Programiranje 1

Beleške sa vežbi

Smer *Informatika*
Matematički fakultet, Beograd

Jelena Tomašević

Sadržaj

1

[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.*

```
/* 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  */
```

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

```
/* 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 Pokazivači - osnovni pojmovi

Primer 6 *Ilustracija rada sa pokazivačkim promenljivim.*

```
#include <stdio.h>
main()
{
    int x = 3;

    /* Adresu promenjive x zapamticemo u novoj promeljivoj.
       Nova promenljiva je tipa pokazivaca na int (int*) */
    int* px;

    printf("Adresa promenljive x je : %p\n", &x);
    printf("Vrednost promenljive x je : %d\n", x);


    px = &x;
    printf("Vrednost promenljive px je (tj. px) : %p\n", px);
    printf("Vrednost promenljive na koju ukazuje px (tj. *px) je : %d\n", *px);

    /* Menjamo vrednost promenljive na koju ukazuje px */
    *px = 6;
    printf("Vrednost promenljive na koju ukazuje px (tj. *px) je : %d\n", *px);

    /* Posto px sadrzi adresu promenljive x, ona ukazuje na x tako da je
       posredno promenjena i vrednost promenljive x */
    printf("Vrednost promenljive x je : %d\n", x);


}

Izlaz (u konkretnom slucaju):
Adresa promenljive x je : 0012FF88
Vrednost promenljive x je : 3
Vrednost promenljive px je (tj. px) : 0012FF88
Vrednost promenljive na koju ukazuje px (tj. *px) je : 3
Vrednost promenljive na koju ukazuje px (tj. *px) je : 6
```

```
Vrednost promenljive x je : 6
```

Primer 7 *swap : Demonstracija prenosa argumenata preko pokazivača.*

```
#include <stdio.h>

/* Pogresna verzija funkcije swap. Zbog prenosa po vrednosti, funkcija
   razmenjuje kopije promenljivih iz main-a, a ne samih promenljivih */
void swap_wrong(int x, int y)
{
    int tmp;

    printf("swap_wrong: ");
    printf("Funkcija menja vrednosti promenljivim na adresama : \n");
    printf("x : %p\n", &x);
    printf("y : %p\n", &y);


    tmp = x;
    x = y;
    y = tmp;
}

/* Resenje je prenos argumenata preko pokazivaca */
void swap(int* px, int* py)
{
    int tmp;

    printf("swap : Funkcija menja vrednosti promenljivim na adresama : \n");
    printf("px = %p\n", px);
    printf("py = %p\n", py);

    tmp = *px;
    *px = *py;
    *py = tmp;
}

main()
{
    int x = 3, y = 5;
    printf("Adresa promenljive x je %p\n", &x);
    printf("Vrednost promenljive x je %d\n", x);
    printf("Adresa promenljive y je %p\n", &y);
    printf("Vrednost promenljive y je %d\n", y);

    /* Pokusavamo zamenu koristeci pogresnu verziju funkcije */
    swap_wrong(x, y);

    printf("Posle swap_wrong:\n");
    printf("Vrednost promenljive x je %d\n", x);
    printf("Vrednost promenljive y je %d\n", y);

    /* Vrsimo ispravnu zamenu. Funkciji swap saljemo adrese promenljvih
```

```
       x i y, a ne njihove vrednosti */
    swap(&x, &y);

    printf("Posle swap:\n");
    printf("Vrednost promenljive x je %d\n", x);
    printf("Vrednost promenljive y je %d\n", y);
}

Izlaz u konkretnom slucaju:
Adresa promenljive x je 0012FF88
Vrednost promenljive x je 3
Adresa promenljive y je 0012FF84
Vrednost promenljive y je 5
swap_wrong: Funkcija menja vrednosti promenljivim na adresama :
x : 0012FF78
y : 0012FF7C
Posle swap_wrong:
Vrednost promenljive x je 3
Vrednost promenljive y je 5
swap : Funkcija menja vrednosti promenljivim na adresama :
px = 0012FF88
py = 0012FF84
Posle swap:
Vrednost promenljive x je 5
Vrednost promenljive y je 3
```

Primer 8 *Demonstracija više povratnih vrednosti funkcije koristeći prenos preko pokazivača.*

```
/* Funkcija istovremeno vraca dve vrednosti - kolicnik i ostatak dva data broja.
   Ovo se postize tako sto se funkciji predaju vrednosti dva broja (x i y) koji se dele
   i adrese dve promenljive na koje ce se smestiti rezultati */
void div_and_mod(int x, int y, int* div, int* mod)
{
    printf("Kolicnik postavljam na adresu : %p\n", div);
    printf("Ostatak postavljam na adresu : %p\n", mod);
    *div = x / y;
    *mod = x % y;
}

main()
{
    int div, mod;
    printf("Adresa promenljive div je %p\n", &div);
    printf("Adresa promenljive mod je %p\n", &mod);

    /* Pozivamo funkciju tako sto joj saljemo vrednosti dva broja (5 i 2)
       i adrese promenljvih div i mod na koje ce se postaviti rezultati */
    div_and_mod(5, 2, &div, &mod);

    printf("Vrednost promenljive div je %d\n", div);
    printf("Vrednost promenljive mod je %d\n", mod);

}
```

```
Izlaz u konkretnom slucaju:
Adresa promenljive div je 0012FF88
Adresa promenljive mod je 0012FF84
Kolicnik postavljam na adresu : 0012FF88
Ostatak postavljam na adresu : 0012FF84
Vrednost promenljive div je 2
Vrednost promenljive mod je 1
```

1.3 Strukture - uvežbavanje

Primer 9 *Strukture se u funkcije prenose po vrednosti. Moguće je koristiti pokazivače na strukture.*

```
#include <stdio.h>

typedef struct point
{
    int x, y;
} POINT;

/* Zbog prenosa po vrednosti tacka ne moze biti ucitana */
void get_point_wrong(POINT p)
{
    printf("x = ");
    scanf("%d", &p.x);
    printf("y = ");
    scanf("%d", &p.y);
}

/* Koriscenjem prenosa preko pokazivaca, uspevamo */
void get_point(POINT* p)
{
    /* p->x je skraceni zapis za (*p).x */

    printf("x = ");
    scanf("%d", &p->x);
    printf("y = ");
    scanf("%d", &p->y);
}


main()
{
    POINT a = {0, 0};

    printf("get_point_wrong\n");
    get_point_wrong(a);
    printf("a: x = %d, y = %d\n", a.x, a.y);

    printf("get_point\n");
    get_point(&a);
```

```
    printf("a: x = %d, y = %d\n", a.x, a.y);

}
```

Primer 10 *Datoteka cije se ime unosi sa komandne linije sadrži podatke o studentima (ime, prezime, broj indeksa). Podaci su korektno zadati. Nema više od 1000 studenata. Prvo formirati niz struktura u memoriji, a onda ih ispisiati.*

```
#include <stdio.h>

#define MAXL 100
#define MAXN 1000

typedef struct student {
    char ime[MAXL];
    char prezime[MAXL];
    short int indeks;
} student;

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

    FILE *in; int j,i=0;
    student niz[MAXN];

    if(argc!=2)
    {
        fprintf(stderr,"Neispravno pozivanje! Koriscenje: %s <ime datoteke>\n",argv[0]);
        return -1;
    }
    if(!(in=fopen(argv[1],"r")))
    {
        fprintf(stderr,"Ne mogu da otvorim datoteku %s za citanje.",argv[1]);
        return -1;
    }

    while(!feof(in))
    {
        fscanf(in,"%s %s %d",&niz[i].ime, &niz[i].prezime, &niz[i].indeks);
        i++;
    }

    /* Sredjujemo zadnji scanf koji ucitava EOF */
    i--;

    for(j=0;j<i;j++)
    {
        fprintf(stdout,"Ime: %s\nPrezime: %s\nIndeks: %d\n\n",
                        niz[j].ime, niz[j].prezime, niz[j].indeks);
    }

    fclose(in);
    return 0;
```

```
}
```

Primer 11 *Sa standardnog ulaza se učitava niz od* `n (n<100)` *tačaka u ravni takvih da nikoje tri tačke nisu kolinearne. Tačke se zadaju parom svojih koordinata (celi brojevi). Ispitati da li taj niz tačaka odredjuje konveksni mnogougao i rezultat ispisati na standardni izlaz.*

```
#include<stdio.h>

typedef struct tacka
{
int x;
int y;
} TACKA;

/* F-ja ispituje da li se tacke T3 i T4 nalaze sa iste strane prave
 odredjene tackama T1 i T2.*/
int SaIsteStranePrave(TACKA T1,TACKA T2, TACKA T3, TACKA T4)
{
    int t3 = (T3.y - T1.y)*(T2.x - T1.x) - (T2.y - T1.y) * (T3.x - T1.x);
    int t4 = (T4.y - T1.y)*(T2.x - T1.x) - (T2.y - T1.y) * (T4.x - T1.x);
    return (t3 * t4 > 0);
}

main()
{
    TACKA mnogougao[100];
    int j,i;
    int n;
    int konveksan = 1;

    do
    {
        printf("Unesite broj temena mnogougla:\n");
        scanf("%d",&n);
        if(n<3)
            printf("Greska! Suvise malo tacaka! Pokusajte ponovo!\n");
    }
    while(n<3);

    printf("Unesite koordinate temena mnogougla takve da nikoja tri
                                        temena nisu kolinearna!\n");
    for(i=0;i<n;i++)
    scanf("%d %d", &mnogougao[i].x, &mnogougao[i].y);

    /* Da bi mnogougao bio konveksan potrebno (i dovoljno) je da kada se
    povuce prava kroz bilo koja dva susedna temena mnogougla sva ostala
    temena budu sa iste strane te prave.*/
    for(i=0;konveksan&&i<n-1;i++)
    {
        for(j=0;konveksan&&j<i-1;j++)
            konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
                        mnogougao[i+1],mnogougao[j],mnogougao[j+1]);
        for(j=i+2;konveksan&&j<n-1;j++)
```

```
            konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
                        mnogougao[i+1],mnogougao[j],mnogougao[j+1]);
        if(i!=0&&i!=n-1&&i+1!=0&&i+1!=n-1)
            konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
                        mnogougao[i+1],mnogougao[0],mnogougao[n-1]);
    }
    for(j=1;konveksan&&j<n-2;j++)
        konveksan=konveksan && SaIsteStranePrave(mnogougao[0],
                        mnogougao[n-1],mnogougao[j],mnogougao[j+1]);

    if(konveksan)
        printf("Uneti mnogougao jeste konveksan!\n");
    else
        printf("Uneti mnogougao nije konveksan!\n");

}
```