



Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Низи од знаци

Структурно програмирање

ФИНКИ 2013

Текстуални низи вовед

- Многу јазици имаат стандардно дефиниран податочен вид текстуална низа (стринг)
 - Типични примери вклучуваат: Basic, Turbo Pascal, Scheme/Lisp, Java
 - Тоа значи дека сите детали околу користењето на низите се имплементирани во компајлерот и извршната околина
- Текстуални низи во C
 - Во програмскиот јазик C не е дефиниран стандарден податочен тип текстуална низа.
 - Постои договорна конвенција текстуалните низи да се сместуваат во низи од знаци.
 - Според конвенцијата крајот на текстуалната низа се означува со NULL терминатор (знак со код 0).

Декларација и иницијализација

- За да се декларира текстуална низа потребно е да се декларира вектор од знаци, или за низата да се декларира покажувач
 - формат:
`char Niza[Broj];`
`char *niza;`
- Работата со низи од знаци во C се сведува на работа со покажувачи на NULL терминирани низи од знаци
- Во стандардната библиотека `string.h` се дадени голем број на готови функции за работа со текстуални низи

Декларација и иницијализација

- Пример за декларирање и иницијализација на текстуална низа

```
char s1[10];
char *s2;
char s3[10] = "foo";
char s5[5] = "foo";
char a[]="hello\n";
const char* b="hello\n";
char *s4 = s3;
```

```
char *x="abcd";
x[1]='q';
Нема да работи!!!
```

a[0] *b	a[1] *(b+1)	a[2] *(b+2)	a[3] *(b+3)	a[4] *(b+4)	a[5] *(b+5)	a[6] *(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);
```

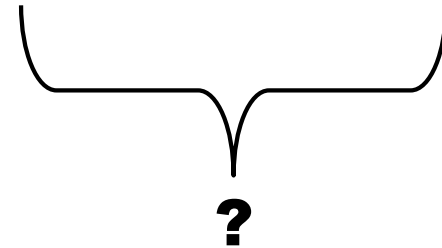
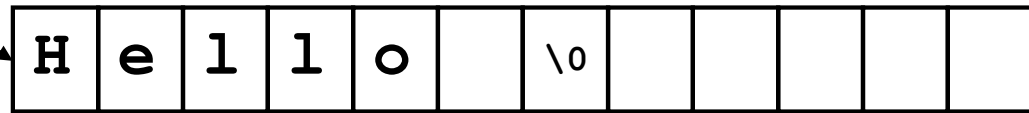
```
printf("%s\n", a);
```

Пример

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

a

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



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

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

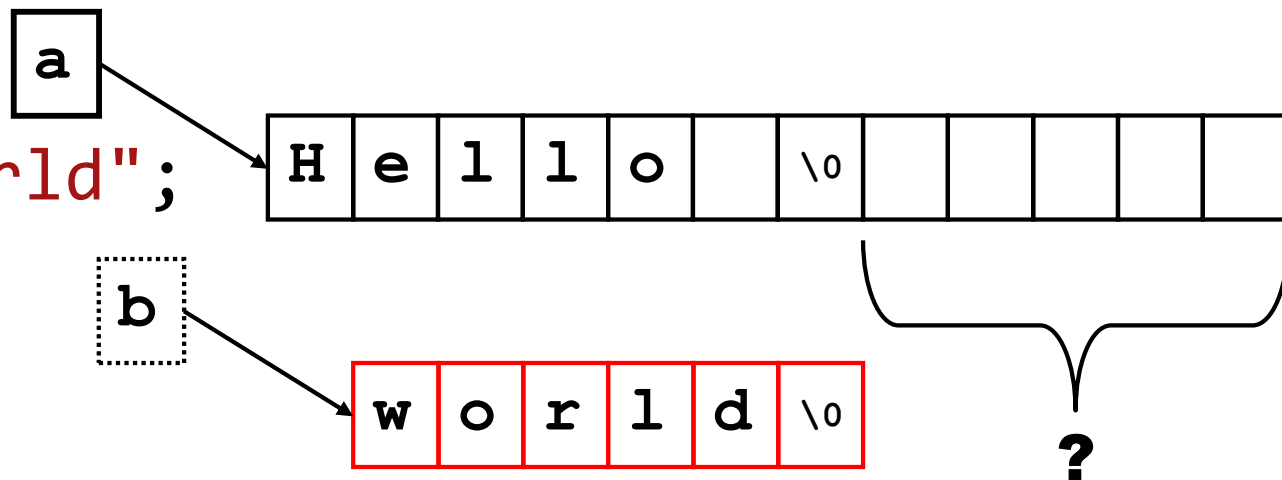
Пример

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

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

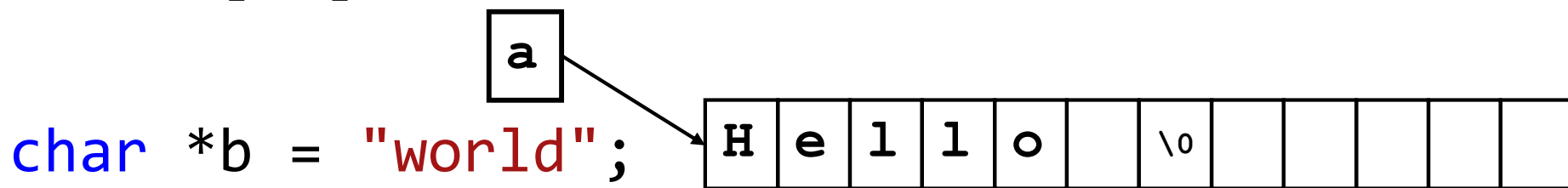
```
strcat(a, b);
```

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

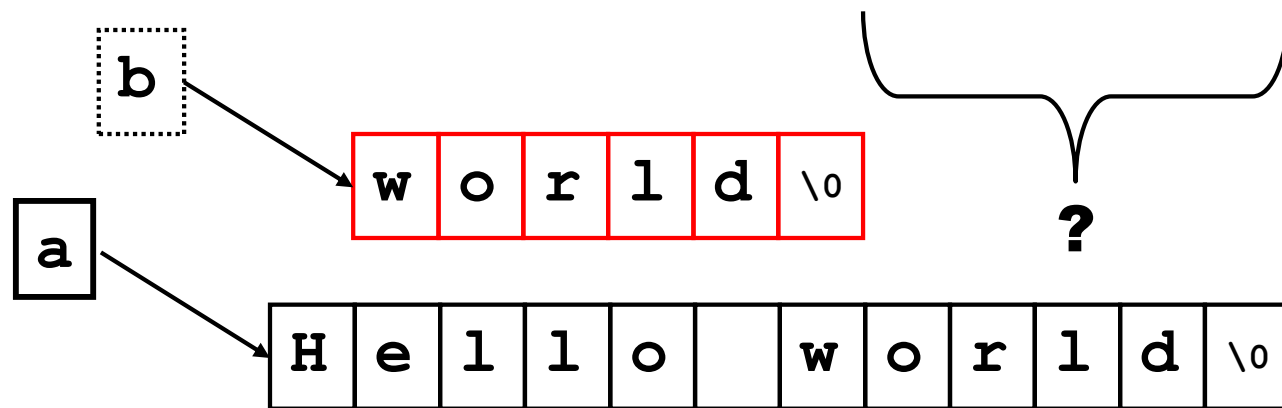


Пример

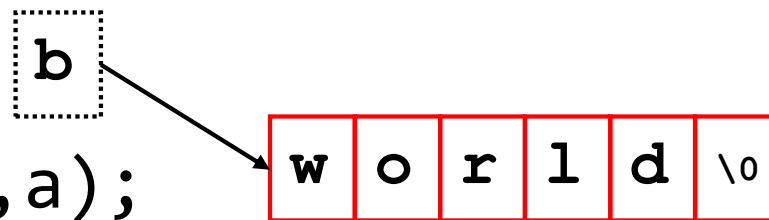
```
char a[12] = "Hello ";
```



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



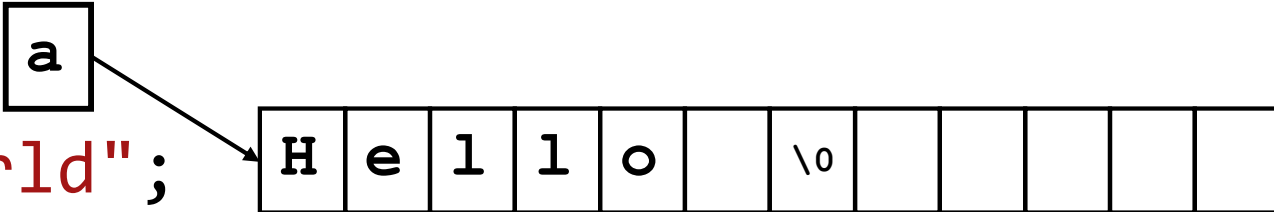
```
printf("%s\n", a);
```



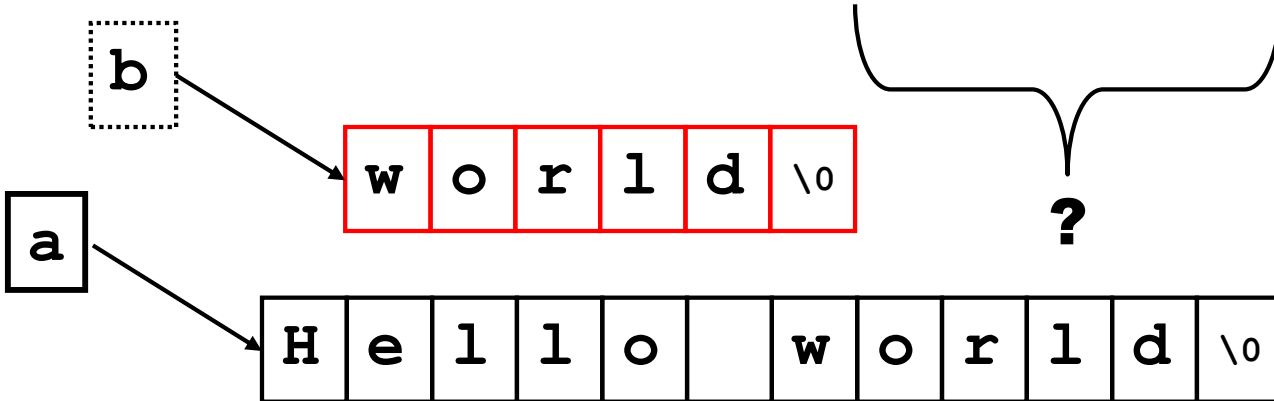
Пример

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

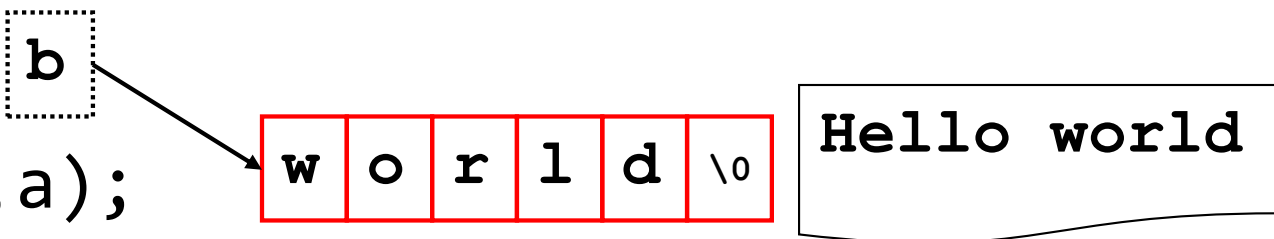
```
char *b = "world";
```



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



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



NULL терминатор

- Што се случува ако низата не завршува со NULL терминатор?

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

Автоматски се
додава '\0' на крајот

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

foo

NULL терминатор

Забелешка:

- Ако `s1` не завршува со знакот `NULL` тогаш наредбата `printf` ќе ги прикаже сите знаци додека не најде 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);
```

Sample output

```
fooâ-•VGot 2  
,_VGot 2  
Got 20  
_VGot 2  
_Got 16  
Got 1
```

ПРАШАЊЕ:

Како се однесува `char s[3]="foo"`?
Нема да има место за `null`!

Текстуалните низи се покажувачи

```
char s1[10] = "foo";
```

```
char *s2 = "bar";
```

- Дали може да се изведе следното доделување?

```
s2 = s1;
```

ДА

- Дали може да се изведе следното доделување?

```
s1 = s2;
```

НЕ

Копирање на текстуални низи

```
char s1[10] = "foo";
```

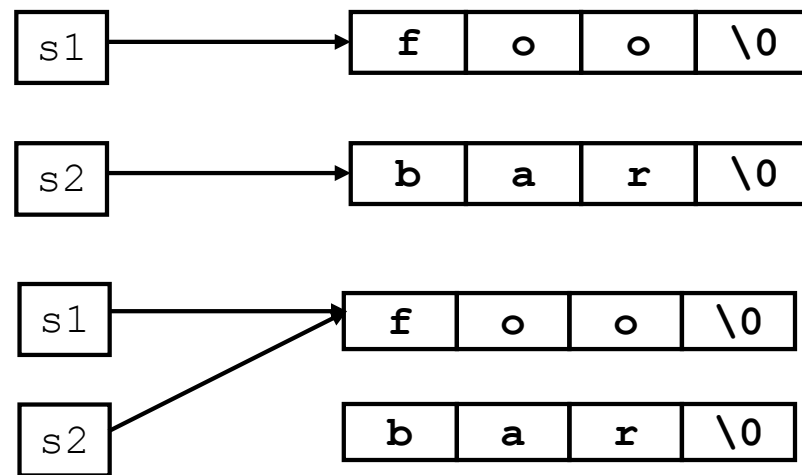
```
char *s2 = "bar";
```

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

НЕ

- Но тоа сепак изгледа како да била извршена операцијата копирање
- Ако се печатат `s1` и `s2` се добива:

```
foo
foo
```



Внесување на текстуални низи

- Со користење на наредбата `scanf()`

- формат

- `scanf("%s", str);`

- не е потребно да се користи адресниот оператор `&`.
- `scanf()` ги пресконува празните места, потоа ги презема сите знаци и ги сместува во `str` се додека не најде на празно место.
- `scanf()` секогаш додава `\0` на крајот од низата.
- текстуална низа внесена со `scanf` никогаш нема да содржи знак за празно место.

%s и безбедност

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

- Користење на %s во scanf() (fscanf(), sscanf(), ...) не се препорачува од безбедносни причини
- GNU specific (не е C стандард) воведен модификатор на %s форматот %as кој автоматски ја алоцира потребната меморија
- Microsoft specific (не е C стандард) воведени функции scanf_s(), ...

Внесување на текстуални низи со getchar()

```
#define N 10
char ch, str[N];
int i=0;
while((i<N-1)&&(ch=getchar())!='\n')
    str[i++]=ch;
str[i]='\0';
```

- Ги чита и празните места се додека не се притисне ENTER или додека не се прочитаат потребниот број знаци (9)

Внесување на текстуални низи

co gets()

- Наредба gets

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

- gets() чита ред внесен од стандардниот влезен уред во бафер (мемориски простор) кон кој покажува s сè додека не најде на ознака за нов ред или EOF, што ги заменува со '\0'.
- не ги прескокнува празните места
- наредбата **не проверува дали меморискиот простор е доволен** за сместување на сите знаци внесени од тастатура
- Користењето на gets() не се препорачува од безбедносни причини

Операции за внесување на текстуални низи

```
#include <stdio.h>
char buffer[10];
int main(){
    gets(buffer);
    printf("buffer = %s\n", buffer);
    return 0;
}
```

ДОБРО ОДНЕСУВАЊЕ

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

ЛОШО ОДНЕСУВАЊЕ

```
C:\>demo
Now is the time for all good men to come to the aid of their
buffer = Now is the time for all good men to come to the aid of their
Segmentation fault

Process returned -1073741819 (0xc0000005)    execution time : 8.141 s
```

Приказ на текстуални низи

■ со наредбата puts()

□ пример

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

■ со наредбата printf()

□ пример

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

Проблеми со форматирано %s читање

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

int main()
{
    char *str1, str2[20];
    str1 = (char *) malloc (10);

    puts ("Vnesete niza od 9 znaci ili pomalku.");
    scanf ("%9s%*[^\\n]*%c", str1);
    printf ("Ja vnesovte nizata:%s\\n", str1);

    puts ("\\nVnesete nova niza.");
    scanf ("%19s", str2);
    printf ("Potoa ja vnesovte nizata:%s\\n", str2);
    free(str1);
    return 0;
}
```

Vnesete niza od 9 znaci ili pomalku.
12345↵
Ja vnesovte nizata:12345

Vnesete nova niza.
qwerty↵
Potoa ja vnesovte nizata:qwerty

Vnesete niza od 9 znaci ili pomalku.
123456789xyz↵
Ja vnesovte nizata:123456789

Vnesete nova niza.
Potoa ja vnesovte nizata:xyz

Vnesete niza od 9 znaci ili pomalku.
123456789abcd↵
Ja vnesovte nizata:123456789

Vnesete nova niza.
qwerty↵
Potoa ja vnesovte nizata:qwerty

Ќе бидат
игнорирани

Операции со текстуални низи

Датотеката `string.h` ги содржи заглавијата на функциите што овозможуваат работа со текстуални низи

- **strlen(a)**: функцијата враќа број на знаци во низата `a`
- **strcmp(a,b)**: функцијата ги споредува низите `a` и `b`, и враќа вредноста 0 ако низите се еднакви, 1 ако првата низа е „поголема“ од втората, -1 ако првата низа е „помала“ од втората.

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

```
strcmp("abf", "aba") == 1;
```

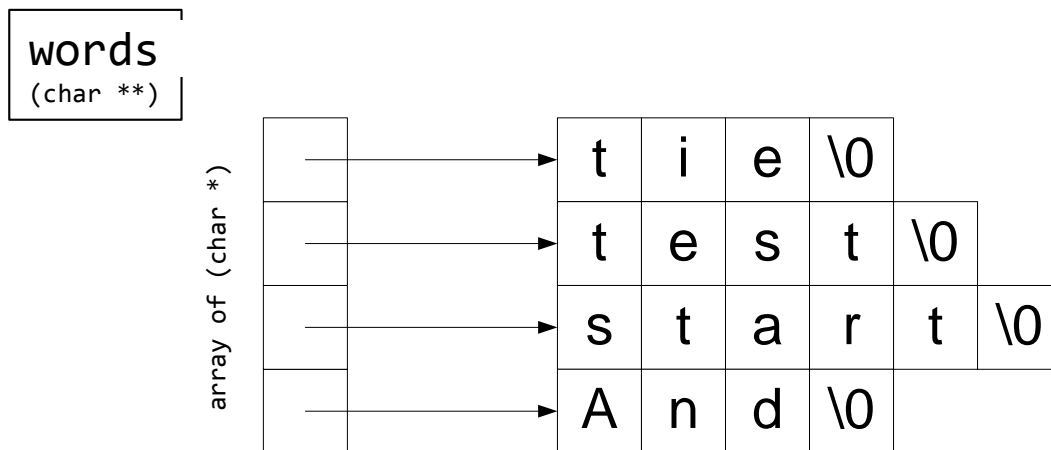
```
strcmp("abc", "abc") == 0;
```

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

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

Вектори од текстуални низи

```
char *words[]={ "tie", "test", "start", "And"};
```



Аргументи во main функцијата

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

```
int main(int argc, char **argv)
```

```
{
```

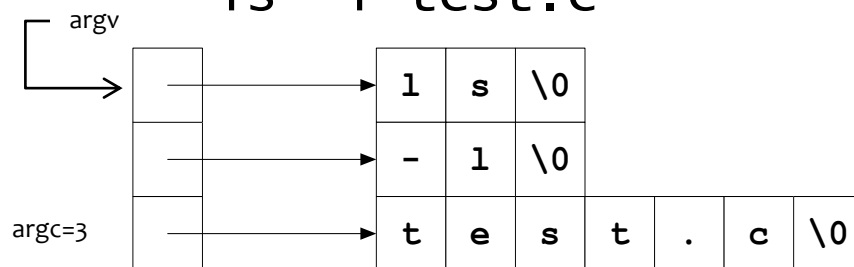
```
    printf("Hello, world!\n");
```

```
    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("%s\n", 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?
hello
how
are
you
today?
```

```
argc=5
argv[0]="hello"
argv[1]="how"
...
argv[4]="today?"
```


string.h

String manipulation

- [strcpy](#) - copies one string to another
- [strncpy](#) - writes exactly n bytes to a string, copying from src or adding nulls
- [strcat](#) - appends one string to another
- [strncat](#) - appends no more than n bytes from one string to another

String examination

- [strlen](#) - returns the length of a string
- [strcmp](#) - compares two strings
- [strncmp](#) - compares a specific number of bytes in two strings
- [strchr](#) - finds the first occurrence of a byte in a string
- [strrchr](#) - finds the last occurrence of a byte in a string
- [strspn](#) - finds in a string the first occurrence of a byte not in a set of bytes
- [strcspn](#) - finds in a string the last occurrence of a byte not in a set of bytes
- [strpbrk](#) - finds in a string the first occurrence of a byte in a set of bytes
- [strstr](#) - finds in a string the first occurrence of a substring
- [strtok](#) - finds in a string the next occurrence of a token

Memory manipulation

- [memset](#) - fills a buffer with a repeated byte

Numeric conversions in the stdlib.h

- [atof](#) - converts a string to a floating-point value
- [atoi](#) - converts a string to an integer

Имплементација на функциите во string.h

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

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

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

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

Имплементација на функциите во string.h

```
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;
}
```

```
char *strcat(char *s1, const char *s2){
    char *p=s1;
    while(*p) p++;
    while (*p++=*s2++);
    return s1;
}
```

Прашања?