

# Масиви и низове

Трифон Трифонов

Увод в програмирането,  
спец. Компютърни науки, 1 поток, 2021/22 г.

15–29 ноември 2021 г.

Тази презентация е достъпна под лиценза Creative Commons Признание-Некомерсиално-Споделяне на споделеното 4.0 Международен 

# 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> ]  
    [ = { <израз> { , <израз> } } ] ] ;
```

## 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 };`
- `float f[4] = { 2.3, 4.5 };  $\iff$   
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**(<низ<sub>1</sub>>, <низ<sub>2</sub>>)

- 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)

# Вградени функции за низове

- **strcat**(<низ<sub>1</sub>>, <низ<sub>2</sub>>)
  - **конкатенация** (слепване) на низове
  - записва символите на <низ<sub>2</sub>> в края на <низ<sub>1</sub>>
  - старият терминиращ символ се изтрива и се записва нов
  - връща <низ<sub>1</sub>>
  - **отговорност на програмиста е да осигури, че в <низ<sub>1</sub>> да има достатъчно място да поеме всички символи на <низ<sub>2</sub>>**
- **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>

# Задачи за низове

- Да се провери дали даден низ е **палиндром**
  - чете се еднакво в двете посоки
  - "abba", "racecar", "risetovotesir", "wasitacaroracatisaw"
- Да се преброят думите в даден низ
  - Считаме, че за разделители служат всички символи, които не са букви.
- Да се пресметне аритметичен израз, записан в низ
  - $\langle \text{израз} \rangle ::= \langle \text{число} \rangle \{ \langle \text{операция} \rangle \langle \text{число} \rangle \} =$
  - $\langle \text{число} \rangle ::= \langle \text{цифра} \rangle \{ \langle \text{цифра} \rangle \}$
  - $\langle \text{операция} \rangle ::= + \mid - \mid * \mid / \mid \%$

# Проблеми при работа с низове

- Излизане извън буфера (buffer overflow)
  - ~~char a[10] = "Hello, world!";~~
  - char b[] = "Hello,", c[] = " world!";
  - ~~streat(b, c);~~
  - ~~strcpy(b, c);~~
- Нетерминирани низове (non-terminated strings)
  - char word[5] = { 'H', 'e', 'l', 'l', 'o' };
  - ~~cout << strlen(word);~~
  - char word2[10];
  - ~~strcpy(word2, word);~~

# Ограничени операции

- **strncpy**(<буфер>, <низ>, *n*)
  - копира първите *n* символа на <низ> в <буфер>
  - винаги записва точно *n* символа в <буфер>, допълвайки с '\0' при нужда
  - ако <низ> има повече от *n* символа, не записва '\0'!
  - връща <буфер>
- **strncat**(<низ<sub>1</sub>>, <низ<sub>2</sub>>, *n*)
  - конкатенира първите *n* символа на <низ<sub>2</sub>> след <низ<sub>1</sub>>
  - винаги поставя '\0' на края
  - все още е отговорност на програмиста да осигури достатъчно място в <низ<sub>1</sub>>!
  - връща <низ<sub>1</sub>>
- **strncmp**(<низ<sub>1</sub>>, <низ<sub>2</sub>>, *n*)
  - сравнява първите *n* символа на <низ<sub>1</sub>> и <низ<sub>2</sub>>
  - връща < 0, 0 или > 0, също като strcmp

# Многомерни масиви

Масив, чиито елементи...	наричаме
...са масиви,	многомерен масив
...не са масиви,	едномерен масив
...са $n$ -мерни масиви,	$n + 1$ -мерен масив

- `<тип> <идентификатор> [[<константа>]] { [<константа>] } [ = { <израз> { , <израз> } } ] ;`
- първата размерност може да бъде изпусната, ако е даден инициализиращ списък
- **Примери:**
  - `int a[2][3] = {{1, 2, 3}, {4, 5, 6}};`
  - `double b[5][6] = {0.1, 0.2, 0.3, 0.4};`
  - `int c[4][5] = {{1, 2}, {3, 4, 5, 6}, {7, 8, 9}, {10}};`
  - `float f[][2][3] = {{{1.2, 2.3, 3.4}, {4.5, 5.6, 6.7}},  
{{7.8, 8.9, 9.1}, {1.2, 3.4, 3.4}},  
{{5.6}, {6.7, 7.8}}};`

# Физическо представяне на многомерни масиви

a											
a[0]						a[1]					
a[0][0]			a[0][1]			a[1][0]			a[1][1]		
a[0][0][0]	a[0][0][1]	a[0][0][2]	a[0][1][0]	a[0][1][1]	a[0][1][2]	a[1][0][0]	a[1][0][1]	a[1][0][2]	a[1][1][0]	a[1][1][1]	a[1][1][2]

## Задачи за многомерни масиви

- Да се въведе от клавиатурата матрица от числа
- Да се изведе на екрана матрица от числа
- Да се транспонира правоъгълна матрица от числа
- Да се намерят сумите на всяка колона в матрица от числа
- Да се намерят редовете в матрица от числа, в които се среща  $x$
- Да се провери дали в матрица от цели числа има колона, чиито най-малък елемент е четно число
- Да се изведат елементите на дадена матрица от числа по диагонали
- Да се слоят “шахматно” две матрици от числа с еднакви размерности



## Обхождане на матрици

				11
			7	12
		4	8	13
	2	5	9	14
1	3	6	10	15

1	3	6	10	15
2	5	9	14	
4	8	13		
7	12			
11				

1	3	6	10	15
2	5	9	14	19
4	8	13	18	22
7	12	17	21	24
11	16	20	23	25

1				
2	4			
3	6	9		
5	8	11	13	
7	10	12	14	15

1	4	9	16	25
2	3	8	15	24
5	6	7	14	23
10	11	12	13	22
17	18	19	20	21

25	10	11	12	13
24	9	2	3	14
23	8	1	4	15
22	7	6	5	16
21	20	19	18	17