Programiranje I

Beleške sa vežbi

Smer *Informatika*
Matematički fakultet, Beograd

Sana Stojanović

Sadržaj

1 Zadaci za vežbanje

1.
```
/* Program pronalazi najduzu liniju sa ulaza. */

#include <stdio.h>
#define MAX_DUZINA_LINIJE 80

int ucitaj_liniju(char linija[])
{
    int c;
    int pozicija = 0;

    while ((c=getchar())!=EOF && c != '\n' && pozicija<MAX_DUZINA_LINIJE-1)
        linija[pozicija++]=c;

    if (c=='\n')
        linija[pozicija++] = c;

    linija[pozicija]=0;

    return pozicija;
}

void kopiraj_liniju(char u[], char iz[])
{
    int pozicija;
    pozicija = 0;

    do
    {
    u[pozicija] = iz[pozicija];
    pozicija++;
    } while (u[pozicija-1]!=0);
}

main()
{
    char linija[MAX_DUZINA_LINIJE];
    char najduza_linija[MAX_DUZINA_LINIJE];
    int duzina_linije = 0;
    int duzina_najduze_linije = 0;
    najduza_linija[0] = 0;

    while( (duzina_linije = ucitaj_liniju(linija)) != 0)
    {
        if (duzina_linije>duzina_najduze_linije)
```

```
        {
            kopiraj_liniju(najduza_linija, linija);
            duzina_najduze_linije = duzina_linije;
        }
    }
    printf("Najduza linija je : %s\n",najduza_linija);
}
```

2. /* Program pronalazi najduzu liniju sa ulaza(II verzija).*/

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

#define MAX_DUZINA_LINIJE 80

int ucitaj_liniju(char linija[])
{
    int c;
    int pozicija = 0;
    while ((c=getchar())!=EOF && c != '\n' && pozicija<MAX_DUZINA_LINIJE-1)
        linija[pozicija++]=c;

    if (c=='\n')
        linija[pozicija++] = c;

    linija[pozicija]=0;

    return pozicija;
}

main()
{
    char linija[MAX_DUZINA_LINIJE];
    char najduza_linija[MAX_DUZINA_LINIJE];

    int duzina_linije = 0;
    int duzina_najduze_linije = 0;

    najduza_linija[0] = 0;

    while( (duzina_linije = ucitaj_liniju(linija)) != 0)
    {
        if (duzina_linije>duzina_najduze_linije)
        {
            strcpy(najduza_linija, linija);
```

```
            duzina_najduze_linije = duzina_linije;
        }
    }
    printf("Najduza linija je : %s\n",najduza_linija);
}
```

2 Define

1.
```
/* Demonstracija pretprocesorske direktive #define */

#include<stdio.h>
/* Racuna sumu dva broja */
#define sum(a,b) ((a)+(b))

/* Racuna kvadrat broja - pogresna verzija */
#define square_w(a) a*a

/* Racuna kvadrat broja */
#define square(a) ((a)*(a))

/* Racuna minimum tri broja */
#define min(a, b, c) (a)<(b) ? ((a)<(c) ? (a) : (c)) : ((b)<(c) ? (b) : (c))

main()
{
    printf("sum(3,5) = %d\n", sum(3,5));
    printf("square_w(5) = %d\n", square_w(5));
    printf("square_w(3+2) = %d\n", square_w(3+2));
    printf("square(3+2) = %d\n", square(3+2));
    printf("min(1,2,3) = %d\n", min(1,2,3));
    printf("min(1,3,2) = %d\n", min(1,3,2));
    printf("min(2,1,3) = %d\n", min(2,1,3));
    printf("min(2,3,1) = %d\n", min(2,3,1));
    printf("min(3,1,2) = %d\n", min(3,1,2));
    printf("min(3,2,1) = %d\n", min(3,2,1));
}

Izlaz iz programa:
sum(3,5) = 8
square_w(5) = 25
square_w(3+2) = 11
square(3+2) = 25
min(1,2,3) = 1
min(1,3,2) = 1
min(2,1,3) = 1
```

```
min(2,3,1) = 1
min(3,1,2) = 1
min(3,2,1) = 1
```

2.
```
/* Demonstracija pretprocesorske direktive #define */

#include <stdio.h>
#define KUBW(a) (a * a * a)
#define KUB(a) ( (a) * (a) * (a))

main()
{
int b=1;

printf("KUBW(%d) = %d\n", 2*b+4, KUBW(2*b+4) );
printf("KUB(%d) = %d\n", 2*b+4, KUB(2*b+4) );
}

Izlaz:
KUBW(6) = 22
KUB(6) = 216
```

3.
```
/* Ilustacija beskonacne petlje */

#define forever for(;;)
```

4.
```
/* Moguce je definisati makroe sa argumentima tako da tekst zamene bude
razlicit za razlicita pojavljivanja makroa. */

#define max(A, B) ((A)>(B) ? (A) : (B))

/*
Na osnovu ovoga ce linija

x=max(p+q, r+s)

biti zamenjena linijom

x=((p+q) > (r+s) ? (p+q) : (r+s));
*/

/*
Treba voditi racuna o sporednim efektima. Sledeca linija koda pruzrokovace
uvecanje vrednosti i i j za dva.

max(i++, j++)
*/
```

```
/*
Takodje treba voditi racuna o zagradama. Sledeci makro prouzrokovace
neocekivane rezultate za poziv square(a+1)

#define square(x) x*x
*/
```

5.
```
#include <stdio.h>
#define max1(x,y) (x>y?x:y)
#define max2(x,y) ((x)>(y)?(x):(y))
#define swapint(x,y) { int z; z=x; x=y; y=z; }
#define swap(t,x,y) { \
    t z; \
    z=x; \
    x=y; \
    y=z; }

main()
{

int x=2,y=3;

printf( "max1(x,y) = %d\n", max1(x,y) );
/* max1(x,y) = 3 */

/* Zamena makroom se ne vrsi
   unutar niski pod navodnicima*/
printf( "max1(x=5,y) = %d\n", max1(x,y) );
/* max1(x=5,y) = 3 */

printf( "max1(x++,y++) = %d\n", max1(x++,y++) );
/* max1(x++,y++) = 4 */

printf( "x = %d, y = %d\n", x, y );
/* x = 3, y = 5 */

swapint(x,y);
printf( "x = %d, y = %d\n", x, y );
/* x = 5, y = 3 */
swap(int,x,y);
printf( "x = %d, y = %d\n", x, y );
/* x = 3, y = 5 */
}

Izlaz:
```

```
max1(x,y) = 3
max1(x=5,y) = 3
max1(x++,y++) = 4
x = 3, y = 5
x = 5, y = 3
x = 3, y = 5
```

3 Pokazivači

1.
```
/* Demonstracija vise povratnih vrednosti funkcije koristeci prenos preko
pokazivaca.*/
/* 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
```

4 Rad sa datotekama

1. /* Program koji sa standardnog ulaza ucitava pozitivan ceo broj, a na standardni izlaz ispisuje vrednost tog broja sa razmenjenim vrednostima bitova na poziciji i, j. Pozicije i, j se ucitavaju kao parametri komandne linije. Smatrati da krajnji desni bit binarne reprezentacije je 0-ti bit. Pri resavanju nije dozvoljeno koristiti pomocni niz niti aritmeticke 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;
}
```

2. /* Iz datoteke cije se ime zadaje kao argrument komandne linije, ucitati cele brojeve sve dok se ne ucita nula, i njihov zbir upisati u datoteku cije se ime takodje 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 koju datoteku treba upisati 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;
}
```

5 Formiranje HTML dokumenta

1. ```
   /* Program koji formira html dokument.
      Prilikom pokretanja programa koristiti redirekciju:
   ```
   ```

```
   a.out >primer.html

   kako bi se rezultat rada programa upisao u datoteku primer.html.
*/

#include <stdio.h>
main()
{
printf("<html><head><title>Ova stranica je napravljena u c-u
        </title></head>");
printf("<body><h3 align=center> Rezultat </h3></body></html>");
}
```

2.
```
/* Program koji generise html dokument sa engleskim alfabetom. */

#include <stdio.h>
main()
{
    int i;
    printf("<HTML><head><title>Engleski alfabet</title><head>\n");
    printf("<body><ul>");
    for(i=0;i<=25;i++)
        printf("<li> %c %c \n",'A'+i,'a'+i);
    printf("</ul></body></HTML>\n");
}
```

3.
```
/* Napisati program koji generise html dokument koji prikazuje tablicu
mnozenja za brojeve od 1 do 10. */

#include<stdio.h>
main()
{
    int i,j;
    printf("<html><head><title>Mnozenje</title></head>");
    printf("<body><h3 align=center> Rezultat </h3>");
    printf("<table border=1>\n");

    /* Prva vrsta sadrzi brojeve od 1 do 10*/
    printf("<tr>");
    printf("<th></th>");
    for(i=1; i<=10; i++)
        printf("<th> %d </th>\n", i);
    printf("</tr>");

    for(i=1; i<=10; i++)
```

```
    {
        printf("<tr>");

        /* Na pocetku svake vrste stampamo broj
        odgovarajuce vrste*/
        printf("<th>%d</th>", i);

        for(j=1; j<=10; j++)
            printf("<td>%d\t</td>\n", i*j);

        printf("</tr>");
    }
    printf("</table>");
    printf("</body></html>");
    return 0;
}
```

4. ```
   /* Program ilustruje citanje etiketa iz neke HTML datoteke. */
   #include <stdio.h>
   #include <ctype.h>

   /* Maksimalna duzina etikete */
   #define MAX_TAG 100

   #define OTVORENA 1
   #define ZATVORENA 2
   #define GRESKA 0

   /* Funkcija ucitava sledecu etiketu
      i smesta njen naziv u niz s duzine max.
      Vraca OTVORENA za otvorenu etiketu,
      ZATVORENA za zatvorenu etiketu,
      odnosno GRESKA inace */
   int gettag(FILE *f, char s[], int max)
   {
       int c, i;
       int zatvorenost=OTVORENA;

       /* Preskacemo sve do znaka '<' */
       while ((c=fgetc(f))!=EOF && c!='<')
           ;

       /* Nismo naisli na etiketu */
       if (c==EOF)
           return GRESKA;
   ```
   ```

```
    /* Proveravamo da li je etiketa zatvorena */
    if ((c=fgetc(f))=='/')
        zatvorenost=ZATVORENA;
    else
        ungetc(c,f);

    /* Citamo etiketu dok nailaze slova
       i smestamo ih u nisku */
    for (i=0; isalpha(c=fgetc(f))
        && i<max-1; s[i++] = c)
            ;

    /* Vracamo poslednji karakter na ulaz
       jer je to bio neki karakter koji nije
       slovo*/
    ungetc(c,f);

    s[i]='\0';

    /* Preskacemo atribute do znaka > */
    while ((c=fgetc(f))!=EOF && c!='>')
        ;

    /* Greska ukoliko nismo naisli na '>' */
    return c=='>' ? zatvorenost : GRESKA;
}

main()
{
    char tag[MAX_TAG];
    int zatvorenost;

    FILE* f;

    /* Ucitavamo ime datoteke */
    char ime_datoteke[256];
    printf("Unesite naziv html dokumenta iz kog se vrsi citanje etiketa: ");
    scanf("%s", ime_datoteke);

    /* Otvaramo datoteku i proveravamo da li smo uspeli */
    if ( (f = fopen(ime_datoteke, "r")) == NULL)
    {
        printf("Greska : datoteka %s ne moze biti otvorena\n",
                ime_datoteke);
        return 1;
```

```
    }

    while ((zatvorenost = gettag(f,tag,MAX_TAG))>0)
    {
     if (zatvorenost==OTVORENA)
        printf("Otvoreno : %s\n",tag);
     else
        printf("Zatvoreno : %s\n",tag);
    }

    fclose(f);
}
```

6 Strukture

1. ```
   /* Datoteka cije se ime unosi sa komandne linije sadrzi podatke o
   studentima (ime, prezime, broj indeksa). Podaci su korektno zadati.
   Nema vise 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;
    }

    /* ISPRAVITI */
    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;
}
```

2. ```
   /* Sa standardnog ulaza se ucitava niz od n (n<100) tacaka u ravni takvih
   da nikoje tri tacke nisu kolinearne. Tacke se zadaju parom svojih
   koordinata (celi brojevi). Ispitati da li taj niz tacaka 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");
}
```