

# Низове

доц. д-р Нора Ангелова

---

# Низове

## Логическо описание

Крайна или празна редица от символи, заградени в кавички.

Пример:

```
"abc"; "";
```

## *Дефиниция*

Броят на символите в редицата се нарича **дължина** на низа.

Пример:

```
"abc" // Низ с дължина 3
```

## *Дефиниция*

Низ с дължина 0, се нарича **празен**.

Пример:

```
"" // Празен низ
```

# Низ != масив от символи

- "abc" vs 'a', 'b', 'c'

# Низове

## *Дефиниция*

Низ, който се съдържа в даден низ, се нарича негов **подниз**.

Пример:

"a1b2c3"

"b2"

// Подниз на "a1b2c3"

## *Дефиниция*

**Конкатенация** на два низа е низ, получен като в края на първия низ се запише вторият.

Пример:

"1\*a"; "=a\*1";

"1\*a=a\*1"; // Конкатенация на низовете "1\*a" и "=a\*1"

# Низове

## *Сравнение на низове (лексикографско)*

1. Сравнява се всеки символ от първия низ със символа от съответната позиция на втория низ.
2. Сравнението продължава до намиране на два различни символа или до края на поне един от символните низове.

\* Ако кодът на символ от първия низ е по-малък от кода на съответния символ от втория низ, или първият низ е изчерпан, приема се, че първият низ е по-малък от втория.

\* Ако е по-голям или вторият низ е изчерпан – приема се, че първият низ е по-голям от втория.

\* Ако в процеса на сравнение и двата низа едновременно са изчерпани, те са равни

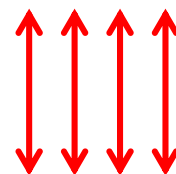
# Низове

## Сравнение на низове (лексикографско)

Пример:

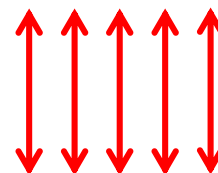
- "abc" == "abc"
- "abc" < "abcd"
- "abc" > "ab"
- "abcd" < "abd"
- "abc" > "aba"

"abcddas" <



"abcdd"

"abcddas" >

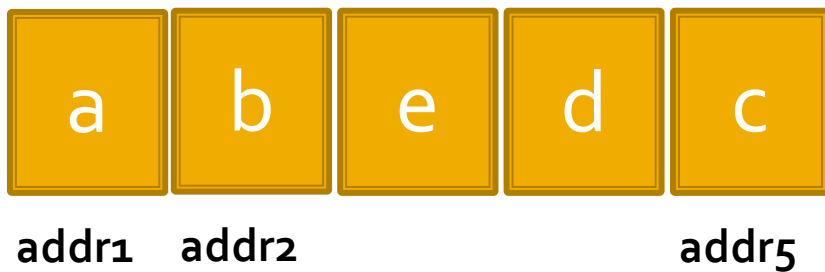


"abcdd"

# Низове

## *Физическо представяне*

Низовете се представят последователно в паметта



# Низове

Реализация на низ:

- чрез едномерен масив от символи;
- чрез указател към тип `char`;

*Забележка:*

*Двата начина са семантично еквивалентни*



# Масив от символи

```
char str[4]; // Определя масив от 4 символа.
```

```
char str2[4] = {'a', 'b'}; // Определя масив от 4 символа и го инициализира.
```

*Забележка:*

*Дефиницията е еквивалентна на:*

```
char str2[4] = {'a', 'b', '\\0', '\\0'};
```

# Масив от символи

```
char str2[4] = {'a', 'b'}; // Определя масив от 4 символа и го инициализира.
```

*Забележка:*

*Всички действия за работа с масиви са валидни и за работа с масиви от символи.*

**! Изключение правят операторите за въвеждане и извеждане (>>, <<)**

**>> - въвежда стойност на променлива от тип масив от символи.**

**<< - извежда стойност на променлива от тип масив от символи.**

# Масив от символи

```
char str2[4] = {'a', 'b'} ;
```

```
std::cout << str2; // Няма да изведе адреса на str2[0]
```

Резултат:

**ab**

Как се реализира извеждането?

# Масив от символи

```
char str2[4] = {'a', 'b'} ;  
std::cout << str2; // Няма да изведе адреса на str2[0]
```

Резултат:

ab

Ако инициализацията на str2 е пълна  
(не завършва с '\0')

```
char str2[4] = {'a', 'b', 'c', 'd'};  
std::cout << str2; // abcd<неопределено>
```

# Низове

(като масиви от символи)

Низ – редица от символи, която завършва с '\0'.

Предимство:

- Не е необходимо да се пази дължината на низа.

# Низове

(като масиви от символи)

```
char <променлива>[size] [= "редица_от_символи" |  
{<редица_от_символи>}]опц;
```

- <променлива> – идентификатор
- size – константен положителен израз

# Низове

(като масиви от символи)

## Множество от стойности

- Всички низове с дължина 0, 1, 2, ... size-1

Пример:

(Три еквивалентни дефиниции)

```
char str1[5] = {'a', 'b'};
```

```
char str2[5] = {'a', 'b', '\0', '\0', '\0'};
```

```
char str3[5] = "ab";
```

# Низове

(като масиви от символи)

// На променлива от тип низ не може да се присвоява константанта от тип низ.

```
char str[5];  
str = {'a', 'b'};
```

или



```
str = "ab";
```



# Низове

(като масиви от символи)

## Въвеждане на низове

- Инициализация при декларация.
- Индексирани променливи.

```
str[0] = '!';
```

- Чрез операции и вградени функции.

```
#include <cstring>
```

# Низове

(като масиви от символи)

## Операции и вградени функции

### Въвеждане на низ

- cin

Пример:

```
char str[5];  
std::cin >> str;
```



Очаква да се въведе стойност **< 5** (**<= 4**).

Знакът за край на низ се добавя автоматично.

Въвеждането продължава до въвеждане на интервал, табулация или знак за нов ред.

# Низове

(като масиви от символи)

- `cin.getline(<str>, <size> [, <char>] опц)`

`<str>` - променлива от тип низ

`<size>` - цял израз

`<char>` - произволен символ (по подразбиране `'\n'`)

Въвеждането продължава до срещане на символа `<char>` или до въвеждането на `<size>-1` символа.

Въведеното се записва в `<str>`.

Пример:

```
char str1[5];  
cin.getline(str1, 5);
```

# Низове

(като масиви от символи)

## Извеждане на низове

- `std::cout`

Пример:

```
char str[5] = "abc";  
std::cout << str;
```

# Низове

(като масиви от символи)

## Библиотеки за работа с низове

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

# Низове

(като масиви от символи)

## Дължина на низ

- `strlen(str)`

Пример:

```
char str[5] = "abc";  
std::cout << strlen(str);    // 3
```

# Низове

(като масиви от символи)

## Конкатенация на низове

- `strcat(<str1>, <str2>)`  
// ! Достатъчно място в `str1` (+ `strlen(str2) + 1`)

Пример:

```
char str1[7] = "abc";
```

```
char str2[5] = "abc";
```

```
std::cout << strcat(str1, str2); // abcabc
```

# Низове

(като масиви от символи)

## Конкатенация на n символа от низ с друг низ

- `strncat(<str1>, <str2>, n)`  
// ! Достатъчно място в `str1`

Прехвърля n символа или е достигнат края на `str2`.

Пример:

```
char str1[7] = "abc";
```

```
char str2[5] = "abc";
```

```
std::cout << strncat(str1, str2, 1); // abca
```



# Низове

(като масиви от символи)

## Сравняване на низове

- `strcmp(<str1>, <str2>)`

Връща: **-1** | **0** | **1**

Пример:

```
char str1[7] = "abc";
```

```
char str2[4] = "abc";
```

```
std::cout << strcmp(str1, str2); // 0
```

# Низове

(като масиви от символи)

## Сравняване на n символа на низове

- `strncmp(<str1>, <str2>, n)`

Връща: **-1** | **0** | **1**

- Ако n е по-малко от дължината на <str1>, се сравняват <str1>, <str2>

Пример:

```
char str1[7] = "abc";
```

```
char str2[4] = "abc";
```

```
std::cout << strncmp(str1, str2, 2); // 0
```

# Низове

(като масиви от символи)

## Копиране на низове

- `strcpy(<str1>, <str2>)`  
// ! Достатъчно място в `str1`

Пример:

```
char str1[10];  
char str2[4] = "abc";  
strcpy(str1, str2);  
std::cout << str1;           // abc
```

# Низове

(като масиви от символи)

## Копиране на низове

```
char str1[10] = "123456789";  
char str2[4] = "abc";  
strcpy(str1, str2);  
std::cout << str1;
```

Изход:

abc

# Низове

(като масиви от символи)

## Копиране на низове

```
char str1[10] = "0123456789";  
char str2[4] = "abc";  
strcpy(str1, str2);  
std::cout << str1;
```

Изход:

```
ERROR - const char[11]
```

# Низове

(като масиви от символи)

## Копиране на n символа на низове

- `strncpy(<str1>, <str2>, n)`  
// ! Достатъчно място в `str1`

Пример:

```
char str1[10];  
char str2[4] = "abc";  
strncpy(str1, str2, 2);  
std::cout << str1;      // ab
```

```
strncpy(str1, str2, 5); // 5 > strlen(str2), допълва се с '\0'  
std::cout << str1;     // abc
```

# Низове

(като масиви от символи)

## Копиране на n символа на низове

- `strncpy(<str1>, <str2>, n)`  
// ! Достатъчно място в `str1`

Пример:

```
char str1[10];  
char str2[4] = "abc";  
strncpy(str1, str2, 12);  
std::cout << str1; // Error; 12 > 10
```

# Низове

(като масиви от символи)

## Търсене на низ в друг низ

- `strstr(<str1>, <str2>)`

Връща:

- Подниза на `<str1>`, започващ от първото срещане на `<str2>` до края на `<str1>`.
- `nullptr`

Пример:

```
char str1[10] = "1234abcdef";  
char str2[5] = "ab";
```

```
std::cout << strstr(str1, str2); // abcdf
```



# Низове

(като масиви от символи)

Пример:

```
char str1[10] = "1234abcdf";
```

```
char str2[5] = "567";
```

```
std::cout << strstr(str1, str2); // Грешка
```

```
if (strstr(str1, str2)) {  
    //...  
}
```

# Низове

(като масиви от символи)

## Търсене на символ в низ

- `strchr(<str1>, <expr>)`

`<expr>` - израз от интегрален или изброен тип с положителна стойност, означаваща ASCII код на символ.

Връща:

- Подниз на `<str1>`, започващ от първото срещане на символ с ASCII `<expr>` до края на `<str1>`.
- `nullptr`

Пример:

```
char str1[10] = "abcdef";  
std::cout << strchr(str1, 'd');    // df
```

# Низове

(като масиви от символи)

Пример:

```
char str1[10] = "1234abcdf";
```

```
std::cout << strchr(str1, '5'); // Грешка
```

```
if (strchr(str1, '5')) {  
    //...  
}
```

# Низове

(като масиви от символи)

## Търсене на първа разлика

- `strspn(<str1>, <str2>)`

Връща:

- дължината на низа до първия различен символ.

Пример:

```
char str1[6] = "abcds";
```

```
char str2[6] = "abdse";
```

```
std::cout << strspn(str1, str2) << endl; // 2
```

# Низове

(като масиви от символи)

## Преобразуване на низ в цяло число

- `atoi(<str>)`

Резултат:

- Преобразува низа `<str>` в число от тип `int`.
- Сканира се до първия символ различен от цифра и знак.
- Интервалите, табулациите и `'\n'` се пренебрегват.

Пример:

```
char str1[10] = "-1234abcd";  
std::cout << atoi(str1);           // -1234
```

# Низове

(като масиви от символи)

## Преобразуване на низ в цяло число

- `atoi(<str>)`

Пример:

```
char str1[10] = "a123bcdf";  
std::cout << atoi(str1);           // 0
```

# Низове

(като масиви от символи)

## Преобразуване на низ в реално число

- `atof(<str>)`

Резултат:

- Преобразува низа `<str>` в число от тип `double`.
- Сканира се до първия символ различен от цифра, знак и `'.'`.
- Интервалите, табулациите и `'\n'` се пренебрегват.

Пример:

```
char str1[10] = "-123.4ab";  
std::cout << atof(str1);           // -123.4
```

# Низове

(като масиви от символи)

## Преобразуване на низ в реално число

- `atof(<str>)`

Пример:

```
char str1[10] = ".4abcdef";  
std::cout << atof(str1);           // 0.4
```

```
char str2[10] = "abcdef";  
std::cout << atof(str2);           // 0
```



# Низове

(като масиви от символи)

Задача.

Да се напише програмен фрагмент, който създава нов обърнат низ.

```
char str[20] = "123454321";
char reverseStr[20];
int length = strlen(str);

for(int i = 0; i < length; ++i) {
    reverseStr[i] = str[length - 1 - i];
}

reverseStr[length] = '\0';
```

# Низове

(като масиви от символи)

Задача 2.

Символен низ е съставен единствено от малки латински букви.

Да се напише програма, която намира и извежда на екрана броя на срещанията на всяка от буквите на низа.

```
#include <iostream.h>
#include <string.h>
const int MAX_SIZE = 100;
int main()
{ char s[MAX_SIZE];
  unsigned int histogram[26] = {0},
              i = 0;
  cout << "s= ";
  cin >> s;
  for(i = 0; i < strlen(s); i++)
    histogram[s[i]-'a']++;
  for(i = 0; i < 26; i++)
    if(histogram[i] > 0)
      cout << (char)('a'+i) << ": "
           << histogram[i] << endl;
  return 0;
}
```

Не гарантира правилно въвеждане.

Използвайте `cin.getline`

`strlen(s)` се извиква всеки път.

Използвайте променлива или константа.

# Низове

(като масиви от символи)

Името на променлива от тип низ е **константен** указател.

```
char str[7] = "123456";  
  
while(*str) {  
    std::cout << *str << " ";  
    str++; // Error  
}
```

# Низове

(като масиви от символи)

Името на променлива от тип низ е **константен** указател.

```
// Указателят str е константен и стойността му не може да се променя
// ВАЖИ И ЗА ВСЕКИ ДРУГ МАСИВ
char str[7] = "abcdef";
char *strElementPtr = str;

while(*strElementPtr) {
    std::cout << *strElementPtr << " ";
    strElementPtr++;           // a b c d e f
}
```

# Низове

(като масиви от символи)

Указател към символ от низа

```
char str[7] = "abcdef";  
char *strElementPtr = str;
```

```
std::cout << strElementPtr + 3; // def
```

# Низове

(като указатели от тип `char`)

# Низове

(като указател от тип char)

```
const char *str = "123456";

while(*str) {
    std::cout << *str << " ";
    str++;
} // 1 2 3 4 5 6
```



# Низове

(като указател от тип `char`)

*\* Не е възможно въвеждане на стойност от тип указател*

```
char *str;  
std::cin >> str;    // ERROR  
(getline, ...)
```

# Низове

(като указател от тип char)

```
char str[7] = "123456";  
*(str+1) = '9';  
str[1] = '9';  
std::cout << str;
```

Изход:

193456

```
const char *str2 = "123456";  
*(str2+1) = '9';           // Error assignment of read-only location  
str2[1] = '9';           // Error assignment of read-only location  
std::cout << str2;
```

Следва продължение...