Edit & Run

Stringovi u C programskom jeziku

- Stringovi u C programskom jeziku su zapravo nizovi karaktera
- Kraj stringa je označen posebnim karakterom, null karakterom
- Null karakter je karakter sa vrednošću 0, a ascii vrednost null karaktera može se pisati i kao ('\0')
- C nema “ugrađene metode za manipulaciju sa stringovima i uopšte za manipulaciju nizovima
- Jedina string-manipulišuća metoda u C-u je zapravo korišćenje string konstante

  String konstantu koristimo svaki put kada pišemo niz karaktera unutar dvostrukih navodnika

  ”ovo je primer stringa”

  Iza ovoga se zapravo krije činjenica da C u pozadini kreira niz karaktera umesto nas koji sadrže taj string terminiran sa null karakterom

  Na primer, možemo da definišemo niz karaktera i da ga inicijalizujemo korišćenjem string konstante:

  ```
  char string[] = "Hello, world!";
  ```

  U ovom slučaju možemo da izostavimo dimenziju niza, jer će kompajler uspešno izračunati dimenziju na osnovu dužine inicijalizacione string konstante. Ovo je JEDINI slučaj kada kompajler ovo radi za nas-u ostalim slučajevima programer mora odlučiti koliko je veliki niz u kome planiramo da čuvamo niz.
- Kako bismo uradili bilo šta druga sa stringom, moramo da pozivamo funkcije
- Postoji nekoliko osnovnih C funkcija za manipulisanje stringovima

STRCPY

Prva od njih je *strcpy*, koja je neophodna jer C ne dozvoljava dodelu vrednosti promenljivama tipa “string”.

```
#include <string.h>

char string1[] = "Hello, world!";
char string2[20];

strcpy(string2, string1);
```

String u koji se kopira je uvek prvi argument funkcije, tako da ova funkcija “mimikuje” dodelu vrednosti promenljivama tipa string.

Važno je primetiti da je neophodno napraviti string2 dovoljno veliki da može da primi sve karaktere stringa string1.

Takođe, na početku svakog fajla koji koristi funkcije koje rade sa stringovima, moramo da uključimo i

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

koji sadrži deklaracije funkcija koje koristimo.

STRCMP

Ova funkcija poredi dva stringa i vraća:

1. 0 ako su jednaki
2. negativan broj ako je prvi string “manji od” drugog stringa leksikografski
3. pozitivan broj u suprotnom

Primer:

```
char string3[] = "this is";
char string4[] = "a test";

if(strcmp(string3, string4) == 0)
        printf("strings are equal\n");
else    printf("strings are different\n");
```

Ne bi trebalo raditi nešto tipa

```
if(strcmp(string3, string4))
        ...
```

jer strcmp ne vraća boolean true/false. Ipak, često srećemo kod sličan ovome

```
if(strcmp(a, b))
```

ili čak

```
if(!strcmp(a, b))
```

Šta ovo radi?!?

STRCAT

```
char string5[20] = "Hello, ";
char string6[] = "world!";

printf("%s\n", string5);

strcat(string5, string6);

printf("%s\n", string5);
```

STRLEN

```
char string7[] = "abc";
int len = strlen(string7);
printf("%d\n", len);
```

Vraća dužinu stringa BEZ null karaktera!!

Kako bismo sami napisali funkciju kora radi kopiranje stringa slično kao strcpy???

A strcmp???

Na kraju, strlen?

Sve do sada pomenuto koristi standardnu C biblioteku za rad sa stringovima. Kako bismo napisali substr funkciju koja bi trebala da radi kao na primeru dole:

```
char string8[] = "this is a test";
char string9[10];
substr(string9, string8, 5, 4);
printf("%s\n", string9);
```

pri čemu želimo da “izdvojimo” sub-string dužine 4 počevši od karaktera 5 (kreće od 0 naravno) iz stringa string8 i kopiramo taj sub-string u string9.

String i karakteri

```
char string[] = "hello, world!";
```

Da li je opravdano da uradimo nešto kao

```
string[0] = 'H';
```

Da li bi moglo i ovako

```
string[0] = "H";
```

```
printf("\n");
```

ili

```
printf('\n');
```

Šta je sa

```
int i = '1';
```

Slično ovome, kao što '1' nije isto što i 1, string "123" nije jednak 123. Ako želimo da konvertujemo string u integer, za to koristimo funkciju atoi.

```
char string[] = "123";
int i = atoi(string);
int j = atoi("456");
```

Kako biste ovo uradili bez funkcije atoi?!?

Kako biste uradili kontra od ovoga- iz integera konvertovati u string?!?

Dodatne funkcije za rad sa stringovima su `strncpy`, `strncat`, i `strncmp`:

1)

```
char * strncpy ( char * destination, const char * source, size_t num );
```

Argumenti

`destination`

Pointer na odredišni niz u koji se kopira sadržaj

`source`

C string koji se kopira

`num`

Maksimalan broj karaktera koji se kopiraju iz *source*

Vraća vrednost *destination*

2)

```
char * strncat ( char * destination, const char * source, size_t num );
```

Argumenti

`destination`

odredište, dovoljno veliko da primi početni string zajedno sa dodatkom.

`source`

C string koji se dodaje

`num`

Maksimalan broj karaktera koji se dodaju

Vraća vrednost *destination*.

3)

```
int strncmp ( const char * str1, const char * str2, size_t num );
```

Argumenti

`str1`

C string koji se poredi.

`str2`

C string koji se poredi.

`num`

Maksimalan broj karaktera koji se porede

Vraća vrednost

return value	indicates
`<0`	prvi karakter koji se ne podudara ima “manju” vrednost u ptr1 nego u ptr2
`0`	jednaki stringovi
`>0`	prvi karakter koji se ne podudara ima “veću” vrednost u ptr1 nego u ptr2

4)

```
char * strchr (const  char * str, int character );
```

Pronalazi prvo pojavljivanje karaktera u stringu

Argumenti

`str`

C string.

`character`

Karakter koji se traži i interno se konvertuje u char

Vraća vrednost

pointer na prvo pojavljivanje u stringu, null ako ga nije bilo

Primer

```
/* strchr example */
#include <stdio.h>
#include <string.h>

int main ()
{
  char str[] = "This is a sample string";
  char * pch;
  printf ("Looking for the 's' character in \"%s\"...\n",str);
  pch=strchr(str,'s');
  while (pch!=NULL)
  {
    printf ("found at %d\n",pch-str+1);
    pch=strchr(pch+1,'s');
  }
  return 0;
}
```

Izlaz:

```
Looking for the 's' character in "This is a sample string"...
found at 4
found at 7
found at 11
found at 18
```

5)

```
char * strrchr (const char * str, int character );
```

Pronalazi poslednje pojavljivanje karaktera u stringu

Argumenti

`str`

C string.

`character`

Karakter koji se traži i interno se konvertuje u char

Vraća vrednost

pointer na poslednje pojavljivanje u stringu, null ako ga nije bilo

```c
/* strrchr example */
#include <stdio.h>
#include <string.h>

int main ()
{
  char str[] = "This is a sample string";
  char * pch;
  pch=strrchr(str,'s');
  printf ("Last occurence of 's' found at %d \n",pch-str+1);
  return 0;
}
```

Izlaz:

```
Last occurrence of 's' found at 18
```

6)

size_t strspn (const char * str1, const char * str2);

Pronalazi preklapanje dva stringa

Argumenti

`str1`

C string koji se traži.

`str2`

C string u kome se traži.

Vraća vrednost

Dužina inicijalnog dela str1 koji sadrži samo karaktere koji se nalaze u str2. Ukoliko su svi karakteri iz str1 sadržani u str2, funcija vraća dužinu stringa str1, a ukoliko prvi karakter iz str1 nije sadržan u str2 funkcija vraća 0.

Primer

```c
/* strspn example */
#include <stdio.h>
#include <string.h>

int main ()
{
  int i;
  char strtext[] = "129th";
  char cset[] = "1234567890";

  i = strspn (strtext,cset);
  printf ("The initial number has %d digits.\n",i);
  return 0;
}
```

Izlaz

```
The initial number has 3 digits.
```

7)

size_t strcspn (const char * str1, const char * str2);

Pronalazi “razlike” do preklapanja

Argumenti

`str1`

C string koji se traži.

`str2`

C string koji sadrži karaktere iz prvog.

Vraća vrednost

Dužinu inicijalnog dela str1 koji se ne sadrži nijedan karakter sadržan u okviru str2. Ovo je zapravo dužina str2 ukoliko nijedan od karaktera iz str2 se ne pojavljuje unutar str1

Primer

```
/* strcspn example */
#include <stdio.h>
#include <string.h>

int main ()
{
  char str[] = "fcba73";
  char keys[] = "1234567890";
  int i;
  i = strcspn (str,keys);
  printf ("The first number in str is at position %d.\n",i+1);
  return 0;
}
```

Izlaz:

```
The first number in str is at position 5
```

8)

char * strpbrk (const char * str1, const char * str2);

Locira karaktere u okviru stringa

Argumenti

`str1`

C string koji se traži.

`str2`

C string koji sadrži karaktere koji se poklapaju.

Vraća vrednost

Pointer na prvo pojavljivanje u str1 bilo koga karaktera koji je deo stringa str2, ili null pointer ukoliok se nijedan od karaktera iz str2 nalazi u str1 pre null-karaktera.

Primer

```
/* strpbrk example */
#include <stdio.h>
#include <string.h>

int main ()
{
  char str[] = "This is a sample string";
  char key[] = "aeiou";
  char * pch;
  printf ("Vowels in '%s': ",str);
  pch = strpbrk (str, key);
  while (pch != NULL)
  {
    printf ("%c " , *pch);
    pch = strpbrk (pch+1,key);
  }
  printf ("\n");
  return 0;
}
```

Izlaz:

```
Vowels in 'This is a sample string': i i a a e i
```

9)

char * strtok (char * str, const char * delimiters);

Deli string na “tokene”

Argumenti

`str`

C string koji se parsira. Obratiti pažnju da se string menja i zamenjuje manjim “tokenima”. Ukoliko je ovaj argument null pointer, funkcija nastavlja pretragu tamo gde je prethodni uspešan poziv funkcije završio.

`delimiters`

C string koji sadrži delimitere, koji mogu da se menjaju od jednog do drugog poziva.

Vraća vrednost

Pointer na početak tokena, ukoliko je pronađen

null pointer u suprotnom. Null pointer se takođe vraća i ako se došlo do kraja stringa (null karaktera) prilikom pretrage stringa.

Primer

```c
/* strtok example */
#include <stdio.h>
#include <string.h>

int main ()
{
  char str[] ="- This, a sample string.";
  char * pch;
  printf ("Splitting string \"%s\" into tokens:\n",str);
  pch = strtok (str," ,.-");
  while (pch != NULL)
  {
    printf ("%s\n",pch);
    pch = strtok (NULL, " ,.-");
  }
  return 0;
}
```

Izlaz:

```
Splitting string "- This, a sample string." into tokens:
This
a
sample
string
```