Programski jezik C, Stringovi

Stringovi su posebna vrsta nizova.
String konstanta je niz znakova.

Interna reprezentacija nekog stringa sadrži nulti znak, '\0' , na kraju, pa je neophodan memorijski prostor za jedan veći od broja znakova između navodnika. ASCII kod nultog znaka je 0.

"hello"

h	e	l	l	o	\0

Ovakva reprezentacija znači da ne postoji ograničenje na to koliko dugačak neki string može biti, ali programi moraju da pregledaju ceo string da bi utvrdili njegovu dužinu.

*** Funkcija strlen(s) standardne biblioteke vraća dužinu svog argumenta s, isključujući završni znak '\0'
Evo naše verzije:

```
/* strlen: vraca duzinu od s */
int strlen(char s[]){
   int i;
   i=0;
   while( s[i]!='\0' )
       i++;
    return i;
}
```

strlen() i druge funkcije za stringove deklarisane su u zaglavlju <string.h>

*** Funkcija ucitaj_rec() učitava sa ulaza, u niz dimenzije lim, na čiji početak pokazuje pokazivač s, reč, tj. čita do kraja ulaza, prve beline ili dok ne učita najviše lim-1 karakter. Poslednje mesto u nizu na koji pokazuje s se čuva za znak '\0' kojim se mora završiti svaki string. Vraća dužinu učitane reči.

```
int ucitaj_rec(char s[], int lim){
    int c, i;
    for(i=0; i<lim-1 && (c=getchar())!=EOF)
           && c!='\n' && c!=' ' && c!='\t'; i++)
        s[i]=c;
    s[i]='\0';
    return i;
}
```

*** Funkcija squeeze(s, c) uklanja sve primerke znaka c iz stringa s

```
/* squeeze: brise sve c iz s */
void squeeze( char s[], int c){
  int i, j;
  for(i=j=0; s[i]!='\0'; i++)
       if(s[i]!=c)
            s[j++]=s[i];
  s[j]='\0';
}
```

svaki put kada se naiđe na znak koji nije jednak c, on se kopira na trenutnu j poziciju i samo tada se promenljiva j uvećava kako bi bila spremna za sledeći znak. To je ekvivalentno ovom kodu:

```
if( s[i]!=c ){
   s[j]=s[i];
   j++;
}
```

*** Standardna funkcija strcat(s, t) spaja string t na kraj stringa s. Funkcija strcat() pretpostavlja da u stringu s ima dovoljno mesta za ovu kombinaciju. Naša funkcija strcat() ne vraća nijednu vrednost; verzija iz standardne biblioteke vraća pokazivač na rezultujući string.

```
/* strcat:
   nadovezuje t na s;
   s mora biti dovoljno velik */
void strcat( char s[], char t[] ){
   int i, j;
   i = j = 0;
   while( s[i] != '\0' )
      i++;
   while( (s[i++]=t[j++]) != '\0' )
      ;
}
```

```
void strcat( char *s, char *t ){
    while(*s != '\0' )
         s++;
    for( ; *t !=0; s++, t++)
         *s = *t;
    *s='\0';
}
```

*** Funkcija atoi() za pretvaranje stringa u njegov numerički ekvivalent.

Ulazni string može da sadrži vodeći prazan prostor i opcioni znak + ili –

koraci:

- preskočiti prazan prostor, ako postoji
- pročitati znak broja, ako postoji
- pročitati celobrojni deo i pretvoriti ga

U svakoj fazi se obavlja odgovarajući deo i ostavlja čisto stanje za naredni korak
Ceo postupak se završava kada se naiđe na prvi znak koji ne može biti deo nekog broja

```
#include <ctype.h>
int atoi(char s[]){
  int i, n, sign;
  for( i=0; isspace(s[i]); i++)  /* preskacemo beline */
       ;
  sign = (s[i]=='-') ? -1 : 1;
  if( s[i]=='+' || s[i]=='-' )  /* preskacemo znak */
       i++;
  for(n=0; isdigit(s[i]); i++)
       n = 10*n + (s[i]-'0');

  return sign*n;
}
```

*** reverse(s) preokreće string s u mestu

```
#include<string.h>
/* reverse: obrce string s u mestu */
void reverse(char s[]){
    int c, i, j;
    for( i=0, j=strlen(s)-1; i<j; i++,j--){
       c=s[i];
       s[i]=s[j];
       s[j]=c;
    }
}
```

*** itoa() ceo broj pretvara u nisku znakova (inverzna od atoi()). Jednostavni metodi za generisanje cifara ih generišu u pogrešnom redosledu. String generišemo naopačke, a zatim ga preokrenemo.

```
/* itoa: konvertuje n u karaktere u s */
void itoa( int n, char s[] ){
   int i, sign;

   if( (sign=n) < 0)  /* upamtimo znak */
      n=-n;           /* n je sada pozitivno */

   i = 0;
   do{       /* generisemo cifre u inverznom poretku */
      s[i++]=n%10 + '0';   /* dobijanje sledece cifre */
    }while( (n/=10)>0 );

  if( sign<0 )
      s[i++]='-';

  s[i]='\0';

  reverse(s);
 }
```

*** trim() uklanja završne razmake, tabulatore i nove redove sa kraja datog stringa, koristeći break za izlaz iz petlje kada se naiđe na krajnji desni znak koji nije razmak, tabulator ili novi red.

```
/* trim: uklanja beline sa kraja s */
int trim(char s[]){
  int n;

  for(n=strlen(s)-1; n>=0; n--)
     if( s[n]!=' ' && s[n]!='\t' && s[n]!='\n' )
        break;
  s[n+1]='\0';
  return n+1;
}
```

funkcija trim() vraća dužinu stringa. Petlja for počinje od kraja i ide unazad tražeći prvi znak koji nije razmak, tabulator ili novi red. Petlja se prekida kada se takav znak pronađe ili kada n postane negativno (tj. kada je ceo string pretražen). Ovo je ispravno ponašanje čak i kada je string prazan ili ako sadrži samo znake praznog prostora

*** strindex(s, t) vraća poziciju ili indeks gde string t počinje u stringu s, ili -1 ako s ne sadrži t.
Pošto nizovi u C-u počinju od pozicije nula, indeksi će biti nula ili pozitivni i zato neka negativna vrednost kao što je to -1 jeste pogodna za signaliziranje nepripadanja stringa t.
(standardna biblioteka obezbeđuje funkciju strstr() koja je slična funkciji strindex(), osim što ona vraća pokazivač, a ne indeks)

```
/* strindex: vraca indeks t-a u s-u, -1 ako ga nema */
int strindex( char s[], char t[] ){
  int i, j, k;
  for( i=0; s[i]!='\0'; i++){
     for( j=i, k=0; t[k]!='\0' && s[j]==t[k]; j++,k++)
        ;
     if(k>0 && t[k]=='\0')
       return i;
  }
  return -1;
}
```

*** atof(s) string s pretvara u njemu ekvivalentan broj sa pokretnim zarezom dvostruke tačnosti.
Prihvata opcioni znak broja ili decimalnu tačku, kao i neobavezni celobrojni deo ili deo sa decimalama. Nije funkcija visokog kvaliteta za konverziju ulaza. Standardna biblioteka sadrži funkciju atof() deklarisanu u zaglavlju <stdlib.h>

```
#include<ctype.h>
double atof(char s[]){
  double val, power;
  int i, sign;

  for( i=0; isspace(s[i]); i++)  /* preskakanje belina */
       ;
  sign = ( s[i]=='-' ) ? -1 : 1;

  if( s[i]=='+' || s[i]=='-' )
    i++;

  for(val=0.0; isdigit(s[i]); i++)
     val = 10.0*val + (s[i]-'0');

  if(s[i]=='.')
     i++;
```

```
    for(power=1.0; isdigit(s[i]); i++){
        val = 10.0*val + (s[i]-'0');
        power *= 10;
    }

    return sign*val/power;
}
```

Inicijalizacija

Znakovni nizovi su specijalan slučaj inicijalizacije.
Može:

```
char niska[] = "zdravo";
```

i to je skraćeni zapis ekvivalentne deklaracije:

```
char niska[] = {'z', 'd', 'r', 'a', 'v', 'o', '\0'};
```

Veličina niza je 7 (6 slova plus završni znak '\0')

Stringovi, printf i scanf

specifikator konverzije %s

```
char niska[]="ovo je niska";
char rec[20];
...
printf("%s", niska);   /* ispisuje sve karaktere stringa niska do
                          terminirajuce nule, ne ukljucujuci i nju */
scanf("%s", rec);  /* u niz rec ucitava sekvencu ne-blanko karaktera
sa ulaza. niz mora biti dovoljno velik da prihvati karaktere i '\0'
karakter.  ( blanko karakteri: ' ' ,'\n' , '\t' )
```

*** Kada se ime niza prenosi nekoj funkciji, ono što se prenosi je lokacija početnog elementa niza. Unutar pozvane funkcije, ovaj argument je lokalna promenljiva i zato parametar koji je ime niza predstavlja pokazivač, tj. promenljivu koja sadrži adresu.

*** druga verzija funkcije strlen():

```
/* strlen: vraca duzinu stringa s */
int strlen(char *s){
    int n;
    for(n=0; *s != '\0'; s++)
        n++;
    return n;
}
```

treća verzija:

```
int strlen(char *s){
    char *p;
    for(p=s; *p!='\0'; p++)
        ;
    return p-s;
}
```

pošto je s pokazivač, njegovo uvećavanje za 1, s++ ne utiče na znakovni string u funkciji koja je pozvala strlen(), već samo za 1 uvećava kopiju tog pokazivača koja je privatna za strlen.

Svi ovi pozivi:

```
strlen("hello, world");   /* string konstanta */
strlen( array );   /* char array[100] */
strlen( ptr );      /* char *ptr */
```

su ispravni.

Znakovni pokazivači i funkcije

Ako je promenljiva pmessage deklarisana na sledeći način:

```
char *pmessage;
```

tada naredba:

```
pmessage = "now is the time";
```

dodeljuje promenljivoj pmessage pokazivač na dati niz znakova. Ovo nije kopiranje stringa; radi se samo sa pokazivačima.

U sledećim definicijama postoji jedna važna razlika:

```
char nmessage[] = "now is the time";   /* niz */
char *pmessage = "now is the time";   /* pokazivac */
```

nmessage je niz, tačno onoliko veliki koliko je potrebno za smeštanje navedene niske znakova i znaka '\0' kojima se ovaj niz inicijalizuje. Pojedinačni znakovi unutar niza se mogu promeniti, ali nmessage uvek pokazuje na isti memorijski prostor.

pmessage je pokazivač koji je inicijalizovan da pokazuje na string konstantu. Ovaj pokazivač se kasnije može promeniti tako da pokazuje na nešto drugo, ali je rezultat nedefinisan ako pokušate da modifikujete string konstantu.

```
nmessage++;   /* NE MOZE */
nmessage[2]='t';  /* MOZE */
nmessage=pmessage; /* NE MOZE */
```

```
pmessage++; /* MOZE */
pmessage[2]='t'; /* NE MOZE */
pmessage=nmessage; /* MOZE */
/* a posle toga moze i
   pmessage[2]='t';
   jer pmessage ne pokazuje vise
   na string konstantu */
```

Sledi nekoliko verzija dve korisne funkcije.

*** Prva funkcija je strcpy(s,t) koja kopira string t u string s.
s=t kopira pokazivač, a ne znakove
da bismo kopirali znakove, treba nam petlja

verzija sa indeksima:

```
void strcpy(char *s, char *t){
   int i;
   i=0;
   while((s[i]=t[i])!='\0')
     i++;
   }
```

verzija sa pokazivačima:

```
void strcpy(char *s, char *t){
   while((*s=*t)!='\0'){
       s++;
       t++;
    }
}
```

pošto se argumenti prenose po vrednosti, funkcija strcpy() može koristiti parametre s i t na svaki način koji joj odgovara. Ovde su to pogodno inicijalizovani pokazivači koji se pomeraju duž nizova po jedan znak, sve dok '\0' koji završava string t ne bude kopiran u s.

Još jedan način:

```
void strcpy(char *s, char *t){
   while((*s++ = *t++) != '\0')
      ;
}
```

vrednost izraza *t++ je znak na koji t pokazuje pre nego što se t uveća za jedan. Postfiksni operator ++ ne menja t pre nego što se ovaj znak uzme. Na isti način, taj znak se smešta u staru poziciju pokazivača s pre nego sto se s uveća za 1. Ovaj znak je i vrednost koja se upoređuje sa '\0' radi kontrolisanja petlje.
Krajnji rezultat je da se znakovi kopiraju iz t u s sve do završnog znaka '\0' uključujući i njega.

Poređenje sa '\0' je suvišno jer je pitanje samo da li je izraz nula.

```
void strcpy(char *s, char *t){
    while( *s++ = *t++ )
        ;
}
```

*** Funkcija strcpy() u standardnoj biblioteci (<string.h>) vraća ciljni string kao svoju povratnu vrednost.

*** strcmp(s, t) poredi stringove s i t i vraća negativnu vrednost, nulu ili pozitivnu vrednost ako je s leksikografski manje od, jednako ili veće od t. Ova vrednost se dobija oduzimanjem znakova na prvoj poziciji u kojoj se s i t razlikuju

verzija sa indeksima

```
int strcmp(char *s, char *t){
   int i;
   for(i=0; s[i]==t[i]; i++)
      if(s[i]=='\0')
        return 0;
   return s[i] - t[i];
 }
```

verzija sa pokazivačima

```
int strcmp( char *s, char *t){

     for( ; *s==*t; s++,t++ )
          if(*s=='\0')
              return 0;
     return *s - *t;
}
```

još jedna verzija sa pokazivačima

```
int strcmp(char *s, char *t){
   for( ; *s==*t && *s!='\0'; s++, t++)
          ;
   return *s - *t;
}
```

*** Ako za nešto što nam treba već postoji funkcija u <string.h> koja to radi, koristićemo nju (nećemo pisati svoju funkciju za to).