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 };` $\Longleftrightarrow$ `int a[2] = { 5, 8 };`

Дефиниция на масив

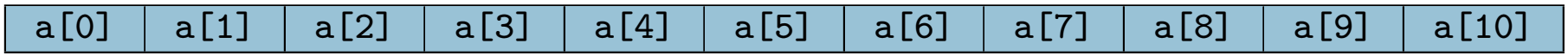
<тип> <идентификатор> [[<константа]]
[= { <константа> {, <константа> } }] ;

Примери:

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

Физическо представяне

a



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

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

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

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

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 слеят два масива подредени в нарастващ ред

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

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>

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(`<низ$_1$>, <низ$_2$> `)`

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>
- `strcmp(`<низ$_1$>, <низ$_2$> `)`
  - сравнява два низа лексикорафски (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(`<низ$_1$>, <низ$_2$> `)`
  - сравнява два низа лексикорафски (речникова наредба)
  - връща -1, ако <низ$_1$> е преди <низ$_2$>

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>
- `strcmp(`<низ$_1$>, <низ$_2$> `)`
  - сравнява два низа лексикорафски (речникова наредба)
  - връща -1, ако <низ$_1$> е преди <низ$_2$>
  - връща 0, ако <низ$_1$> съвпада с <низ$_2$>

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>
- `strcmp(`<низ$_1$>, <низ$_2$> `)`
  - сравнява два низа лексикорафски (речникова наредба)
  - връща -1, ако <низ$_1$> е преди <низ$_2$>
  - връща 0, ако <низ$_1$> съвпада с <низ$_2$>
  - връща 1, ако <низ$_1$> е след <низ$_2$>

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>
- `strcmp(`<низ$_1$>, <низ$_2$> `)`
  - сравнява два низа лексикорафски (речникова наредба)
  - връща -1, ако <низ$_1$> е преди <низ$_2$>
  - връща 0, ако <низ$_1$> съвпада с <низ$_2$>
  - връща 1, ако <низ$_1$> е след <низ$_2$>
  - **Интуиция:** "знакът" на "разликата" <низ$_1$> $-$ <низ$_2$>

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>
- `strcmp(`<низ$_1$>, <низ$_2$> `)`
  - сравнява два низа лексикорафски (речникова наредба)
  - връща -1, ако <низ$_1$> е преди <низ$_2$>
  - връща 0, ако <низ$_1$> съвпада с <низ$_2$>
  - връща 1, ако <низ$_1$> е след <низ$_2$>
  - **Интуиция:** "знакът" на "разликата" <низ$_1$> $-$ <низ$_2$>
  - **Свойство:** `strcmp(s1, s2) == -strcmp(s2, s1)`

Вградени функции за низове

- `strcat(`<низ$_1$>, <низ$_2$> `)`

Вградени функции за низове

- `strcat(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове

Вградени функции за низове

- `strcat(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>

Вградени функции за низове

- `strcat(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов

Вградени функции за низове

- `strcat(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>

Вградени функции за низове

- `strcat(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>
  - отговорност на програмиста е да осигури, че в <низ$_1$> да има достатъчно място да поеме всички символи на <низ$_2$>

Вградени функции за низове

- `strcat(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>
  - отговорност на програмиста е да осигури, че в <низ$_1$> да има достатъчно място да поеме всички символи на <низ$_2$>
- `strchr(`<низ>, <символ>`)`

Вградени функции за низове

- `strcat(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>
  - отговорност на програмиста е да осигури, че в <низ$_1$> да има достатъчно място да поеме всички символи на <низ$_2$>
- `strchr(`<низ>, <символ>`)`
  - търсене на <символ> в <низ>

Вградени функции за низове

- `strcat(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>
  - отговорност на програмиста е да осигури, че в <низ$_1$> да има достатъчно място да поеме всички символи на <низ$_2$>
- `strchr(`<низ>, <символ>`)`
  - търсене на <символ> в <низ>
  - връща суфикса на <низ> от първото срещане на <символ>

Вградени функции за низове

- `strcat(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>
  - отговорност на програмиста е да осигури, че в <низ$_1$> да има достатъчно място да поеме всички символи на <низ$_2$>
- `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(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>
  - отговорност на програмиста е да осигури, че в <низ$_1$> да има достатъчно място да поеме всички символи на <низ$_2$>
- `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(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>
  - отговорност на програмиста е да осигури, че в <низ$_1$> да има достатъчно място да поеме всички символи на <низ$_2$>
- `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(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>
  - отговорност на програмиста е да осигури, че в <низ$_1$> да има достатъчно място да поеме всички символи на <низ$_2$>
- `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(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>
  - отговорност на програмиста е да осигури, че в <низ$_1$> да има достатъчно място да поеме всички символи на <низ$_2$>
- `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(`<низ$_1$>, <низ$_2$> `)`
  - конкатенация (слепване) на низове
  - записва символите на <низ$_2$> в края на <низ$_1$>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ$_1$>
  - отговорност на програмиста е да осигури, че в <низ$_1$> да има достатъчно място да поеме всички символи на <низ$_2$>
- `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>