

Масиви и низове

Трифон Трифонов

Увод в програмирането,
спец. Компютърни науки, 1 поток,
спец. Софтуерно инженерство,
2016/17 г.

9–16 ноември 2016 г.

Логическо описание

Масивът

- е съставен тип данни
- представя крайни редици от елементи
- всички елементи са от един и същи тип
- позволява произволен достъп до всеки негов елемент по номер (индекс)

Дефиниция на масив

```
<тип> <идентификатор> [ [ <константа> ]  
    [ = { <константа> { , <константа> } } ] ] ;
```

Дефиниция на масив

```
<тип> <идентификатор> [ [ <константа> ]  
    [ = { <константа> { , <константа> } } ] ] ;
```

Примери:

- `bool b[10];`

Дефиниция на масив

```
<тип> <идентификатор> [ [ <константа> ]  
    [ = { <константа> { , <константа> } } ] ] ;
```

Примери:

- `bool b[10];`
- `double x[3] = { 0.5, 1.5, 2.5 }, y = 3.8;`

Дефиниция на масив

```
<тип> <идентификатор> [ [ <константа> ]  
    [ = { <константа> { , <константа> } } ] ] ;
```

Примери:

- `bool b[10];`
- `double x[3] = { 0.5, 1.5, 2.5 }, y = 3.8;`
- `int a[] = { 3 + 2, 2 * 4 }; \iff int a[2] = { 5, 8 };`

Дефиниция на масив

```
<тип> <идентификатор> [ [ <константа> ]
    [ = { <константа> { , <константа> } } ] ;
```

Примери:

- `bool b[10];`
- `double x[3] = { 0.5, 1.5, 2.5 }, y = 3.8;`
- `int a[] = { 3 + 2, 2 * 4 }; ⇔ int a[2] = { 5, 8 };`
- `float f[4] = { 2.3, 4.5 }; ⇔`
`float f[4] = { 2.3, 4.5, 0, 0 };`

Физическо представяне

a

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
------	------	------	------	------	------	------	------	------	------	-------

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:
 - `x = a[2];` (*rvalue*)

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:
 - `x = a[2];` (*rvalue*)
 - `a[i] = 7;` (*lvalue!*)

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:
 - `x = a[2];` (*rvalue*)
 - `a[i] = 7;` (*lvalue!*)
 - **Внимание:** няма проверка за коректност на индекса!

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:
 - `x = a[2];` (*rvalue*)
 - `a[i] = 7;` (*lvalue!*)
 - **Внимание:** няма проверка за коректност на индекса!
- Няма присвояване

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:
 - `x = a[2];` (*rvalue*)
 - `a[i] = 7;` (*lvalue!*)
 - **Внимание:** няма проверка за коректност на индекса!
- Няма присвояване
 - ~~`a = b`~~

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:
 - `x = a[2];` (*rvalue*)
 - `a[i] = 7;` (*lvalue!*)
 - **Внимание:** няма проверка за коректност на индекса!
- Няма присвояване
 - ~~`a = b`~~
- Няма поелементно сравнение

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:
 - `x = a[2];` (*rvalue*)
 - `a[i] = 7;` (*lvalue!*)
 - **Внимание:** няма проверка за коректност на индекса!
- Няма присвояване
 - ~~`a = b`~~
- Няма поелементно сравнение
 - `a == b` винаги връща *false* ако *a* и *b* са различни масиви, дори и да имат еднакви елементи

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:
 - `x = a[2];` (*rvalue*)
 - `a[i] = 7;` (*lvalue!*)
 - **Внимание:** няма проверка за коректност на индекса!
- Няма присвояване
 - ~~`a = b`~~
- Няма поелементно сравнение
 - `a == b` винаги връща `false` ако `a` и `b` са различни масиви, дори и да имат еднакви елементи
- Няма операции за вход и изход

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:
 - `x = a[2];` (*rvalue*)
 - `a[i] = 7;` (*lvalue!*)
 - **Внимание:** няма проверка за коректност на индекса!
- Няма присвояване
 - ~~`a = b`~~
- Няма поелементно сравнение
 - `a == b` винаги връща `false` ако `a` и `b` са различни масиви, дори и да имат еднакви елементи
- Няма операции за вход и изход
 - ~~`cin >> a;`~~

Операции за работа с масиви

- Достъп до елемент по индекс: `<масив> [<цяло_число>]`
- Примери:
 - `x = a[2];` (rvalue)
 - `a[i] = 7;` (lvalue!)
 - **Внимание:** няма проверка за коректност на индекса!
- Няма присвояване
 - ~~`a = b`~~
- Няма поелементно сравнение
 - `a == b` винаги връща `false` ако `a` и `b` са различни масиви, дори и да имат еднакви елементи
- Няма операции за вход и изход
 - ~~`cin >> a;`~~
 - `cout << a;` извежда адреса на `a`

Задачи за масиви

- Да се въведе масив от числа

Задачи за масиви

- Да се въведе масив от числа
- Да се изведе масив от числа

Задачи за масиви

- Да се въведе масив от числа
- Да се изведе масив от числа
- Да се намери сумата на числата в даден масив

Задачи за масиви

- Да се въведе масив от числа
- Да се изведе масив от числа
- Да се намери сумата на числата в даден масив
- Да се провери дали дадено число се среща в масив

$$\exists 0 \leq i < n \\ a[i] == x$$

Задачи за масиви

- Да се въведе масив от числа
- Да се изведе масив от числа
- Да се намери сумата на числата в даден масив
- Да се провери дали дадено число се среща в масив
- Да се провери дали числата в масив нарастват монотонно

$$\neg \exists 0 \leq i < n-1 \quad a[i] > a[i+1]$$
$$\forall 0 \leq i < n-1 \quad a[i] \leq a[i+1]$$

Задачи за масиви

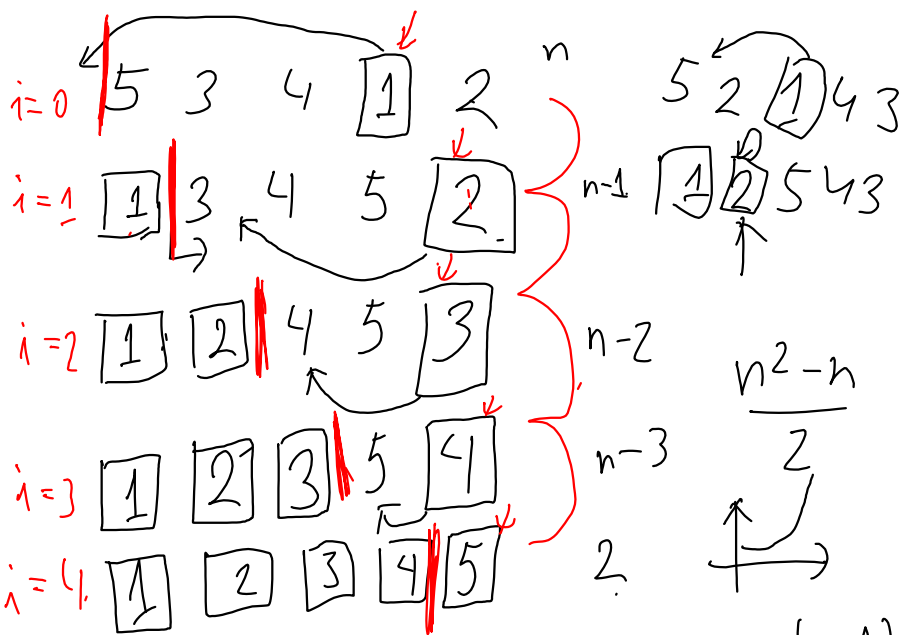
- Да се въведе масив от числа
- Да се изведе масив от числа
- Да се намери сумата на числата в даден масив
- Да се провери дали дадено число се среща в масив
- Да се провери дали числата в масив нарастват монотонно
- Да се провери дали всички числа в даден масив са различни

$$\forall 0 \leq i < j < n \quad a[i] \neq a[j]$$

$$\exists 0 \leq i < j < n \quad a[i] = a[j]$$

Задачи за масиви

- Да се въведе масив от числа
- Да се изведе масив от числа
- Да се намери сумата на числата в даден масив
- Да се провери дали дадено число се среща в масив
- Да се провери дали числата в масив нарастват монотонно
- Да се провери дали всички числа в даден масив са различни
- Да се подредят числата в даден масив в нарастващ ред



$$2+3+\dots+n = \frac{n(n-1)}{2}$$

Задачи за масиви

- Да се въведе масив от числа
- Да се изведе масив от числа
- Да се намери сумата на числата в даден масив
- Да се провери дали дадено число се среща в масив
- Да се провери дали числата в масив нарастват монотонно
- Да се провери дали всички числа в даден масив са различни
- Да се подредят числата в даден масив в нарастващ ред
- Да се слоят два масива подредени в нарастващ ред

a, n 1 3 6 7 10 | \rightarrow 1 2 3 5 6 7 7 8 10
 b, m 2 5 7 8

Низове: описание и представяне

- **Описание:** **Низ** наричаме последователност от символи

Низове: описание и представяне

- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**

Низове: описание и представяне

- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**
- **Представяне в C++:** Масив от символи (**char**), в който след последния символ в низа е записан **терминиращият символ** `'\0'`

Низове: описание и представяне

- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**
- **Представяне в C++:** Масив от символи (`char`), в който след последния символ в низа е записан **терминиращият символ** `'\0'`
 - `'\0'` е първият символ в ASCII таблицата, с код 0

NUL

Низове: описание и представяне

- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**
- **Представяне в C++:** Масив от символи (`char`), в който след последния символ в низа е записан **терминиращият символ** `'\0'`
 - `'\0'` е първият символ в ASCII таблицата, с код 0
- **Примери:**

Низове: описание и представяне

- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**
- **Представяне в C++:** Масив от символи (**char**), в който след последния символ в низа е записан **терминиращият символ** `'\0'`
 - `'\0'` е първият символ в ASCII таблицата, с код 0
- **Примери:**
 - `char word[] = { 'H', 'e', 'l', 'l', 'o', '\0' };`

Низове: описание и представяне

- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**
- **Представяне в C++:** Масив от символи (**char**), в който след последния символ в низа е записан **терминиращият символ** `'\0'`
 - `'\0'` е първият символ в ASCII таблицата, с код 0
- **Примери:**
 - `char word[] = { 'H', 'e', 'l', 'l', 'o', '\0' };`
 - `char word[6] = { 'H', 'e', 'l', 'l', 'o' };`

Низове: описание и представяне

- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**
- **Представяне в C++:** Масив от символи (**char**), в който след последния символ в низа е записан **терминиращият символ** `'\0'`
 - `'\0'` е първият символ в ASCII таблицата, с код 0
- **Примери:**
 - `char word[] = { 'H', 'e', 'l', 'l', 'o', '\0' };`
 - `char word[6] = { 'H', 'e', 'l', 'l', 'o' };`
 - `char word[100] = "Hello";`

Низове: описание и представяне

- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**
- **Представяне в C++:** Масив от символи (**char**), в който след последния символ в низа е записан **терминиращият символ** `'\0'`
 - `'\0'` е първият символ в ASCII таблицата, с код 0
- **Примери:**
 - `char word[] = { 'H', 'e', 'l', 'l', 'o', '\0' };`
 - `char word[6] = { 'H', 'e', 'l', 'l', 'o' };`
 - `char word[100] = "Hello";`
 - ~~`char word[5] = "Hello";`~~

Низове: описание и представяне

- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**
- **Представяне в C++:** Масив от символи (**char**), в който след последния символ в низа е записан **терминиращият символ** `'\0'`
 - `'\0'` е първият символ в ASCII таблицата, с код 0
- **Примери:**
 - `char word[] = { 'H', 'e', 'l', 'l', 'o', '\0' };`
 - `char word[6] = { 'H', 'e', 'l', 'l', 'o' };`
 - `char word[100] = "Hello";`
 - ~~`char word[5] = "Hello";`~~
 - `char word[6] = "Hello";`

Низове: описание и представяне

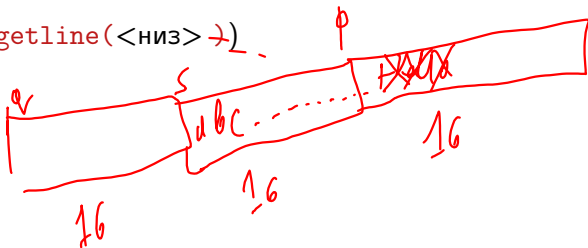
- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**
- **Представяне в C++:** Масив от символи (**char**), в който след последния символ в низа е записан **терминиращият символ** '\0'
 - '\0' е първият символ в ASCII таблицата, с код 0
- **Примери:**
 - `char word[] = { 'H', 'e', 'l', 'l', 'o', '\0' };`
 - `char word[6] = { 'H', 'e', 'l', 'l', 'o' };`
 - `char word[100] = "Hello";`
 - ~~`char word[5] = "Hello";`~~
 - `char word[6] = "Hello";`
 - ~~`char word[5] = { 'H', 'e', 'l', 'l', 'o' };`~~

Операции за работа с низове

- Вход (>>, `cin.getline(<низ>)`)

Операции за работа с низове

- Вход (`>>`, `cin.getline(<низ> ↗)`)
- Изход (`<<`)



Операции за работа с низове

- Вход (`>>`, `cin.getline(<низ>)`)
- Изход (`<<`)
- Индексиране (`[]`)

Операции за работа с низове

- Вход (`>>`, `cin.getline(<низ>)`)
- Изход (`<<`)
- Индексиране (`[]`)
- Няма присвояване! (~~`a = b`~~)

Операции за работа с низове

- Вход (`>>`, `cin.getline(<низ>)`)
- Изход (`<<`)
- Индексиране (`[]`)
- Няма присвояване! (~~`a=b`~~)
- Няма поелементно сравнение! (~~`a==b`~~)

Операции за работа с низове

- Вход (`>>`, `cin.getline(<низ>)`)
- Изход (`<<`)
- Индексиране (`[]`)
- Няма присвояване! (~~`a=b`~~)
- Няма поелементно сравнение! (~~`a==b`~~)
- но...

Операции за работа с низове

- Вход (`>>`, `cin.getline(<низ>)`)
- Изход (`<<`)
- Индексиране (`[]`)
- Няма присвояване! (~~`a = b`~~)
- Няма поелементно сравнение! (~~`a == b`~~)
- но...
- ...има вградени функции!

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<НИЗ>)`

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`

- връща дължината на <низ>, т.е. броя символи без '\0'

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`

- връща дължината на <низ>, т.е. броя символи без '\0'

- `strcpy(<буфер>, <низ>)`

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`

- връща дължината на <низ>, т.е. броя символи без '\0'

- `strcpy(<буфер>, <низ>)`

- прехвърля всички символи от <низ> в <буфер>

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`

- връща дължината на <низ>, т.е. броя символи без '\0'

- `strcpy(<буфер>, <низ>)`

- прехвърля всички ~~символи~~ от <низ> в <буфер>
- връща <буфер>

`strcpy("a", "a") = 2`

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`

- връща дължината на <низ>, т.е. броя символи без '\0'

- `strcpy(<буфер>, <низ>)`

- прехвърля всички символи от <низ> в <буфер>
- връща <буфер>
- отговорност на програмиста е да осигури, че в <буфер> да има достатъчно място да поеме всички символи на <низ>

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`

- връща дължината на <низ>, т.е. броя символи без '\0'

- `strcpy(<буфер>, <низ>)`

- прехвърля всички символи от <низ> в <буфер>
- връща <буфер>
- отговорност на програмиста е да осигури, че в <буфер> да има достатъчно място да поеме всички символи на <низ>

- `strcmp(<низ1>, <низ2>)`

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`

- връща дължината на <низ>, т.е. броя символи без `'\0'`

- `strcpy(<буфер>, <низ>)`

- прехвърля всички символи от <низ> в <буфер>
- връща <буфер>
- отговорност на програмиста е да осигури, че в <буфер> да има достатъчно място да поеме всички символи на <низ>

- `strcmp(<низ1>, <низ2>)`

- сравнява два низа **лексикографски** (речникова наредба)

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`
 - връща дължината на <низ>, т.е. броя символи без `'\0'`
- `strcpy(<буфер>, <низ>)`
 - прехвърля всички символи от <низ> в <буфер>
 - връща <буфер>
 - отговорност на програмиста е да осигури, че в <буфер> да има достатъчно място да поеме всички символи на <низ>
- `strcmp(<низ1>, <низ2>)`
 - сравнява два низа **лексикографски** (речникова наредба)
 - връща `-1`, ако <низ₁> е преди <низ₂>

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`

- връща дължината на <низ>, т.е. броя символи без `'\0'`

- `strcpy(<буфер>, <низ>)`

- прехвърля всички символи от <низ> в <буфер>
- връща <буфер>
- отговорност на програмиста е да осигури, че в <буфер> да има достатъчно място да поеме всички символи на <низ>

- `strcmp(<низ1>, <низ2>)`

- сравнява два низа **лексикографски** (речникова наредба)
- връща `-1`, ако <низ₁> е преди <низ₂>
- връща `0`, ако <низ₁> съвпада с <низ₂>

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`
 - връща дължината на <низ>, т.е. броя символи без `'\0'`
- `strcpy(<буфер>, <низ>)`
 - прехвърля всички символи от <низ> в <буфер>
 - връща <буфер>
 - отговорност на програмиста е да осигури, че в <буфер> да има достатъчно място да поеме всички символи на <низ>
- `strcmp(<низ1>, <низ2>)`
 - сравнява два низа **лексикографски** (речникова наредба)
 - връща `-1`, ако <низ₁> е преди <низ₂> \leftrightarrow true
 - връща `0`, ако <низ₁> съвпада с <низ₂> \leftrightarrow false
 - връща `1`, ако <низ₁> е след <низ₂> \leftrightarrow true

~~if (strcmp(s1, s2) ...)~~

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`

- връща дължината на <низ>, т.е. броя символи без '\0'

- `strcpy(<буфер>, <низ>)`

- прехвърля всички символи от <низ> в <буфер>
- връща <буфер>
- отговорност на програмиста е да осигури, че в <буфер> да има достатъчно място да поеме всички символи на <низ>

- `strcmp(<низ1>, <низ2>)`

- сравнява два низа **лексикографски** (речникова наредба)
- връща **-1**, ако <низ₁> е преди <низ₂>
- връща **0**, ако <низ₁> съвпада с <низ₂>
- връща **1**, ако <низ₁> е след <низ₂>
- **Интуиция:** "знакът" на "разликата" <низ₁> - <низ₂>

5 3 +
3 5 -

Вградени функции за низове

```
#include <cstring>
```

- `strlen(<низ>)`

- връща дължината на <низ>, т.е. броя символи без `'\0'`

- `strcpy(<буфер>, <низ>)`

- прехвърля всички символи от <низ> в <буфер>
- връща <буфер>
- **отговорност на програмиста е да осигури, че в <буфер> да има достатъчно място да поеме всички символи на <низ>**

- `strcmp(<низ1>, <низ2>)`

- сравнява два низа **лексикографски** (речникова наредба)
- връща `-1`, ако <низ₁> е преди <низ₂>
- връща `0`, ако <низ₁> съвпада с <низ₂>
- връща `1`, ако <низ₁> е след <низ₂>
- **Интуиция:** “знакът” на “разликата” <низ₁> – <низ₂>
- **Свойство:** `strcmp(s1, s2) == -strcmp(s2, s1)`

Вградени функции за низове

- `strcat(<НИЗ1>, <НИЗ2>)`

Вградени функции за низове

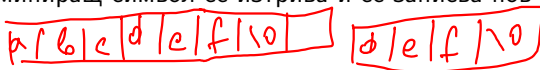
- `strcat(<низ1>, +=<низ2>)`
 - **конкатенация** (слепване) на низове

Вградени функции за низове

- `strcat(<низ1>, <низ2>)`
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>

Вградени функции за низове

- `strcat(<низ1>, <низ2>)`
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминиращ символ се изтрива и се записва нов



Вградени функции за низове

- `strcat(<низ1> <низ2>)` $a + 5$
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминаращ символ се изтрива и се записва нов
 - връща <низ₁>

Вградени функции за низове

- `strcat(<низ1>, <низ2>)`
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминаращ символ се изтрива и се записва нов
 - връща <низ₁>
 - **отговорност на програмиста е да осигури, че в <низ₁> да има достатъчно място да поеме всички символи на <низ₂>**

Вградени функции за низове

- `strcat(<низ1>, <низ2>)`
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминаращ символ се изтрива и се записва нов
 - връща <низ₁>
 - **отговорност на програмиста е да осигури, че в <низ₁> да има достатъчно място да поеме всички символи на <низ₂>**
- `strchr(<низ>, <символ>)`

Вградени функции за низове

- `strcat(<низ1>, <низ2>)`
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминаращ символ се изтрива и се записва нов
 - връща <низ₁>
 - **отговорност на програмиста е да осигури, че в <низ₁> да има достатъчно място да поеме всички символи на <низ₂>**
- `strchr(<низ>, <символ>)`
 - търсене на <символ> в <низ>

Вградени функции за низове

- `strcat(<низ1>, <низ2>)`
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминаращ символ се изтрива и се записва нов
 - връща <низ₁>
 - **отговорност на програмиста е да осигури, че в <низ₁> да има достатъчно място да поеме всички символи на <низ₂>**
- `strchr(<низ>, <символ>)`
 - търсене на <символ> в <низ>
 - връща **суфикса** на <низ> от първото срещане на <символ>

Вградени функции за низове

- `strcat(<низ1>, <низ2>)`
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминаращ символ се изтрива и се записва нов
 - връща <низ₁>
 - **отговорност на програмиста е да осигури, че в <низ₁> да има достатъчно място да поеме всички символи на <низ₂>**
- `strchr(<низ>, <символ>)`
 - търсене на <символ> в <низ>
 - връща **суфикса** на <низ> от първото срещане на <символ>
 - връща 0, ако <символ> не се среща в <низ>

Вградени функции за низове

- **strcat**(<низ₁>, <низ₂>)
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминаращ символ се изтрива и се записва нов
 - връща <низ₁>
 - **отговорност на програмиста е да осигури, че в <низ₁> да има достатъчно място да поеме всички символи на <низ₂>**
- **strchr**(<низ>, <символ>)
 - търсене на <символ> в <низ>
 - връща **суфикса** на <низ> от първото срещане на <символ>
 - връща 0, ако <символ> не се среща в <низ>
- **strstr**(<низ>, <подниз>)

Вградени функции за низове

- **strcat**(<низ₁>, <низ₂>)
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминаращ символ се изтрива и се записва нов
 - връща <низ₁>
 - **отговорност на програмиста е да осигури, че в <низ₁> да има достатъчно място да поеме всички символи на <низ₂>**
- **strchr**(<низ>, <символ>)
 - търсене на <символ> в <низ>
 - връща **суфикса** на <низ> от първото срещане на <символ>
 - връща 0, ако <символ> не се среща в <низ>
- **strstr**(<низ>, <подниз>)
 - търсене на <подниз> в <низ>

Вградени функции за низове

- **strcat**(<низ₁>, <низ₂>)
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминиращ символ се изтрива и се записва нов
 - връща <низ₁>
 - **отговорност на програмиста е да осигури, че в <низ₁> да има достатъчно място да поеме всички символи на <низ₂>**
- **strchr**(<низ>, <символ>)
 - търсене на <символ> в <низ>
 - връща **суфикса** на <низ> от първото срещане на <символ>
 - връща 0, ако <символ> не се среща в <низ>
- **strstr**(<низ>, <подниз>)
 - търсене на <подниз> в <низ>
 - т.е. символите на <подниз> да се срещат последователно в <низ>

Вградени функции за низове

- **strcat**(<низ₁>, <низ₂>)
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминиращ символ се изтрива и се записва нов
 - връща <низ₁>
 - **отговорност на програмиста е да осигури, че в <низ₁> да има достатъчно място да поеме всички символи на <низ₂>**
- **strchr**(<низ>, <символ>)
 - търсене на <символ> в <низ>
 - връща **суфикса** на <низ> от първото срещане на <символ>
 - връща 0, ако <символ> не се среща в <низ>
- **strstr**(<низ>, <подниз>)
 - търсене на <подниз> в <низ>
 - т.е. символите на <подниз> да се срещат последователно в <низ>
 - връща **суфикса** на <низ> от първото срещане на <подниз>

Вградени функции за низове

- **strcat**(<низ₁>, <низ₂>)
 - **конкатенация** (слепване) на низове
 - записва символите на <низ₂> в края на <низ₁>
 - старият терминиращ символ се изтрива и се записва нов
 - връща <низ₁>
 - **отговорност на програмиста е да осигури, че в <низ₁> да има достатъчно място да поеме всички символи на <низ₂>**
- **strchr**(<низ>, <символ>)
 - търсене на <символ> в <низ>
 - връща **суфикса** на <низ> от първото срещане на <символ>
 - връща 0, ако <символ> не се среща в <низ>
- **strstr**(<низ>, <подниз>)
 - търсене на <подниз> в <низ>
 - т.е. символите на <подниз> да се срещат последователно в <низ>
 - връща **суфикса** на <низ> от първото срещане на <подниз>
 - връща 0, ако <символ> не се среща в <низ>