Edit & Run

Predavanje 2:

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

```
/* 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

```
/* 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

```
/* 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
```

Stringovi u C++ programskom jeziku

Za razliku od C programskog jezika, C++ programski jezik definiše klasu string za rad sa stringovima.

```
string testString;
testString = “Ovo je string.”;
```

što se naravno može spojiti u jedan izraz:

```
string testString = “Ovo je string.”;
```

Kada se koriste kao izlaz, cout prihvata promenljivu tipa string:

```
std::cout << testString << std::endl;
```

Kako bi se koristila klasa string, C++ header <string> mora biti uključen na početku programa. Takođe da se ne bi svaki put koristilo std::string, moguće je na početku dodati i:

```
#include <string>;
using namespace std;
```

Osnovne operacije

1. Broj karaktera stringa

Broj karaktera u stringu dobija se korišćenjem metode *length*. Primer:

```
#include <string>
#include <iostream>
using namespace std;
int main() {
	string small, large;
	small = "I am short";
	large = "I, friend, am a long and elaborate string indeed";
	cout << "The short string is " << small.length()  << " characters." <<
endl;
	cout << The long string is " << large.length()  << " characters." << endl;
	return 0;
}
```

Izlaz:

```
The short string is 10 characters.
The long string is 48 characters.
```

2. Pristup individualnim karakterima

Individualnim karakterima se može pristupiti korišćenjem uglastih zagrada (kao kod pristupanja elementima niza), pri čemu indeks (za string *str*) mora biti u opsegu 0 .. str.length() - 1. Na taj način se karakteri mogu čitati i upisivati.

```
#include <string>
#include <iostream>
using namespace std;

int main() {
	string test;
	test = "TestiRamo čitanj3 karaktera stringa";
	char ch = test[5]; // ch je 'R'
	test[16] = 'e'; // ispravljamo pogrešan karakter
	cout << test << endl;
	cout << "ch = " << ch << endl;
	return 0;
}
```

Izlaz:

```
TestiRamo čitanje karaktera stringa
ch = R
```

Alternativno, postoji metoda **at**(int *index*) koja vraća element stringa na poziciji index ali se znatno ređe koristi.

3. Prosleđivanje, vraćanje, dodeljivanje vrednosti

C++ stringovi su dizajnirani tako da se ponašaju kao primitivni tipovi sa stanovišta dodeljivanja vrednosti. Ako se vrednost jednog stringa dodeljuje drugom, dolazi do kopiranja sekvence karaktera:

```
string str1 = "hello";
string str2 = str1; // makes a new copy
str1[0] = 'y'; // changes str1, but not str2
```

Prosleđivanje stringova i vraćanje stringova kao povratne vrednosti fukcija klonira string – ako se prosledi string parametar unutar funkcije kojoj je prosleđen, promene se ne vide u funkciji koja je pozvala datu funkciju, osim ako je string prosleđen preko reference.

4. Poređenje stringova

Stringove možete porediti korišćenjem operatora ==, !=, <, <=, >, >=. Ovi operatori porede stringove leksikografski, karakter po karakter, pri čemu se vodi računa o velikim i malim slovima.

```
#include <string>
#include <iostream>
using namespace std;

int main() {
	string myName = "Pedja";
	while (true) {
		cout << "Enter your name (or 'quit' to exit): ";
		string userName;
		getline(cin, userName);
		if (userName == "Marko") {
			cout << "Hi, Marko! Welcome back!" << endl;
		} else if (userName == "quit") {
			// izlazak iz programa
			cout << endl;
			break;
		} else if (userName != myName) {
		// user did not enter quit, Marko, or Pedja
			cout << "Hello, " << userName << endl;
		} else {
			cout << "Oh, it’s you, " << myName << endl;
		}
	}
	return 0;
}
```

Izlaz:

```
Enter your name (or 'quit' to exit): Pedja
Oh, it's you, Pedja
```

```
Enter your name (or 'quit' to exit): Marko
Hi, Marko! Welcome back!
Enter your name (or 'quit' to exit): Vlada
Hello, Vlada
Enter your name (or 'quit' to exit): quit
```

5. Konkatenacija stringova

Za konkatenaciju stringova koristimo operaciju +, ili += ako želimo da dodamo elemente na kraj već postojećeg stringa:

```
string str = “Hello”;
str += “ world”;
```

Primer:

```
#include <string>
#include <iostream>
using namespace std;
int main() {
	string firstname = "Marko";
	string lastname = " Markovic";
	string fullname = firstname + lastname; // concat the two strings
	fullname += ", senior programer"; // append another string
	fullname += '.'; // append a single char
	cout << firstname << lastname << endl;
	cout << fullname << endl;
	return 0;
}
```

Izlaz:

```
Marko Markovic
Marko Markovic, senior programer.
```

6. Pretraživanje u stringu

Koristi se metoda find kojoj je prosleđuje string (ili karakter) koji se traži. Povratna vrednost je ili početna pozicija na kojoj je pronađen traženi string (ili karakter) ili konstanta **string::npos** koja daje informaciju da nije pronađen.
Takođe, moguće je proslediti još jedan celobrojni podatak koji govori odakle da se krene sa pretragom:

```
#include <string>
#include <iostream>
using namespace std;
int main() {
	string sentence = "Yes, we went to Gates after we left the dorm.";
	int firstWe = sentence.find("we"); // finds the first "we"
	int secondWe = sentence.find("we", firstWe + 1); // finds "we" in "went"
	int thirdWe = sentence.find("we", secondWe + 1); // finds the last "we"
```

```
	int gPos = sentence.find('G');
	int zPos = sentence.find('Z'); // returns string::npos
	cout << "First we: " << firstWe << endl;
	cout << "Second we: " << secondWe << endl;
	cout << "Third we: " << thirdWe << endl;
	cout << "Is G there? ";
	cout << (gPos != string::npos ? "Yes!" : "No!") << endl;
	cout << "Is Z there? ";
	cout << (zPos != string::npos ? "Yes!" : "No!") << endl;
	return 0;
}
```

Izlaz:

```
First we: 5
Second we: 8
Third we: 28
Is G there? Yes!
Is Z there? No!
```

7. Podstringovi

Pravljenje novog stringa koristeći podstring već postojećeg string *str* radi se pomoću:

```
str.substr(start, length)
```

koja vraća novi string koji se sastoji od karaktera iz stringa *str* počevši od indeksa *start* i ima dužinu *length*. String *str* se ne menja, jer se pravi kopija stringa sa specificiranim karakterima. Ukoliko se drugi argument (length) ne prosledi, ide se do kraja ulaznog stringa. Primer:

```
#include <string>
#include <iostream>
using namespace std;

int main() {
	string oldSentence;
	oldSentence = "The quick brown fox jumped WAY over the lazy dog";
	int len = oldSentence.length();
	cout << "Original sentence: " << oldSentence << endl;
	int found = oldSentence.find("WAY ");
	string newSentence = oldSentence.substr(0, found);
	cout << "Modified sentence: " << newSentence << endl;
	newSentence += oldSentence.substr(found + 4);
	cout << "Completed sentence: " << newSentence << endl;
	return 0;
}
```

Izlaz:

```
Original sentence: The quick brown fox jumped WAY over
the lazy dog
Modified sentence: The quick brown fox jumped
Completed sentence: The quick brown fox jumped over the
```

```
lazy dog
```

Ukoliko je *start* argument veći od dužine stringa, vraćen je prazan string, ako je *length* veće od broja preostalih karaktera u stringu počevši od *start*, vraća se manji broj karaktera od length (onoliko koliko ih ima do kraja stringa počevši od *length*). Oba parametra moraju da budu pozitivni.

8. Dodavanje ili zamena karaktera u stringu

```
str1.insert(start, str2)
```

dodaje string *str2* na poziciju *start* u stringu *str1*, pomerajući preostale karaktere.

```
str1.replace(start, length, str2)
```

uklanja iz *str1* ukupno length karaktera počevši od pozicije *start*, zamenjujući ih sa kopijom stringa *str2*. Obe metode menjaju string *str1*.

```
#include <string>
#include <iostream>
using namespace std;

int main() {
      string sentence = "RSZES sucks.";
      cout << sentence << endl;
      // Insert "kind of" at position 7 in sentence
      sentence.insert(6, "kind of ");
      cout << sentence << endl;
      // Replace the 10 characters "kind of su"
      // with the string "ro" in sentence
      sentence.replace(6, 10, "ro");
      cout << sentence << endl;
      return 0;
}
```

Izlaz:

```
RSZES sucks.
RSZES kind of sucks.
RSZES rocks.
```

9. Dobijanje C stringa (char *) od klase string

Iako možete istovremeno koristiti nizove karaktera terminirane sa ‘\0’ i std::string u vašem programu, uglavnom se koristi std::string pošto obezbeđuje znatno veći skup operacija koje nad njima mogu da se vrše. Ipak, u nekim situacijama neophodno je koristiti C stringove (npr za funkcije koje zahtevaju kao argumente niz karaktera terminiran sa ‘\0’) i tada se koristi metoda

```
str.c_str()
```

kako bi se od stringa str dobio ekvivalentan C string. Primer:

```
#include <string>
#include <iostream>
#include <fstream>
using namespace std;


int main() {
      ifstream fs;
      string filename = "courseinfo.txt";
      string s;
      // open function requires a C-style string, must convert!
      fs.open(filename.c_str());
      if (fs.fail())
            return -1; // could not open the file!
      // process the file
      fs.close();
      return 0;
}
```

10. Korišćenje svih ostalih C funkcija za manipulaciju stringova

Nakon što se doda

```
#include <cstring>
```

sve funkcije za rad sa C stringovima navedene ranije mogu da se koriste i u C++ programima. Na primer:

```
#include <cstring>
#include <iostream>

int main()
{
    char input[100] = "A bird came down the walk";
    char *token = std::strtok(input, " ");
    while (token != NULL) {
        std::cout << token << '\n';
        token = std::strtok(NULL, " ");
    }
}
```

Izlaz:

```
A
bird
came
```

```
down
the
walk
```