

# Масиви и низове

Трифон Трифонов

Увод в програмирането,  
спец. Компютърни науки, 1 поток, 2018/19 г.

15–22 ноември 2018 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> } } ] ] ;
```

## 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>]`
- 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
- 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

# 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>, <число>)`)
  - `>>` въвежда до разделител (интервал, табулация, нов ред)
  - `cin.getline(<низ>, <число>)` въвежда до нов ред, но не повече от `<число>-1` символа
- Изход (`<<`)
- Индексиране (`[]`)
- Няма присвояване! (~~`a = b`~~)
- Няма поелементно сравнение! (~~`a == b`~~)
- но...
- ...има вградени функции!



# 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 число  $< 0$ , ако <низ<sub>1</sub>> е преди <низ<sub>2</sub>>
- връща  $0$ , ако <низ<sub>1</sub>> съвпада с <низ<sub>2</sub>>
- връща число  $> 0$ , ако <низ<sub>1</sub>> е след <низ<sub>2</sub>>
- **Интуиция:** “знакът” на “разликата” <низ<sub>1</sub>> – <низ<sub>2</sub>>
- **Свойство:** `strcmp(s1, s2) == -strcmp(s2, s1)`