Универзитет “Св. Кирил и Методиј” во Скопје
Факултет за електротехника и информациски технологии

ПРОГРАМИРАЊЕ И АЛГОРИТМИ

ТЕКСТУАЛНИ НИЗИ

- Програмски јазик C –

ТЕКСТУАЛНИ НИЗИ-ВОВЕД

- Многу јазици имаат стандардно дефиниран податочен тип текстуална низа (стринг)
  - Basic, Turbo Pascal, Scheme/Lisp, Java

- Текстуални низи во програмскиот јазик C
  - Не е дефиниран стандарден податочен тип текстуална низа.
  - Според конвенцијата текстуалните низи се сместуваат во низи од знаци.
  - Според конвенцијата, крајот на текстуалната низа се означува со NULL знакот (знак со код 0).

ДЕКЛАРАЦИЈА НА ТЕКСТУАЛНА НИЗА

- За да се декларира текстуална низа:
  - се декларира вектор од знаци, или
  - за низата да се декларира покажувач

- Формат:

```
#define Broj 100
char Niza[Broj];
char *niza;
```

ДЕКЛАРАЦИЈА И ИНИЦИЈАЛИЗАЦИЈА

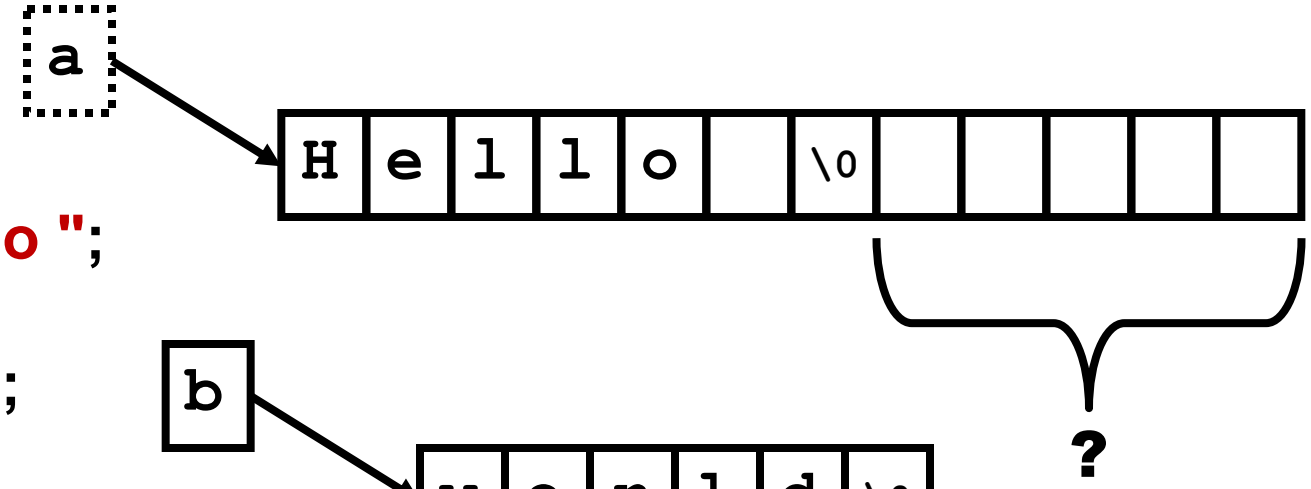
- Примери:

```
char s1[10];
char *s2;
char s3[10] = "foo";
char *s4 = "bar";
char s5[10] = "foo";

char a[]="hello\n";
char* b="hello\n";
```

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
*b	*(b+1)	*(b+2)	*(b+3)	*(b+4)	*(b+5)	*(b+6)
h	e	l	l	o	\n	null
104	101	108	108	111	10	0

ПРИМЕР:

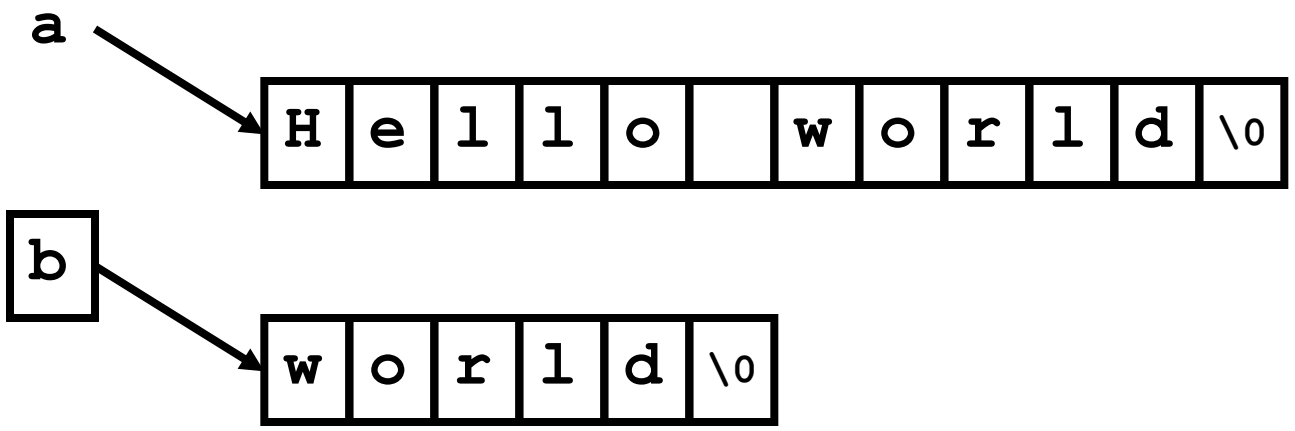



```
char a[12] = "Hello ";
```

```
char *b = "world";
```



```
strcat(a, b);
```




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

```
Hello world
```

ПРАШАЊЕ:

Што се случува ако низата не завршува со NULL?

```
char s1[10] = "foo";
printf("%s\n", s1);
```

Ако се изврши претходниот код, тогаш на компјутерскиот екран ќе биде прикажано

```
foo
```

КОМЕНТАР:

- Ако **s1** не завршува со знакот **null** тогаш наредбата **printf** ќе ги прикаже сите знаци додека не наиде на знак **null** или програмата ќе заврши со порака за грешка (*segment fault etc.*).

- **Пример:**

```
char s1[10];
s1[0] = 'f';
s1[1] = 'o';
s1[2] = 'o';
printf("%s\n", s1);
```

ПРАШАЊЕ:
Како се однесува char s[3]=“foo” ?
Нема да има место за **null** !

foo**tk?**□

ПРИМЕРИ

```
char s1[10] = "foo";
char *s2 = "bar";
```

Дали следната наредба е исправна?

```
s2 = s1;
```

ДА!

```
char s1[10] = "foo";
char *s2 = "bar";
```

Дали следната наредба е исправна?

```
s1 = s2;
```

НЕ!

ПРИМЕРИ

```
char s1[10] = "foo";
char *s2 = "bar";
```

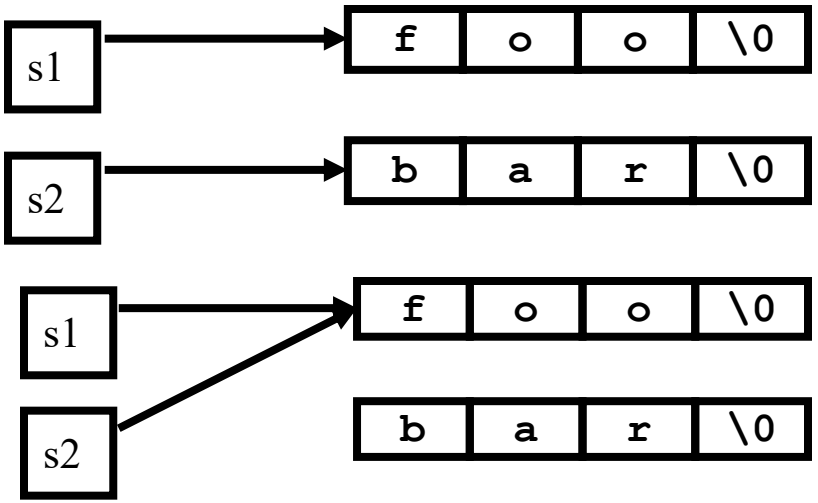
- Дали наредбата **s2 = s1** овозможува копирање на содржината на текстуалните низи?

  НЕ!!!!!!

- Сепак изгледа како да била извршена операцијата копирање

```
char s1[10] = "foo";
char *s2 = "bar";

s2 = s1;
printf("%s\n", s1);
printf("%s\n", s2);
```



Што печати програмата?

```
foo
foo
```

ВНЕСУВАЊЕ НА ТЕКСТУАЛНИ НИЗИ

- Со користење на наредбата scanf()
  - Формат: **scanf(”%s”, str);**
  - не е потребно да се користи адресниот оператор **&**.
  - **scanf()** запира на првото празно место, потоа ги презема сите знаци пред истото и ги сместува во **str** (доколку нема празно место запира на знак за нов ред).
  - **scanf()** секогаш додава **'\0'** на крајот од низата.
  - текстуална низа внесена со **scanf()** никогаш нема да содржи знак за празно место.

ВНЕСУВАЊЕ НА ТЕКСТУАЛНИ НИЗИ

Пример:

```
#include <stdio.h>
        int main(){
                char s[10];
                scanf("%s",s);   /* bad */
                scanf("%9s",s); /* good */
                printf("%s",s);
                return 0;
        }
```

ВНЕСУВАЊЕ НА ТЕКСТУАЛНИ НИЗИ СО getchar()

```
#include <stdio.h>
#define N 10
int main(){
  char ch,str[N];
  int i=0;

  while((i<N-1) &&(ch=getchar())!='\n')
  {
       str[i]=ch;
       i++;
  }
  str[i]='\0';
  printf("%s\n",str);
  return 0;
}
```

Зошто?

За да има место за ‘\0’

ВНЕСУВАЊЕ НА ТЕКСТУАЛНИ НИЗИ СО gets()

- Формат:

```
char *gets(char *s);
```

- чита линија внесена од стандардниот влезен уред во бафер (меморискиот простор) кон кој покажува **s** се додека не наиде на ознака за нов ред или **EOF**, што ги заменува со **'\0'**.

- **не** ги прескокнува празните места

- Функцијата **не проверува** дали меморискиот простор е доволен за сместување на сите знаци внесени од тастатура!

ВНЕСУВАЊЕ НА ТЕКСТУАЛНИ НИЗИ СО gets()

Пример:

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

char buffer[10];

int main(){
        gets(buffer);
        printf("buffer = %s\n", buffer);
        return 0;
}
```

```
C:\>demo
hello
buffer = hello
```

It stops when either the newline character (\n) is read or when the end-of-file is reached, whichever comes first.

```
warning: this program uses gets(), which is unsafe.
abcdsaldjalskjdaklsdjklsajdklsa
buffer = abcdsaldjalskjdaklsdjklsajdklsa
```

ПРИКАЗ НА ТЕКСТУАЛНИ НИЗИ

- Со функцијата puts()
  - Пример:

```
char str[]="Hello World!";
puts(str); /*dodava oznaka \n po str*/
```

- Со функцијата printf()
  - Пример:

```
printf("%s",str);
```

ОПЕРАЦИИ СО ТЕКСТУАЛНИ НИЗИ

- Датотеката **string.h** ги содржи заглавјата на функциите што овозможуваат работа со текстуални низи:

- `int strcmp(const char *string1, const char *string2)`
  - споредба на string1 и string2
  - if Return value < 0 then it indicates str1 is less than str2.
  - if Return value > 0 then it indicates str2 is less than str1.
  - if Return value = 0 then it indicates str1 is equal to str2.

- `int strncmp(const char *string1, const char *string2, size_t n)`
  - споредба на string1 и string2 до првите n знаци.
  - if Return value < 0 then it indicates str1 is less than str2.
  - if Return value > 0 then it indicates str2 is less than str1.
  - if Return value = 0 then it indicates str1 is equal to str2.

ОПЕРАЦИИ СО ТЕКСТУАЛНИ НИЗИ

- **strcmp(a,b)**: функцијата ги споредува низите a и b, и враќа вредност:
  - 0 ако низите се **еднакви**,
  - >0 ако првата низа е ,,**поголема**” од втората, и
  - <0 ако првата низа е ,,**помала**” од втората.

```
strcmp("abc", "abd") == -1;
strcmp("abf","aba") == 1;
strcmp("abc","abc") == 0
```

ОПЕРАЦИИ СО ТЕКСТУАЛНИ НИЗИ

- Датотеката **string.h** ги содржи заглавјата на функциите што овозможуваат работа со текстуални низи:

- `char *strcpy(char *string1,const char *string2)`
  - копирање на содржината на string2 во string1.
  - returns a pointer to the destination string1t.

- `char *strncpy(char *string1,const char *string2, size_t n)`
  - копирање на содржината на првите n знаци од string2 во string1. Останатите знаци од string1 си остануваат.
  - returns the final copy of the copied string

```
char s1[10] = "foo";
char *s2 = "foobar";
char *s3 = ”abc";
```

```
strcpy(s1, s2); // s1 = “foobar”
strncpy(s2, s3, 3); // s2 = “abcbar”
```

ОПЕРАЦИИ СО ТЕКСТУАЛНИ НИЗИ

- **strcpy(a,b)**: ја копира содржината на низата **b** во **a**, притоа не проверувајќи дали **a** има место за сместување на **b**

- Пример:

```
char s1[10] = "foo";
char *s2 = "foobar";
strcpy(s2, s1);
strcpy(s1, s2);
```

ОПЕРАЦИИ СО ТЕКСТУАЛНИ НИЗИ

- Датотеката **string.h** ги содржи заглавјата на функциите што овозможуваат работа со текстуални низи:

- `size_t strlen(const char *string)`
  - наоѓање на должината на текстуалната низа string.
  - returns the length of string

- `char *strcat(char *string1, const char *string2)`
  - конкатенација (додавање, прилепување) на string2 кон string1.
  - returns a pointer to the resulting string1

- `char *strncat(char *string1, const char *string2, size_t n)`
  - конкатенација на n знаци од string2 на string1.
  - returns a pointer to the resulting string1

ОПЕРАЦИИ СО ТЕКСТУАЛНИ НИЗИ

- **strlen(a)**: функцијата враќа број на знаци во низата **a**

Пример за користење:

```
length = strlen("HELLO"); /* length = 5 */
```

Реализација:

```
size_t strlen(const char *s){
        size_t n;
        for(n=0;*s!='\0';s++)n++;
        return n;
}

size_t strlen(const char *s){
        const char *p=s;
        while (*s) s++;
        return s-p;
}
```

ОПЕРАЦИИ СО ТЕКСТУАЛНИ НИЗИ

- **Реализација:**

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

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

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

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

ОПЕРАЦИИ СО ТЕКСТУАЛНИ НИЗИ

- **strcat(a,b)**: ја додава содржината на низата **b** на крајот од низата **a**

Реализација:

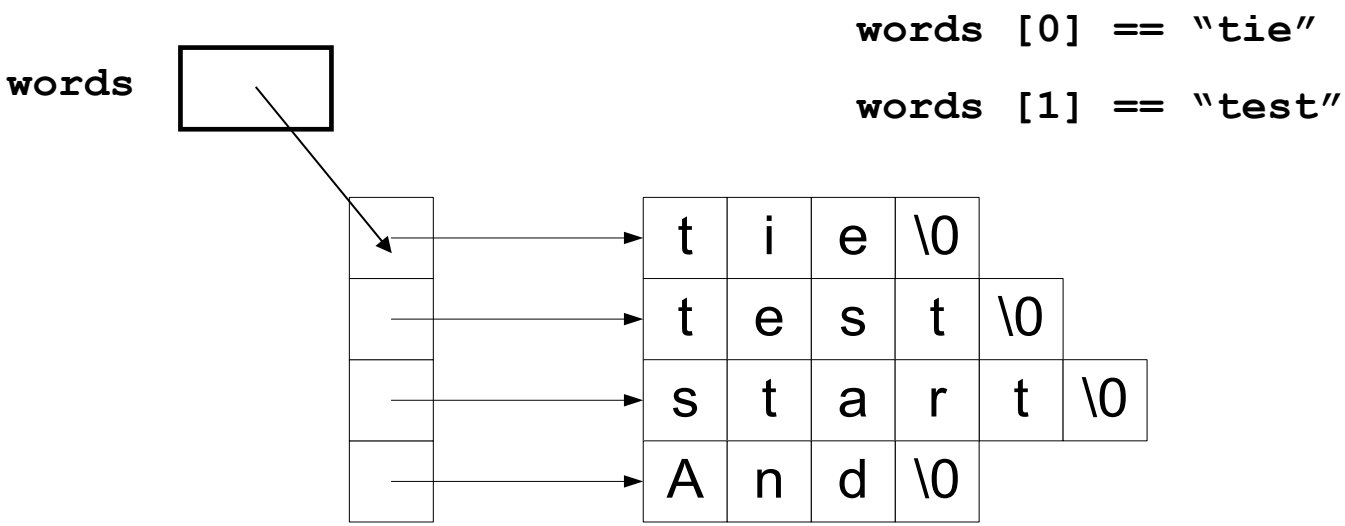
```
char * strcat(char *s1, const char *s2){
    char *p=s1;
    while(*p) p++;   // while (*p!='/0')

    while (*p++=*s2++);

    return s1;
}
```

ВЕКТОРИ ОД ТЕКСТУАЛНИ НИЗИ

```
char *words[]={“tie”, “test”, ”start”, ”And”}
char words[][]={“tie”, “test”, ”start”, ”And”}
char **words={“tie”, “test”, ”start”, ”And”}
```

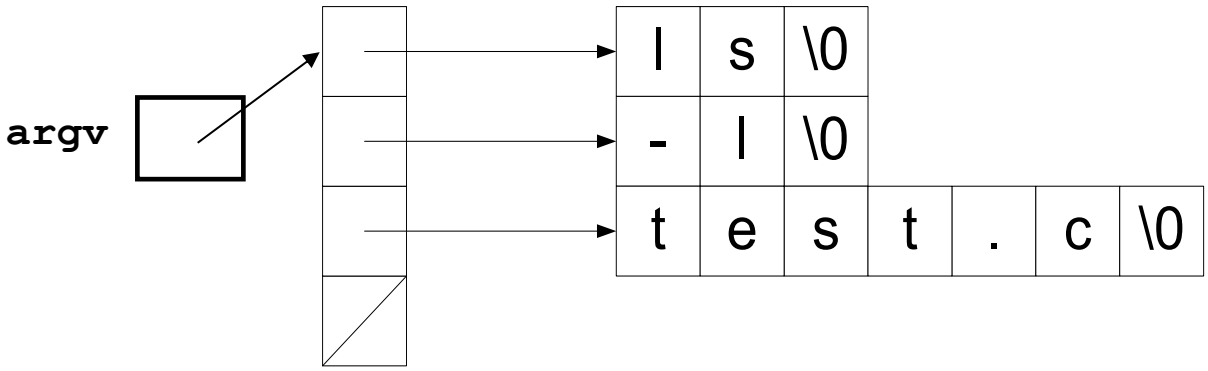


АРГУМЕНТИ ВО main ФУНКЦИЈАТА

- Често во литературата може да се сретнете со следната конструкција:

```
#include <stdio.h>
int main(int argc, char **argv){
printf("Hello, world!\n");
return 0;
}
```

Во командна линија
ls -l test.c



- Аргументите во main функцијата претставуваат едноставен начин за пренесување на информација во програмата (имиња на датотеки, опции, итн.)
- **argc** се однесува на бројот на аргументи
  - цел број што го содржи бројот на аргументи што се пренесуваат во програмата
- **argv** претставува вектор од текстуални низи
  - првиот елемент на овој вектор е името на програмата

КОРИСТЕЊЕ НА АРГУМЕНТИТЕ ОД КОМАНДНА ЛИНИЈА

```
#include <stdio.h>
int main(int argc, char **argv) {
        int i;
        for(i = 0; i < argc; i++)
                printf("argv[%d]=%s\n",i,argv[i]);
        return 0;
}
```

- Телото на циклусот се повторува за секој аргумент од командната линија

Излез од програмата:

```
> gcc hello.c –o hello
> hello how are you today?
argv[0]=hello
argv[1]=how
argv[2]=are
argv[3]=you
argv[4]=today?
```

Во зависност од компајлерот:

./hello how are you today?

или

./a.out hello how are you today?

КРАЈ