))
                        break;

                /* Ispisujemo procitani broj */
                printf("Procitano : %d\n", br);
        }

        /* Funkciju feof ne bi trebalo pozivati pre pokusaja citanja.
           Sledeci kod moze dovesti do greske:

                while (!feof(f))
                        fscanf(f,"%d",&br);

        */


        /* Zatvaramo datoteku */
        fclose(f);
}
```

2. Program demonstrira ”a” - append mod datoteka - nadovezivanje.

```
#include <stdio.h>

main()
{
        FILE* datoteka;

        /* Otvaramo datoteku za nadovezivanje i proveravamo da li je
           doslo do greske */
        if ( (datoteka=fopen("dat.txt","a"))==NULL)
        {
          fprintf(stderr,"Greska : nisam uspeo da otvorim dat.txt\n");
          return 1;
        }
```

```
        /* Upisujemo sadrzaj  u datoteku */
        fprintf(datoteka,"Zdravo svima\n");

        /* Zatvaramo datoteku */
        fclose(datoteka);
}
```

3. Napisati program koji kopira datoteku čije se ime zadaje kao prvi argument komandne linije u datoteku čije ime se zadaje kao drugi argument komandne linije (uz svaku prepisanu liniju dopisati i njen redni broj).

```
#include <stdio.h>

#define MAX_LINE 256

/* Funkcija getline koju smo ranije pisali moze se sada realizovati
   preko funkcije fgets */
int getline(char s[], int lim)
{
        char* c = fgets(s, lim, stdin);
        return c==NULL ? 0 : strlen(s);
}

main(int argc, char* argv[])
{
        char line[MAX_LINE];
        FILE *in, *out;
        int line_num;

        //Prvo proveravamo da li je dat dovoljan broj argumenata
        //komandne linije
        if (argc != 3)
        {
          fprintf(stderr,"Upotreba : %s ulazna_datoteka izlazna_datoteka\n",
                  argv[0]);
          return 1;
        }

        if ((in = fopen(argv[1],"r")) == NULL)
        {
          fprintf(stderr, "Neuspesno otvaranje datoteke %s\n", argv[1]);
          return 1;
        }

        if ((out = fopen(argv[2],"w")) == NULL)
```

```
        {
          fprintf(stderr, "Neuspesno otvaranje datoteke %s\n",argv[2]);
          return 1;
        }

        /*  Prepisivanje karakter po karakter je moguce ostvariti preko:
                int c;
                while ((c=fgetc(in)) != EOF)
                        putc(c,out);
        */

        line_num = 1;
        /* Citamo liniju po liniju sa ulaza*/
        while (fgets(line, MAX_LINE, in) != NULL)
        {
          /* Ispisujemo broj linije i sadrzaj linije na izlaz */
          fprintf(out, "%-3d :\t", line_num++);
          fputs(line, out);
        }

        /* Zatvaramo datoteke */
        fclose(in);
        fclose(out);
}
```