Programiranje 2

Beleške sa vežbi

Školska 2006/2007 godina

Matematički fakultet, Beograd

Jelena Tomašević

March 13, 2007

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] Zasnovano na primerima sa sajta http://www.matf.bg.ac.yu/∼filip

```
/* 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 *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 Polinomi

Primer 5 *Program ilustruje rad sa polinomima.*

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

#define max(a, b) ((a) > (b) ? (a) : (b))

/*
*   Polinom 3*x^3 + 2*x + 1.5 se predstavlja kao:
*   stepen -> 3
```

```
*   koeficijenti -> 1.5 2 0 3 x x x x x ... x
*/
typedef struct polinom {
    float koeficijenti[21];
    int stepen;
} Polinom;

/*
*   Funkcija vraca koeficijent uz x^i u polinomu p
*   Zbog efikasnosti ne prenosimo celu strukturu vec
*   samo pokazivac na strukturu.
*/
float vratiKoeficijent(Polinom* p, int i) {
    return i <= p->stepen ? p->koeficijenti[i] : 0.0f;
}

/*
*   Funkcija postavlja koeficijent uz x^i u polinomu p na
*   dati koeficijent k
*/
void postaviKoeficijent(Polinom* p, int i, float k)
{
    int j;
    /*
       Ukoliko je stepen polinoma bio manji, postavljamo
           sve koeficijente izmedju na 0.0 i uvecavamo stepen
        */
    if (i > p->stepen) {
        for (j = p->stepen+1; j < i; j++)
            p->koeficijenti[j] = 0.0f;
        p->stepen = i;
    }

    p->koeficijenti[i] = k;
}

/*
*    Funkcija kreira polinom datog stepena sa datim nizom
*    koeficijenata. Pretpostavlja se da su koeficijenti
*    u datom nizu koeficijenti[] uredjeni opadajuce po stepenima
*    polinoma.
*/
Polinom napraviPolinom(int stepen, float koeficijenti[]) {
    int i;
    Polinom p;
    p.stepen = stepen;
    for (i = 0; i <= stepen; i++) {
        postaviKoeficijent(&p, i, koeficijenti[stepen - i]);
    }
    return p;
}
```

```
/*
*   Funkcija ispisuje polinom u citljivijem obliku.
*   Npr: 3.0*x^3 + 0.0*x^2 + 2.0*x^1 + 1.5*x^0
*/
void ispisiPolinom(Polinom* p) {
    int i;
    for (i = p->stepen; i >= 0; i--) {
        printf("%.2f*x^%d", vratiKoeficijent(p, i), i);
        if (i > 0)
            printf(" + ");
    }
    printf("\n");
}

/*
*   Funkcija izracunava vrednost polinoma u tacki x
*   Hornerovom shemom. Npr.
*   3*x^3 + 2*x + 1.5   =
*   (((0*x + 3)*x + 0)*x + 2)*x + 1.5
*   Postupak izracunavanja p(10):
*   0.0
*   10 * 0.0 + 3 = 3.0
*   10 * 3.0 + 0 = 30.0
*   10 * 30.0 + 2 = 302.0
*   10 * 302.0 + 1.5 = 3021.5
*/
float vrednost(Polinom* p, float x) {
    int i;
    float suma = 0.0f;
    for (i = p->stepen; i >= 0; i--) {
        suma = suma*x + vratiKoeficijent(p, i);
    }
    return suma;
}

/*
*   Funkcija sabira dva polinoma
*/
Polinom saberi(Polinom* p, Polinom* q) {
    int i;
    Polinom zbir;
    zbir.stepen = max(p->stepen, q->stepen);
    for (i = 0; i <= zbir.stepen; i++)
        postaviKoeficijent(&zbir, i,
            vratiKoeficijent(p, i) + vratiKoeficijent(q, i));
    return zbir;
}

/*
*   Funkcija mnozi dva polinoma. Npr.
*
*   1*x^2 + 2*x + 3
```

```
*   4*x + 7
*
*   0.0*x^3 + 0.0*x^2 + 0.0*x + 0.0
*   0.0*x^3 + 0.0*x^2 + 0.0*x + 21.0    i=0 j=0
*   0.0*x^3 + 0.0*x^2 + 12.0*x + 21.0   i=0 j=1
*   0.0*x^3 + 0.0*x^2 + 26.0*x + 21.0   i=1 j=0
*   0.0*x^3 + 8.0*x^2 + 26.0*x + 21.0   i=1 j=1
*   0.0*x^3 + 15.0*x^2 + 26.0*x + 21.0  i=2 j=0
*   4.0*x^3 + 15.0*x^2 + 26.0*x + 21.0  i=2 j=1
*/

Polinom pomnozi(Polinom* p, Polinom* q) {
    int i, j;
    Polinom proizvod;
    proizvod.stepen = p->stepen + q->stepen;
    for (i = 0; i <= proizvod.stepen; i++) {
        postaviKoeficijent(&proizvod, i, 0.0f);
    }

    for (i = 0; i <= p->stepen; i++) {
        for (j = 0; j <= q->stepen; j++) {
            /* r[i+j] = r[i+j] + p[i]*q[j] */
            postaviKoeficijent(&proizvod, i+j,
                vratiKoeficijent(&proizvod, i+j) +
                vratiKoeficijent(p, i) *
                vratiKoeficijent(q, j));
        }
    }
    return proizvod;
}

main() {
    float pkoeficijenti[] = {1.0f, 2.0f, 1.0f};
    Polinom p = napraviPolinom(2, pkoeficijenti);
    float qkoeficijenti[] = {1.0f, 1.0f};
    Polinom q = napraviPolinom(1, qkoeficijenti);
    Polinom r = pomnozi(&p, &q);
    ispisiPolinom(&r);
}
```

1.3 Rad sa velikim celim brojevima

Primer 6 *Program ilustruje rad sa velikim celim brojevima. Brojevi se interno reprezentuju preko niza svojih cifara.*

```
#include <stdio.h>
#include <ctype.h>
#define MAX_CIFRE 100

/* Funkcija obrce cifre prosledjenog niza */
void obrni_cifre (int cifre[], int duzina)
{
```

```
    int i, j;
    for (i=0, j=duzina-1; i<j; i++, j--)
    {
        int pom = cifre[i];
        cifre[i] = cifre[j];
        cifre[j] = pom;
    }
}

/* Funkcija sa standardnog ulaza ucitava niz cifara, duzine
   najvise max_cifre i smesta ga u niz brojeva cifre[]. Zatim
   se niz cifre[] obrce kako bi cifre najmanje tezine bile
   na pocetku niza.
   Kao rezultat, funkcija vraca broj cifara ucitanog broja */
int uzmi_broj (int cifre[], int max_cifre)
{
    int duzina = 0;
    char c;

    while ( --max_cifre>0 && isdigit(c=getchar()))
        cifre[duzina++]=c-'0';

    obrni_cifre(cifre, duzina);

    return duzina;
}


/* Funkcija ispisuje "veliki" broj predstavljen
   nizom cifara cifre, duzine duzina, na standardni
   izlaz imajuci u vidu da su cifre u nizu zapisane
   "naopako" tj. pocevsi od cifre najmanje tezine */
void ispisi_broj(int cifre[],int duzina)
{
    int i;
    for (i=duzina-1; i>=0; i--)
        printf("%d",cifre[i]);
    putchar('\n');
}


/* Da li su dva broja data svojim nizovima cifara i duzinama jednaka?
   Funkcija vraca 1 ako jesu, a 0 ako nisu */
int jednaki(int a[], int duzina_a, int b[], int duzina_b)
{
    int i;

    /* Poredimo duzine */
    if (duzina_a != duzina_b)
        return 0;

    /* Ako su brojevi iste duzine, poredimo cifru po cifru
```

```
       pocevsi od pozicije najvece tezine */
    for (i=0; i<duzina_a; i++)
        if (a[i] != b[i])
            return 0;

    return 1;
}


/* Funckija poredi dva broja a i b, data svojim nizovima cifara i duzinama
   i vraca:
      1  ako je a>b
      0  ako je a=b
      -1 ako je b>a
*/
int uporedi(int a[], int duzina_a, int b[], int duzina_b)
{
    int i;
    /* Uporedjujemo duzine brojeva a i b */
    if (duzina_a > duzina_b)
        return 1;
    if (duzina_a < duzina_b)
        return -1;

    /* U ovom trenutku znamo da su brojevi iste duzine, tako da
       prelazimo na poredjenje cifre po cifre, pocevsi od cifre
       najvece tezine */
    for (i=duzina_a-1; i>=0; i--)
    {
        if (a[i] > b[i])
            return 1;
        if (a[i] < b[i])
            return -1;
    }

    return 0;
}

/* Funkcija sabira dva broja data svojim nizovima
   cifara i duzinama i rezultat ispisuje na ekran
 */
void saberi(int a[], int duzina_a, int b[], int duzina_b)
{
    /* Rezultat, zadan svojim nizom cifara i duzinom */
    int rezultat[MAX_CIFRE];
    int duzina_rezultata;
    int i;
    /* Prenos sa prethodne pozicije */
    int prenos = 0;

    /* Sabiranje vrsimo dok ne prodjemo sve cifre duzeg od brojeva a i b */
    for(i=0; i<duzina_a || i<duzina_b; i++)
```

```
    {
        /* Sabiramo i-tu cifra broja a (ako postoji) sa i-tom cifrom
           broja b (ako postoji) i prenos sa prethodne pozicije */
        int cifra_rezulata=((i<duzina_a)? a[i] : 0) +
                    ((i<duzina_b)? b[i] : 0) +
                    prenos;

            /* Nova cifra rezultata */
        rezultat[i] = cifra_rezulata%10;

        /* Prenos na sledecu poziciju */
            prenos = cifra_rezulata/10;
    }

    /* Kada smo zavrsili sa svim ciframa brojeva a i b moguce je da je
    postojao prenos na sledecu poziciju u kom slucaju uvecavamo duzinu
    rezultata. Inace duzinu rezultata postavljamo na duzinu duzeg od
    brojeva a i b, koja se nalazi trenutno u  promenjivoj i */
    if (prenos != 0)
    {
    if (i>=MAX_CIFRE)
        printf("Doslo je do prekoracenja!\n");
    else
    {
        rezultat[i] = prenos;
        duzina_rezultata = i+1;
       }
    }
    else
    duzina_rezultata=i;

    /* Ispisujemo rezultat */
    ispisi_broj(rezultat, duzina_rezultata);
}

/* Funkcija mnozi broj, dat svojim nizom cifara i duzinom, datom cifrom c
   i rezultat ispisuje */
void pomnozi_cifrom (int a[], int duzina_a, int cifra)
{
    /* Rezultat, zadan svojim nizom cifara i duzinom */
    int rezultat[MAX_CIFRE];
    int duzina_rezultat;

    int i, prenos = 0;
    for (i=0; i<duzina_a; i++)
    {
        /* Svaku cifru broja a mnozimo cifrom c, dodajemo na to
           prenos sa prethodne pozicije i to smestamo u promenljivu
           pom */
        int pom  = a[i]*cifra + prenos;

        /* Nova cifra rezultata */
```

```
        rezultat[i] = pom%10;
        /* Prenos na sledecu poziciju */
        prenos = pom/10;
    }

    /* Kada smo zavrsili sa svim ciframa broja a, moguce je da je
       postojao prenos na sledecu poziciju u kom slucaju uvecavamo
       duzinu rezultata. Inace duzinu rezultata postavljamo na duzinu
       broja a, koja se nalazi trenutno u promenjivoj i */
    if (prenos != 0)
    {
        if (i>=MAX_CIFRE)
            printf("Doslo je do prekoracenja !\n");
        else
        {
            rezultat[i] = prenos;
            duzina_rezultata = duzina_a+1;
        }
    }
    else
        duzina_rezultata = duzina_a;

   /* Ispisujemo rezultat */
   ispisi_broj(rezultat, duzina_rezultata);
}


/* Funkcija mnozi dva broja data svojim nizovima cifara i duzinama i proizvod
   ispisuje na standardni izlaz */
void pomnozi (int a[], int duzina_a, int b[], int duzina_b)
{
    /* Ova funkcija se gradi kombinovanjem algoritama mnozenja broja cifrom i
       sabiranja dva broja */

    int i,j,k;

    /* Broj pom ce da sadrzi rezultat mnozenja broja a jednom po jednom cifrom broja b,
       Dok ce na broj rezultat da se dodaje svaki put (10^i)*pom */
    int pom[MAX_CIFRE], duzina_pom;
    int rezultat[MAX_CIFRE], duzina_rezultata;


    duzina_rezultata = 0;

    /* Za svaku cifru broja a */
    for (i=0; i<duzina_a; i++)
    {
        /* vrsimo mnozenje broja b i-tom cifrom broja a */
        int prenos = 0;
        for (j=0; j<duzina_b; j++)
        {
            int pm = b[j]*a[i] + prenos;
```

```
            pom[j] = pm%10;
            prenos = pm/10;
        }

        if (prenos)
        {
            pom[j] = prenos;
            duzina_pom = j+1;
            }
        else
            duzina_pom = j;


            /* Zatim dodajemo broj pom na rezultat, ali dodavanje pocinjemo od i-te cifre rezultata.
        prenos = 0;
        for (k=0; k<duzina_pom || i+k<duzina_rezultata; k++)
            {
            int pm=((i+k<duzina_rezultata) ? rezultat[i+k] : 0) + pom[k] + prenos;
            rezultat[i+k] = pm%10;
            prenos = pm/10;
        }

            if (prenos)
        {
            rezultat[i+k] = prenos;
            duzina_rezultata = i+k+1;
            }
        else
            duzina_rezultata = i+k;

    }

    /* Ispisujemo rezultat */
    ispisi_broj(rezultat, duzina_rezultata);

}


/* Primer koriscenja funkcija */
int main()
{
    /* duzina brojeva a i b */
    int duzina_a, duzina_b;

    /* nizovi cifara brojeva a i b */
    int a[MAX_CIFRE], b[MAX_CIFRE];

    /* Ucitavaju se brojevi */
    printf("Unesite prvi broj : ");
        duzina_a = uzmi_broj(a,MAX_CIFRE);
        printf("Unesite drugi broj : ");
    duzina_b = uzmi_broj(b, MAX_CIFRE);
```

```
    /* Sabiraju se i ispisuje se zbir */
    printf("Zbir je : ");
    saberi(a, duzina_a, b, duzina_b);


    /* Mnoze se i ispisuje se proizvod */
    printf("Proizvod je : ");
        pomnozi(a, duzina_a, b, duzina_b);

    return 0;
}
```

Zadaci za vežbu:

Zadatak 1 *Ilustracija rada sa argumentima komandne linije (korišćenje naredbe switch u programu).*

Zadatak 2 *(Ispitni zadatak, februar 2007.) Sastaviti program koji ispisuje prvih 100 elemenata Fibonačijevog niza zadatog sledećim formulama*

$f_1 = 1$, $f_2 = 2$, $f_n = f_{n-1} + f_{n-2} (n > 2)$

tačno u svakoj cifri. (Napomena: ako je f_n tipa unsigned long moguće je ispisati prva 43 elementa niza.)

Zadatak 3 *Napisati funkciju koja izračunava faktorijel broja tačno u svakoj cifri.*