# Programiranje 1

*Beleške sa vežbi*

Smer *Informatika*
Matematički fakultet, Beograd

Jelena Tomašević

November 27, 2005

# Sadržaj

# Čas 1

[1]

## 1.1 Ugnježdena petlja

**Primer 1** *Ilustracija dve ugnježdene petlje.*

```
#include<stdio.h>
int main()
{
int i,j;

for(i=1; i<=3; i++)
    {
    for(j=1; j<=3; j++)
        printf("%d * %d = %d\t", i, j, i*j);
    printf("\n");
    }
}

Izlaz:
1 * 1 = 1       1 * 2 = 2       1 * 3 = 3
2 * 1 = 2       2 * 2 = 4       2 * 3 = 6
3 * 1 = 3       3 * 2 = 6       3 * 3 = 9
```

**Primer 2** *Napisati funkciju koja izračunava zbir n-tih stepena brojeva od 1 do granice i program koji ilustruje rad ove funkcije.*

```
#include <stdio.h>
void Zbir_stepena (int n, int granica);
main()
{
    Zbir_stepena(2,5);
    Zbir_stepena(3,5);
    Zbir_stepena(4,10);
    return 0;
```

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

```
}

void Zbir_stepena (int n, int granica)
{

    int i,j;   /*brojaci u for petljama */
    long Zbir=0  ,  stepenovan ;

     /*spoljasnji for ciklus obavlja sumiranja*/
    for (i=1; i<=granica; Zbir +=stepenovan, ++i)
        /*unutrasnji for ciklus obavlja stepenovanje */
         for( stepenovan=1,j=1;  j<=n;  stepenovan*= (long) i, ++j) ;
    printf(" Zbir %d. stepena od 1 do %d jeste %ld\n", n,granica,Zbir);
}

Izlaz:
 Zbir 2. stepena od 1 do 5 jeste 55
 Zbir 3. stepena od 1 do 5 jeste 225
 Zbir 4. stepena od 1 do 10 jeste 25333
```

## 1.2 F-je za rad sa stringovima

**Primer 3** `string_reverse` - *obrće nisku karaktera.*

```
#include <stdio.h>

/* Zbog funkcije strlen */
#include <string.h>

/* Ova funkcija racuna duzinu date niske karaktera.
   Umesto nje, moguce je koristiti standardnu funkciju strlen .
 */
int string_length(char s[]) {
    int i;
    for (i = 0; s[i]; i++)
        ;

    return i;
}

/* Funkcija obrce nisku karaktera */
void string_reverse(char s[])
{
    int i, j;
    for (i = 0, j = string_length(s)-1; i<j; i++, j--)
    {
        int tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
```

```
    }

    /* Napomena : razlikovati prethodnu petlju od dve ugnjezdjene petlje:
    for ( i = 0; ....)
        for ( j = duzina(s)-1; ...
     */

}

main() {
    char s[] = "Zdravo svima";
    string_reverse(s);
    printf("%s\n", s);
}

Izlaz:
amivs ovardZ
```

**Primer 4** `strlen`, `strcpy`, `strcat`, `strcmp`, `strchr`, `strstr` - *manipulacija niskama karaktera. Vezbe radi, implementirane su funkcije biblioteke* `string.h`

```
#include <stdio.h>

/* Izracunava duzinu stringa */
int string_length(char s[]) {
    int i;
    for (i = 0; s[i]; i++)
        ;
    return i;
}


/* Kopira string src u string dest.
  Pretpostavlja da u dest ima dovoljno prostora. */
void string_copy(char dest[], char src[]) {
    /* Kopira karakter po karakter, sve dok nije iskopiran karakter '\0' */
    int i;
    for (i = 0; (dest[i]=src[i]) != '\0'; i++)
        ;

    /* Uslov != '\0' se, naravno, moze izostaviti :

    for (i = 0; dest[i]=src[i]; i++)
        ;
    */

}

/* Nadovezuje string t na kraj stringa s.
   Pretpostavlja da u s ima dovoljno prostora. */
```

```
void string_concatenate(char s[], char t[]) {
    int i, j;
    /* Pronalazimo kraj stringa s */
    for (i = 0; s[i]; i++)
        ;

    /* Vrsi se kopiranje, slicno funkciji string_copy */
    for (j = 0; s[i] = t[j]; j++, i++)
        ;
}

/* Vrsi leksikografsko poredjenje dva stringa.
    Vraca :
        0 - ukoliko su stringovi jednaki
        <0 - ukoliko je s leksikografski ispred t
        >0 - ukoliko je s leksikografski iza t
*/ int string_compare(char s[], char t[]) {
    /* Petlja tece sve dok ne naidjemo na prvi razliciti karakter */
    int i;
    for (i = 0; s[i]==t[i]; i++)
        if (s[i] == '\0') /* Naisli smo na kraj oba stringa,
                     a nismo nasli razliku */
            return 0;

    /* s[i] i t[i] su prvi karakteri u kojima se niske razlikuju.
       Na osnovu njihovog odnosa, odredjuje se odnos stringova */
    return s[i] - t[i];
}

/* Pronalazi prvu poziciju karaktera c u stringu s, odnosno -1
   ukoliko s ne sadrzi c */
int string_char(char s[], char c) {
    int i;
    for (i = 0; s[i]; i++)
        if (s[i] == c)
            return i;
        /* nikako
        else
            return -1;
        */
    /* Nije nadjeno */
    return -1;
}

/* Pronalazi poslednju poziciju karaktera c u stringu s, odnosno
-1
   ukoliko s ne sadrzi c */
int string_last_char(char s[], char c) {
    /* Pronalazimo kraj stringa s */
    int i;
```

```
    for (i = 0; s[i]; i++)
        ;

    /* Krecemo od kraja i trazimo c unazad */
    for (i--; i>=0; i--)
        if (s[i] == c)
            return i;

    /* Nije nadjeno */
    return -1;

    /*
    Koristeci string_length :

    for (i = string_length(s) - 1; i>0; i--)
        if (s[i] == c)
            return i;

    return -1;
    */
}

/* Proverava da li string str sadrzi string sub.
   Vraca poziciju na kojoj sub pocinje, odnosno -1 ukoliko ga nema
*/ int string_string(char str[], char sub[]) {
    int i, j;
    /* Proveravamo da li sub pocinje na svakoj poziciji i */
    for (i = 0; str[i]; i++)
        /* Poredimo sub sa str pocevsi od poziciji i
           sve dok ne naidjemo na razliku */
        for (j = 0; str[i+j] == sub[j]; j++)
            /* Nismo naisli na razliku a ispitali smo
               sve karaktere niske sub */
            if (sub[j+1]=='\0')
                return i;
    /* Nije nadjeno */
    return -1;
}

main() {
    char s[100];
    char t[] = "Zdravo";
    char u[] = " svima";

    string_copy(s, t);
    printf("%s\n", s);

    string_concatenate(s, u);
    printf("%s\n", s);
```

```
    printf("%d\n",string_char("racunari", 'n'));
    printf("%d\n",string_last_char("racunari", 'a'));

    printf("%d\n",string_string("racunari", "rac"));
    printf("%d\n",string_string("racunari", "ari"));
    printf("%d\n",string_string("racunari", "cun"));
    printf("%d\n",string_string("racunari", "cna"));
}

Izlaz:
Zdravo
Zdravo svima
4
5
0
5
2
-1
```