

Programiranje 1

Programski jezik C

8. čas

Mirko Spasić

STRINGOVI

- **Niska karaktera ili string** je niz karaktera koji se završava karakterom **'\0'**.
- Karakter **'\0'** ima ASCII vrednost 0 pa se može tumačiti kao logička vrednost "netačno".
- Format za ispis niske pomoću funkcije printf je **%s**.

STRINGOVI

- Konstante tipa niska se navode između znakova navodnika. Na primer

```
char s[]="Primer stringa";
```

- Ovo je isto što i:

```
char s[] = {'P', 'r', 'i', 'm', 'e', 'r', ' ', 's', 't', 'r', 'i',  
'n', 'g', 'a', '\0'};
```

STRINGOVI - primer

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

```
int main() {
```

```
    /* String se moze inicijalizovati na dva nacina:
```

```
    1) Definiseemo niz tipa char, i navedemo listu inicijalizatora,  
    pri cemu poslednji u listi mora biti '\0'
```

```
    2) Definiseemo niz tipa char, i navedemo stringovsku  
    konstantu izmedju dvostrukih navodnika. U ovom slucaju se  
    podrazumeva da se ovako definisan niz zavrшава na  
    "propisan" nacin, znakom '\0' */
```

```
    char s1[] = {'o','v','o',' ','j','e',' ','s','t','r','i','n','g',' ','1','\0'};
```

```
    char s2[] = "ovo je string 2";
```

STRINGOVI - primer

```
int i;  
char *p;  
  
/* Ispis prvog niza  
   karaktera */  
printf("String 1: ");  
i = 0;  
while(s1[i] != '\0')  
    putchar(s1[i++]);  
putchar('\n');
```

```
/* Ispis drugog  
   niza */  
printf("String 2: ");  
p = s2;  
while(*p != '\0')  
    putchar(*p++);  
putchar('\n');  
}
```

STRINGOVI

- *Kakva je razlika između 's' i "s"?*
 - *'s' je karakter*
 - *"s" je string ili niz od dva karaktera 's' i '\0'.*

STRINGOVI - primer

- *Funkcija za ispis niske karaktera - demonstrira prenos niske karaktera u funkciju.*

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

```
/* Uz nisku karaktera nije potrebno  
prenositi dimenziju ukoliko se postuje  
dogovor da se svaka niska  
zavrsava karakterom '\0'.*/
```

```
void stampaj_nisku(char s[]) {  
    int i;  
    for (i = 0; s[i] != '\0'; i++)  
        /* s[i] ce biti netacno samo  
        kada bude jednako sa '\0'  
        ciji je ASCII kod 0. */  
        putchar(s[i]);  
}  
int main() {  
    stampaj_nisku("Zdravo\n");  
    return 0;  
}
```

Izlaz: Zdravo

STRINGOVI - primer

- *Funkcija za učitavanje reči sa ulaza u nisku karaktera.*

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

```
#include <ctype.h>
```

/ Potrebno je zbog funkcije isspace.*

Funkcija isspace ispituje da li je

karakter praznina (blanko, tabulator

ili prelazak u novi red).

*Ona je definisana u okviru ctype.h */*

/ Funkcija učitava rec sa*

standardnog ulaza i smesta je u

niz karaktera s. Ovo uspeva zbog toga sto

se po vrednosti prenosi adresa pocetka

*niza, a ne ceo niz */*

```
void ucitaj_rec(char s[]) {  
    int c, i = 0;  
    while ( ! isspace(c=getchar()) )  
        s[i++] = c;  
    s[i] = '\0';  
}  
  
int main() {  
    /* Obavezno je alocirati memoriju za  
    niz karaktera */  
    char s[100];  
    ucitaj_rec(s);  
    /* Format za ispis stringa se zadaje  
    kao %s */  
    printf("%s\n", s);  
    return 0;  
}
```


STRINGOVI - primeri za vežbu

- Napisati funkciju koja računa dužinu stringa, funkciju koja okreće string i program koji ih testira.
- Napisati funkciju koja uklanja beline, tabilatore, ili znakove za kraj reda sa kraja stringa.
- Implementirati sledeće funkcije koje postoje u zaglavlju string.h: strcpy, strcat, strstr, strchr, strcmp.
- Napisati funkciju koja uklanja dati znak c, gde god da se nađe u nekom stringu c.

ZADATAK za domaći

- *Napisati program koji na osnovu niske cifara broja u dekadnom zapisu izračunava vrednost broja, kao i obrnuto: na osnovu vrednosti broja formira string koji sadrži zapis broja u dekadnom sistemu.*

LINEARNA PRETRAGA NIZOVA

- Funkcija pretražuje niz celih brojeva dužine n , tražeći u njemu element x . Pretraga se vrši prostom iteracijom kroz niz.
- Ako se element pronađe funkcija vraća indeks pozicije na kojoj je pronađen. Ako element nije pronađen u nizu, funkcija vraća -1 , kao indikator neuspješne pretrage.

Linearna pretraga

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

```
#define MAX 100
```

```
int linearna_pretraga(int a[], int n, int x) {
```

```
    int i;
```

```
    for(i = 0; i < n; i++)
```

```
        if (a[i] == x)
```

```
            return i;
```

```
    return -1;
```

```
}
```

Linearna pretraga

```
int main() {  
    int a[MAX];  
    int n, i, x;  
  
    /* Unosimo dimenziju niza */  
    printf("Uneti dimenziju: ");  
    scanf("%d", &n);  
  
    /* Unosimo elemente */  
    printf("Uneti elemente niza:\n");  
    for(i = 0; i < n; i++)  
        scanf("%d", &a[i]);
```

```
    /* Unosimo broj koji se trazi */  
    printf("Uneti element koji se trazi: ");  
    scanf("%d", &x);  
  
    /* Pretrazujemo niz */  
    i = linearna_pretraga(a, n, x);  
  
    /* Ispis poruke */  
    if(i == -1)  
        printf("Element nije u nizu\n");  
    else  
        printf("Element je u nizu na poziciji %d\n", i);  
    return 0;  
}
```

Sortiranje nizova

Selection - sort

- Program demonstrira sortiranje niza celih brojeva, metodom sortiranja izborom najmanjeg elementa.
- U prvoj iteraciji se traži najmanji element u nizu i postavlja se na početnu poziciju (zamenom sa elementom koji se na toj poziciji nalazio ranije). U sledećoj iteraciji se traži najmanji element među preostalim elementima, i postavlja se na sledeću poziciju, itd.


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

```
void sortiraj (int a[], int n) {
```

```
    int i, j;
```

```
    int min;
```

```
    int pom;
```

```
    /* U svakoj iteraciji ove petlje se
    pronalazi najmanji element medju
    elementima a[i], a[i+1],...,a[n-1], i
    postavlja se na poziciju i, dok se
    element na poziciji i premesta na
    poziciju min, na kojoj se nalazio
    najmanji od gore navedenih
    elemenata. */
```

```
    for (i = 0; i < n - 1; i++) {
```

```
        /* Unutrasnja petlja pronalazi poziciju
        min, na kojoj se nalazi najmanji od
        elemenata a[i],...,a[n-1]. */
```

```
        min = i;
```

```
        for (j = i + 1; j < n; j++)
```

```
            if (a[j] < a[min])
```

```
                min = j;
```

```
        /* Zamena elemenata na pozicijama (i) i min.
        Ovo se radi samo ako su (i) i min razliciti */
```

```
        if (min != i) {
```

```
            pom = a[i];
```

```
            a[i] = a[min];
```

```
            a[min] = pom;
```

```
        }
```

```
    }
```

```
}
```



```
/* Test program */
```

```
int main () {
```

```
    int a[MAX];
```

```
    int n;
```

```
    int i;
```

```
    /* Unosimo dimenziju niza */
```

```
    printf ("Uneti dimenziju niza: ");
```

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

```
    /* Unosimo elemente niza */
```

```
    printf ("Uneti elemente niza:\n");
```

```
    for (i = 0; i < n; i++)
```

```
        scanf ("%d", &a[i]);
```

```
    /* Sortiramo niz */
```

```
    sortiraj (a, n);
```

```
    /* Ispisujemo sortirani niz */
```

```
    printf ("Niz nakon sortiranja:\n");
```

```
    for (i = 0; i < n; i++)
```

```
        printf ("%d ", a[i]);
```

```
    putchar ('\n');
```

```
}
```

Binarna pretraga

- Binarna pretraga je mnogo efikasnija od linearne, jer ne moramo proći kroz sve elemente da bi zaključili da li se traženi element nalazi u nizu ili ne.
- Da bi mogli da primenimo binarnu pretragu, niz koji pretražujemo mora da bude SORTIRAN.
- Funkcija koja radi binarnu pretragu vraća indeks u nizu na kom se nalazi traženi element, ili vraća -1 kao indikator da se traženi element ne nalazi u nizu.

Binarna pretraga - primer

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

```
#define MAX 100
```

```
/* Funkcija trazi u sortiranom nizu  
a[] duzine n broj x. Vraca indeks  
pozicije nadjenog elementa
```

```
ili -1, ako element nije pronadjen */
```

```
int binarna_pretraga (int a[], int n, int x) {
```

```
    int l = 0;
```

```
    int d = n - 1;
```

```
    /* Dokle god je indeks l levo od  
    indeksa... d */
```

```
    while (l <= d) {
```

```
        /* Racunamo sredisnji indeks */
```

```
        int s = (l + d) / 2;
```

```
        /* Ako je sredisnji element veci od  
        x, tada se x mora nalaziti u levoj  
        polovini niza */
```

```
        if (x < a[s])
```

```
            d = s - 1;
```

```
/* Ako je sredisnji element manji od  
x, tada se x mora nalaziti u desnoj  
polovini niza */
```

```
    else if (x > a[s])
```

```
        l = s + 1;
```

```
    else
```

```
        /* Ako je sredisnji element  
jednak x, tada smo pronasli x na  
poziciji s */
```

```
        return s;
```

```
    }
```

```
/* ako nije pronadjen vracamo -1 */
```

```
return -1;
```

```
}
```

```
int main () {
```

```
    int a[MAX];
```

```
    int n, i, x;
```

```
    /* Unosimo dimenziju niza */
```

```
    printf ("Uneti dimenziju niza: ");
```

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

```
    /* Unosimo elemente niza */
```

```
    printf ("Uneti elemente niza:\n");
```

```
    for (i = 0; i < n; i++)
```

```
        scanf ("%d", &a[i]);
```

```
    /* Sortiramo niz */
```

```
    sortiraj (a, n);
```

```
/* Ispisujemo sortirani niz */  
printf ("Niz nakon sortiranja:\n");  
for (i = 0; i < n; i++)  
    printf ("%d ", a[i]);  
putchar ('\n');
```

```
/* Unosimo broj koji se trazi */  
printf("Uneti trazen broj: ");  
scanf("%d", &x);
```

```
/* Pozivamo funkciju za sortiranje */  
i = binarna_pretraga(a, n, x);  
  
/* Prikazujemo rezultat */  
if(i == -1)  
    printf("Element nije pronadjen\n");  
else  
    printf("Element pronadjen na  
           poziciji %d\n", i);  
return 0;  
}
```