



Algoritmi i programiranje

Programski jezik C
Stringovi

Znakovni nizovi - Stringovi

Znakovni nizovi ili stringovi

predstavljaju niz znakovnih podataka (jedan ili više), završeni oznakom null (prog. jezik C).

Primer:

“Ovo je jedan string”

“Tekst izmedju navodnika”

Konstantni znakovni niz podataka –

niz znakova uokviren znacima navoda.

Podsetnik:

Niz (polje) predstavlja kontinualno uređenje podataka istog tipa.

Svaki objekat u nizu naziva se **element** niza.

Početni indeks niza je **0**.

Veličina niza je broj elemenata niza (maksimalni indeks + 1)

Opšti oblik deklaracije niza:

tip ime_niza [velicina]

Pristup elementima niza je preko indeksa

(u opsegu od 0 do veličine niza -1)

Operacije za rad sa stringovima:

- **Konkatenacija** – nadovezivanje vrednosti stringova

Primer: Ako označimo operaciju konkatenacije sa ||, rezultat primene konkatenacije na stringove:

"Moja najbolji drug je " || " moj prvi komsija Pera"

je: "Moja najbolji drug je moj prvi komsija Pera"

- **Poređenje** vrednosti stringova

Vrednosti stringova se upoređuju leksikografski, po engleskoj abecedi. Neki string je manji/veći od drugog ako je on po leksikografskom uređenju pre/posle njega

Primer:

String "A" je pre stringa "B"

String "Pera" je posle stringa "Mika"

(poređenje se vrši redom, slovo po slovo)

String "Perci" je posle stringa "Peric"

(slovo c je pre slova i, nadalje nije bitno)

- **Traženje** u stringu

Operacija traženja omogućava da u zadatom stringu pronađete i/ili izdvojite delove tj. druge stringove ili karaktere

Deklaracija stringa u C-u

U C-u **ne postoji** poseban tip za predstavljanje znakovih podataka, već se koristi tip `char`.

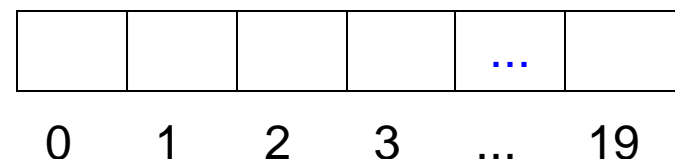
Podsetnik:

`char` – mali celobrojni podatak (dužine 1 bajt) koji može da primi i kod jednog znaka pa se koristi i za predstavljanje znakovnih podataka.

Primer: Predstavljanje znakovnih nizova:

```
char ime[20];  
char prezime[20];
```

ime



Ime je pointer na prvi znak u stringu!!!

VAŽNO: na kraju svakog znakovnog niza stoji simbol nultog znaka, koji se prikazuje kao **znakovna konstanta sa vrednošću `'\0'`**
(poznat kao *null terminated symbol* ili *null character*)

VAŽNO:

Sami ste odgovorni da niz završite nultim znakom!!
Ako zaboravite na završni znak, imaćete običan niz znakova !!

Inicijalizacija stringa

Dva načina za inicijalizaciju:

1. Listom individualnih znakova (karaktera), uključujući null karakter

```
char ime [ ] = {'M', 'i', 'l', 'a', 'n', '\0'};
```

2. Navođenjem konstantne vrednosti za niz znakova

```
char ime [ ] = {"Milan"};
```

Sadržaj rezervisane memorijske lokacije u oba slučaja:

ime

M	i	l	a	n	\0
0	1	2	3	4	5

Napomena:

- U oba slučaja nisu dati podaci o veličini niza (određuje je prevodilac)
- Kod prvog načina **morate da navedete** oznaku kraja stringa kod navođenja inicijalnih vrednosti !!!

Stringovi i naredbe za ulaz/izlaz

scanf()

Poziv:

```
scanf(<format>,<adr_ul_podatka1> [,<adr_ul_podatka2>]...)
```

printf()

Poziv:

```
printf(<format>[,<izraz1>][,<izraz>]...)
```

gde je:

<format> - Znakovni niz koji predstavlja definiciju konverzija koje treba izvršiti pri unosu/prikazu podataka.

%[w][h|l|L]<tip_konverzije>

%[-][+|][#][w[.d]][h|l|L]<tip_konverzije>

<tip_konverzije>

c – znakovna konverzija (rezultat je tipa **char**),

s – konverzija u znakovni niz (niz znakova između dva blanko znaka što znači da učitani niz ne sadrži bl. znake. Iza pročitanih znakova dodaje se **'\0'**)

Konverzija **s** označava da se prilikom učitavanja, podatak prenosi u operativnu memoriju računara u onom obliku kako je sa tastature prihvaćen (kao niz ASCII simbola), jedino se na kraj tog niza dodaje simbol **'\0'**.

scanf

U pozivu funkcije **scanf** navode se adrese memorijskih lokacija gde će pročitani podaci biti upisani, a imena nizova su ujedno i memorijske adrese prvih članova niza, **pri učitavanju znakovnih nizova dovoljno je navesti samo ime niza.**

Primer:

```
char ime[20];  
scanf("%s %s", ime, prezime);
```

M	i	l	a	n	'\0'
0	1	2	3	4	5

printf

Na isti način se `%s` konverzija koristi i u `printf` funkciji.

Na standardni izlaz se prenosi niz karaktera iz operativne memorije od prvog elementa u navedenom nizu do simbola `'\0'`.

Primer:

```
char ime[20], prezime[20];  
printf("%s %s\n", ime, prezime);
```

Primer:

```
char slovo = 'A';  
char boja [] = {"PLAVA"};
```

```
printf("%c je vrednost promenljive slovo", slovo);  
A je vrednost promenljive slovo
```

```
printf("Moja omiljena boja je %s", boja);  
Moja omiljena boja je PLAVA
```


Šta nije dobro kod stringova u C-u?

- C nema dobru podršku za rad sa stringovima;
- Praktično stringovi ne postoje, već se predstavljaju kao polje karaktera

Primer:

U C kodu NE MOŽETE navesti:

```
char ime[50];  
char prezime[50];  
char punoime[100];
```

```
ime = "Arnold"; /* nije dozvoljeno */  
prezime = " Schwarzenegger"; /* nije dozvoljeno */  
punoime = "Mr " + ime + prezime; /* nije dozvoljeno */
```

Ako su **s1** i **s2** C "stringovi" program ne može:

1. da dodeli vrednost jednog stringa drugom : **s1 = s2;**
2. da ih upoređuje: **... s1 < s2 ..**
3. da uradi konkatenciju u jedan string: **... s1 + s2 ...**
4. da funkcija kao rezultat vrati string.

Rad sa stringovima: biblioteka funkcija `<string.h>`

U kodu je neophodno navesti:

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

Napomena: Voditi računa o oznaci kraja stringa!!

Spisak osnovnih funkcija:

Kopiranje (dodela vrednosti stringu)

```
char *strcpy(const char *string1, const char *string2)
```

Kopira string2 u string1, uključujući oznaku kraja stringa.

```
char *strncpy(const char *string1, const char *string2, size_t n)
```

Kopira prvih n karaktera iz string2 u string1.

Konkatenacija

```
char *strcat(const char *string1, char *string2)
```

Dodaje string2 na string1 (konkatenacija).

```
char *strncat(const char *string1, char *string2, size_t n)
```

Dodaje n karaktera iz string2 u string1 (konkatenacija).

/* Primer za strcat */

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

int main()
{
    char example[100];

    strcpy(example, „ELFAK ");
    strcat(example, "is ");
    strcat(example, "over ");
    strcat(example, „50 ");
    strcat(example, "years ");
    strcat(example, "old.");

    printf("%s\n", example);

    return 0;
}
```

ELFAK is over 50 years old.

Spisak funkcija (nastavak):

Funkcije poređenja:

`int strcmp(const char *string1, const char *string2)`

Upoređuje string1 i string2 za određivanje *alphabetic* redosleda.

`int strncmp(const char *string1, const char *string2, size_t n)`

Upoređuje (leksički) prvih n karaktera dva stringa. Vraća 0 ako su string1=string2, <0 ako string1<string2 i >0 ako string1>string2

`int strcasecmp(const char *s1, const char *s2)`

Case insensitive verzija za strcmp().

`int strncasecmp(const char *s1, const char *s2, int n)`

Case insensitive verzija za strncmp().

Ostale funkcije

`char *strerror(int errnum)`

Poruka o grešci za zadati broj greške.

`size_t strlen(const char *string)`

Određuje dužinu stringa.

strlen()

Sintaksa: `len = strlen(ptr);`

gde je `len` ceo broj (integer) i `ptr` je pointer na `char`

Namena:

`strlen()` vraća dužinu niza bez oznake kraja.

Primer: Dužina stringa je 13 tj. *len* dobija vrednost 13.

```
int len;  
char str[15];  
strcpy(str, "Hello, world!");  
len = strlen(str);
```

strcpy()

Sintaksa: `strcpy(ptr1, ptr2);`

gde su `ptr1` i `ptr2` pointeri na `char`

Namena:

`strcpy()` se koristi za kopiranje null-terminated stringa u promenjivu.

```
char S[25];
```

```
char D[25];
```

Varijante:

Upis teksta u string:

```
strcpy(S, "This is String 1.");
```

Kopiranje celog stringa iz S u D:

```
strcpy(D, S);
```

Kopiranje ostatka stringa S u D:

```
strcpy(D, &S[8]);
```

Napomena: Voditi računa o oznaci kraja stringa!!

strncpy()

Sintaksa: `strncpy(ptr1, ptr2, n);`

gde je `n` ceo broj a `ptr1` i `ptr2` pointeri na `char`

Namena:

`strncpy()` se koristi za kopiranje dela stringa.

Treba voditi računa, pošto se `'\0'` kopira *samo* ako je deo stringa koji se kopira.

Varijante korišćenja:

`char S[25];`

`char D[25];`

Uz pretpostavku da se sledeći deo koda izvršava pre svake navedene stavke:

`strcpy(S, "This is String 1.");`

Kopiranje 4 karaktera od početka S u D i stavljanje null na kraju:

`strncpy(D, S, 4);`

`D[4] = '\0';`

Kopiranje dva karaktera iz sredine stringa S u D:

`strncpy(D, &S[5], 2);`

`D[2] = '\0';`

Kopiranje kraja stringa S u D:

`strncpy(D, &S[8], 15); //daje isti rezultat kao strcpy(D, &S[8]);`

Primer: Korišćenje ovih funkcija:

```
char *str1 = "ZDRAVO"; // deklaracija stringa - moze i ovako !!
char *str2;              // posto je ime stringa ukazatelj na prvi znak
int duzina;
duzina = strlen("ZDRAVO");
/* duzina = 6 */
(void) strcpy(str2, str1);
```

```
char *str1 = "ZDRAVO";
char *str2;
int duzina = 2;
strncpy(str2, str1, duzina);
/* str2 = "ZD" */
```

VAŽNO: str2 nije završeno null oznakom!!!

Funkcije `strncat()`, `strncmp()` i `strncpy()` su restriktivnija verzija originalnih funkcija (bez "n" u imenu) – obavljaju istu funkciju, ali za n karaktera

Kao u prethodnom primeru, njihovo korišćenje može narušiti zahtev zadavanja oznake kraja null karakterom!

strcmp()

Sintaksa: `diff = strcmp(ptr1, ptr2);`

gde je `diff` ceo broj a `ptr1` i `ptr2` pointeri na `char`

Namena: `strcmp()` se koristi za poređenje dva stringa. Poređenje se vrši karakter po karakter. Ako su stringovi identični, rezultat je nula (0). Kada se pronade razlika, prestaje se sa poređenjem, i ako je taj karakter u prvom stringu "manji" tj. pre (po ASCII) karaktera iz drugog stringa, vrati negativnu vrednost, inače pozitivnu

Primeri korišćenja:

```
char s1[25] = "pat";
```

```
char s2[25] = "pet";
```

```
//diff će imati negativnu vrednost nakon izvršenja sledećeg koda:
```

```
diff = strcmp(s1, s2);
```

```
//diff će imati pozitivnu vrednost nakon izvršenja sledećeg koda:
```

```
diff = strcmp(s2, s1);
```

```
//diff će imati vrednost nula (0) nakon izvršenja sledećeg koda:
```

```
diff = strcmp(s1, s1);
```

strncmp()

Sintaksa: `diff = strncmp(ptr1, ptr2, n);`

gde su `diff`, `n` celi brojevi a `ptr1` i `ptr2` pointeri na `char`

Namena: `strncmp()` se koristi za poređenje prvih `n` karaktera dva stringa.

Poređenje se vrši karakter po karakter. Ako su stringovi identični, rezultat je nula (0). Kada se pronade razlika, prestaje se sa poređenjem, i ako je taj karakter u prvom stringu "manji" tj. pre (po ASCII) karaktera iz drugog stringa, vrati negativnu vrednost, inače pozitivnu.

Primeri korišćenja:

```
char s1[25] = "pat";
```

```
char s2[25] = "pet";
```

```
//diff će imati negativnu vrednost nakon izvršenja sledećeg koda:
```

```
diff = strncmp(s1, s2, 2);
```

```
//diff će imati pozitivnu vrednost nakon izvršenja sledećeg koda:
```

```
diff = strncmp(s2, s1, 3);
```

```
//diff će imati vrednost nula (0) nakon izvršenja sledećeg koda:
```

```
diff = strncmp(s1, s1, 1);
```

Funkcije za traženje <string.h>

`char *strchr(const char *string, int c)`

Nalazi prvo pojavljivanje karaktera u stringu.

`char *strrchr(const char *string, int c)`

Pronalazi poslednje pojavljivanje karaktera c u stringu.

`char *strstr(const char *s1, const char *s2)`

Locira prvo pojavljivanje stringa s2 u stringu s1.

`char *strpbrk(const char *s1, const char *s2)`

Vraća pointer na prvo pojavljivanje u stringu s1 nekog karaktera iz stringa s2, ili null pointer ako nema karaktera iz s2 u s1

`size_t strspn(const char *s1, const char *s2)`

Vraća broj karaktera na početku s1 koji se poklapaju sa s2.

`size_t strcspn(const char *s1, const char *s2)`

Vraća broj karaktera na početku s1 koji se **ne** poklapaju sa s2.

`char *strtok(char *s1, const char *s2)`

Deli string s1 u sekvencu tokena, svaki od njih je ograničen jednim ili više karaktera iz stringa s2.

`char *strtok_r(char *s1, const char *s2, char **lasts)`

Kao strtok(), osim što pointer na string mora biti zadat od strane pozivne funkcije.

Primer korišćenja strtok

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

int main()
{
    const char str[80] = „Ovo je - www.anypoint.com - website";
    const char s[2] = "-";
    char *token;

    /* izdvajanje prvog tokena */
    token = strtok(str, s);

    /* izdvajanje ostalih tokena */
    while( token != NULL )
    {
        printf( " %s\n", token );
        token = strtok(NULL, s);
    }
    return(0);
}
```

Ovo je
www.anypoint.com
website

strchr() i strrchr()

Primer korišćenja ovih funkcija:

```
char *str1 = "Hello";  
char *ans;  
ans = strchr(str1, 'l');
```

Nakon izvršenja, **ans** ukazuje na lokaciju **str1 + 2**

strpbrk()

generalnija funkcija, koja traži prvo pojavljivanje bilo koje grupe karaktera:

```
char *str1 = "Hello";  
char *ans;  
ans = strpbrk(str1, "aeiou");
```

Sada **ans** pokazuje na lokaciju **str1 + 1**, lokaciju prvog **e**.

strstr()

vraća pointer na specificirani string za traženje ili null pointer ako taj string nije nađen. Ako **s2** ukazuje na string dužine 0 (tj, string ""), funkcija vraća **s1**:

```
char *str1 = "Hello";  
char *ans;  
ans = strstr(str1, "lo");
```

U ovom slučaju, **ans = str1 + 3**.

Primer: Korišćenje nekih navedenih funkcija za traženje

```
#include < string.h>
void main(){
    char linija[100], *deo_teksta;

    /* !!!! inicijalizacija stringa u kodu !!!!*/
    strcpy(linija,"zdravo, ja sam string;");

    printf("Linija: %s\n", linija);

    /* dodavanje na kraj stringa */
    strcat(linija," Ko si ti?");
    printf("Linija: %s\n", linija);

    /* pronadji duzinu linije - strlen vraca duzinu kao tip size_t */
    printf("Duzina linije: %d\n", (int)strlen(linija));

    /* pronadji pojavljivanje podnizova */
    if ( (deo_teksta = strchr ( linija, 'K' ) )!= NULL )
        printf("String koji pocinje sa \"K\" ->%s\n", deo_teksta);
}
```

Zadatak: Nalaženje najkraće reči

Napisati program na C-u za nalaženje najkraće od n reči unetih sa tastature.

Rešenje: Učitavaće se reč po reč sa tastature, izračunavati njihova dužina i porediti sa dužinom do tada najkraće unete reči. Ukoliko dužina tekuće reči bude manja od dužine najkraće, tekuća reč će se kopirati u najkraću reč. Na početku će se za dužinu najkraće reči uzeti vrednost veća od maksimalne moguće dužine reči.

Napomena: U navedenom rešenju nisu korišćene funkcije iz <string.h>. Sa strane je navedeno gde se delovi koda mogu zameniti odgovarajućim funkcijama iz ove biblioteke.

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

```
main()
```

```
{
```

```
    char rec[20], minrec[20];
```

```
    int i, j, n, minduzina, duzina;
```

```
    printf("unesite broj reci\n");
```

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

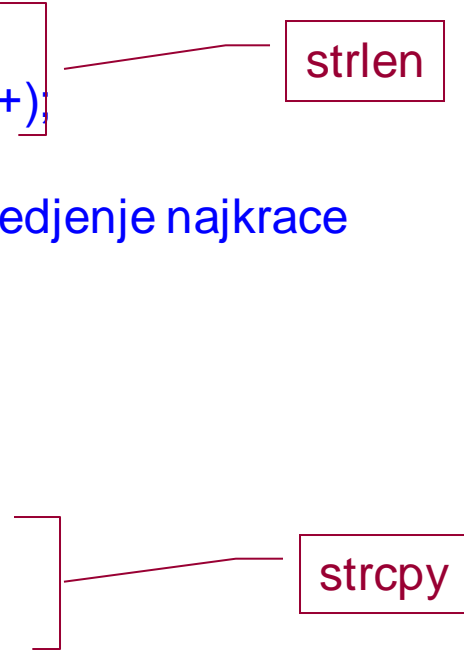
```
    /* s obzirom da je za rec predvidjeno maksimalno 20
```

```
    karaktera, duzina reci moze biti do 19 slova*/
```

```
    minduzina=20;
```

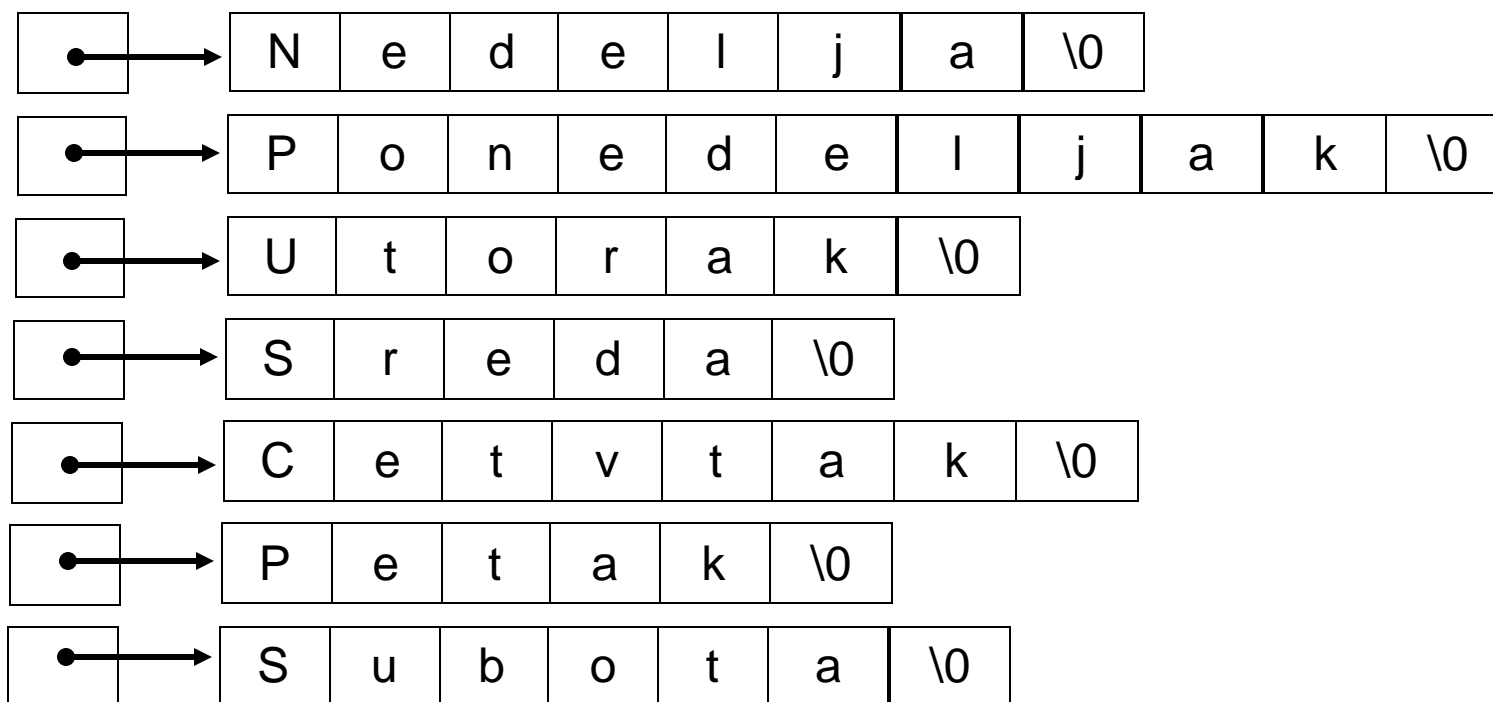
Nalaženje najduže reči (2)

```
for (i=0; i<n; i++)  
{  
    printf("unesite sledecu rec\n");  
    scanf("%s", rec);  
  
    // odredjivanje duzine reci  
    for( duzina=0; rec[duzina]!='\0'; duzina++);  
  
    // da li je uneta rec kraca od pre toga odredjenje najkrace  
    if( duzina<minduzina )  
    {  
        // kopiranje reci  
        j=0;  
        do  
            minrec[j]=rec[j];  
        while( rec[j++] != '\0' );  
    }  
    printf("najkraca rec je: %s", minrec);  
}
```



Nizovi pokazivača

```
static char *imedana[ ] = {  
    "Nedelja", "Ponedeljak",  
    "Utorak", "Sreda", "Cetvrtak",  
    "Petak", "Subota"  
};
```



Primer: String kao polje karaktera

```

#include<stdio.h>
#include<string.h>
#define MAX_DUZ_STRING 80
int main() {
    /* stringovi su polja/ nizovi karaktera
     * čiji je poslednji element NULL
     * karakter koji je različit od '0' */
    char S[MAX_DUZ_STRING];
    int l, i;

    S[0] = 'a';
    S[1] = 'b';
    S[2] = 'c';
    S[3] = 'd';
    S[4] = 'e';
    S[5] = 'g';
    S[6] = '0';
    S[7] = 0;
    l = strlen(S);
    printf("S:\t%s\n",S);
    printf("duzina:\t%d\n",l);

    /* prikaz karaktera iz S */
    printf("Unapred\n");
    for (i = 0; i < l; ++i)
        printf("A[%d] = %c\n",i,S[i]);

    /* prikaz karaktera iz stringa S unazad */
    printf("\nUnazad\n");
    for (i = l-1; i >= 0; --i)
        printf("A[%d] = %c\n",i,S[i]);
}

```

Primer: String UI

```

#include<stdio.h>
#include<string.h>
#define MAX_DUZ_STRING 80
int main() {

    char S1[MAX_DUZ_STRING];
    char S2[MAX_DUZ_STRING];

    int i, l;
    printf("String:\t");
    scanf("%s",S1);
    /* kopiranje svih karaktera uključujući
     * završni NULL karakter!
     */
    l = strlen(S1);
    /* umesto ove petlje dole bolje je koristiti
     funkciju iz biblioteke strcpy(S2,S1) */
    for (i = 0; i < l+1; ++i)
        S2[i] = S1[i];
    /* promena originalnog S1 */
    S1[0] = S1[1] = S1[2] = '*';
    S1[3] = 0;
    /* prikaz oba stringa */
    printf("S1:\t%s\n",S1);
    printf("S2:\t%s\n",S2);
}

```

Primer: Poređenje stringova

```
#include<stdio.h>
#include<string.h>
#define MAX_DUZ_STRING 80
int main() {

    /* Proverite koji je rezultat sledecih naredbi
     * za razlicite vrednosti stringova S1, S2?
     */
    char S1[MAX_DUZ_STRING];
    char S2[MAX_DUZ_STRING];

    int i, l, res;
    printf("String1:\t");
    scanf("%s",S1);
    printf("String2:\t");
    scanf("%s",S2);
    res = strcmp(S1,S2);
    printf("strcmp(%sS1,%sS2) = %d\n",S1,S2,res);
}
```

Primer: Kopiranje stringova

```
#include<stdio.h>
#include<string.h>
#define MAX_DUZ_STRING 80
int main() {

    /* Napomena: Stringovi nisu kao druge regularne promenljive
     * Morate biti pazljivi kod poredjenja stringova i dodele
     * Proverite rezultat koji se prikazuje na kraju main funkcije !!!
     * Da li je to ono sto ocekujete?
     */
    char* S1 = "AAAAAAAAAAAA";
    char* S2 = "BBBBBBBBBBBB";

    int i, l;
    /* dodela S1 u S2 */
    S2 = S1;
    /* promena S1 */
    S1[0] = S1[1] = S1[2] = '*';
    S1[3] = 0;
    /* prikaz oba stringa */
    printf("S1:\t%s\n",S1);
    printf("S2:\t%s\n",S2);
}
```

Primer: Jednakost stringova

```
#include<stdio.h>
#include<string.h>
#define MAX_DUZ_STRING 80
int main() {

    /* Napomena: Stringovi nisu kao druge
     * regularne promenljive
     * Morate biti pazljivi kod poredjenja stringova
     * i dodele */
    char* S1 = "AAAAAAAAAAAA";
    char* S2 = "AAAAAAAAAAAA";

    int cmp1, cmp2, cmp3, cmp4;

    cmp1 = (S1 == S2);
    cmp2 = strcmp(S1,S2);

    printf("S1:\t%s\n",S1);
    printf("S2:\t%s\n",S2);
    printf("S1 == S2:\t%d\n",cmp1);
    printf("strcmp(S1,S2):\t%d\n",cmp2);
    S2 = S1;
    cmp3 = (S1 == S2);
    cmp4 = strcmp(S1,S2);
    printf("\nposle dodele\n");
    printf("S1:\t%s\n",S1);
    printf("S2:\t%s\n",S2);
    printf("S1 == S2:\t%d\n",cmp3);
    printf("strcmp(S1,S2):\t%d\n",cmp4); }
```

Primer: Funkcija *strcat*

```
#include<string.h>
#include<stdio.h>
#define MAX_DUZ_STRING 80
int main() {
    /* strcat je funkcija koja omogucava konkatenciju:
     * ona dodaje string iza poslednjeg karaktera navedenog stringa
     */
    char S1[MAX_DUZ_STRING];
    char S2[MAX_DUZ_STRING];
    strcat(S1,S2);
    printf("S1:\t");
    scanf("%s",S1);
    printf("S2:\t");
    scanf("%s",S2);
    strcat(S1,S2);
    printf("\nPosle primene strcat(S1,S2)\n");
    printf("S1:\t%s\n",S1);
    printf("S2:\t%s\n",S2);
}
```

Primer: Unos stringova sa tastature

```
#include<string.h>
#include<stdio.h>
#define MAX_DUZ_STRING 100
int main() {

    /* ovaj program prikazuje unos
    * sa tastature rec po rec (string po string)
    * i povecava brojac kod svake unete reci,
    * sve dok se ne unese rec "kraj"
    */

    char S[MAX_DUZ_STRING];
    int brojac;

    brojac = 0;
    do {
        printf("string:\t");
        scanf("%s",S);
        if (strcmp(S,"kraj") != 0) ++brojac;
    } while (strcmp(S,"kraj") != 0);

    printf("broj unetih reci:\t%d\n", brojac);
}
```