

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

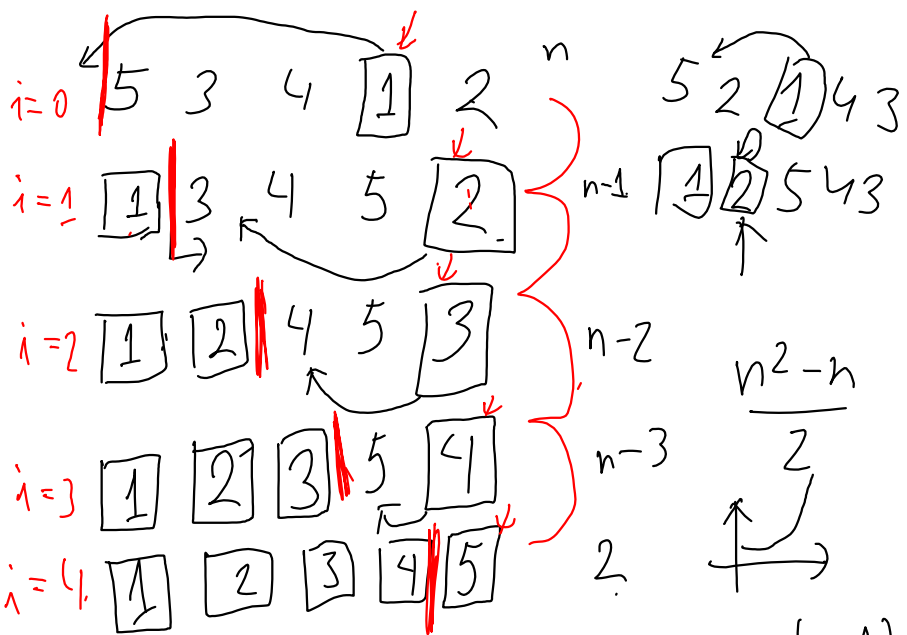
- Да се въведе масив от числа
- Да се изведе масив от числа
- Да се намери сумата на числата в даден масив
- Да се провери дали дадено число се среща в масив
- Да се провери дали числата в масив нарастват монотонно
- Да се провери дали всички числа в даден масив са различни

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

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

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

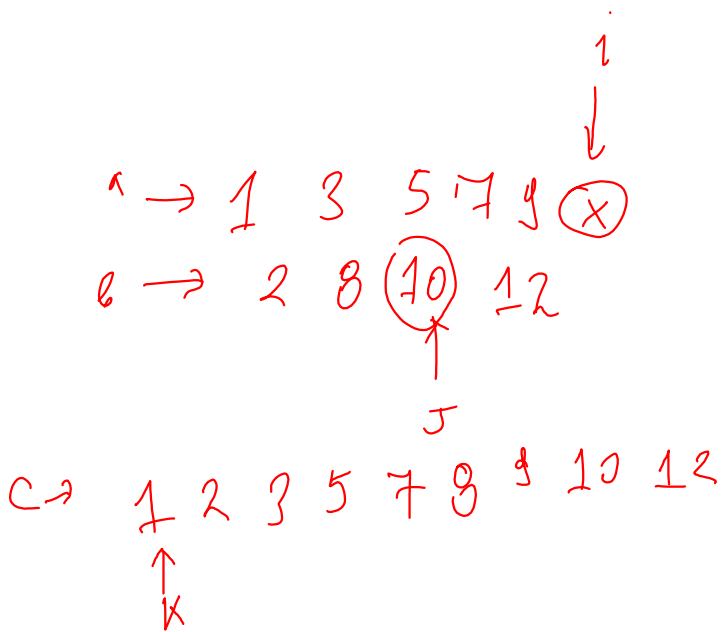


$$2 + 3 + \dots + n = \frac{n(n-1)}{2}$$

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

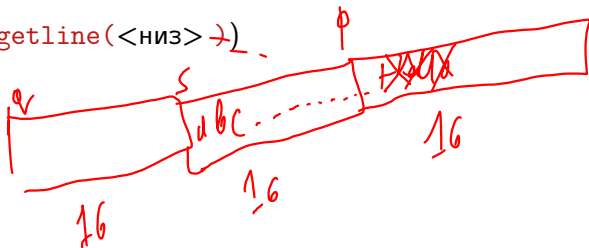
- **Описание:** **Низ** наричаме последователност от символи
 - последователност от 0 символи наричаме **празен низ**
- **Представяне в C++:** Масив от символи (**char**), в който след последния символ в низа е записан **терминиращият символ** `'\0'`
 - `'\0'` е първият символ в ASCII таблицата, с код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

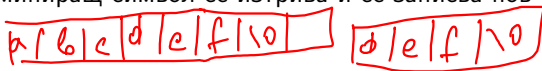
- `strcat(<низ1>, +=<низ2>)`
 - **конкатенация** (слепване) на низове

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>
- Handwritten notes:*
 ↪ true (next to the first bullet of strchr)
 ↪ false (next to the last bullet of strchr)

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>

Задачи за низове

- Да се провери дали даден низ е **палиндром**

Задачи за низове

- Да се провери дали даден низ е **палиндром**
 - чете се еднакво в двете посоки

Задачи за низове

$$n=4$$

$$i=1$$

$$n=7$$

$$i=2$$

$$n=6$$

$$i=2$$

$$n=5$$

$$i=1$$



- Да се провери дали даден низ е палиндром

- чете се еднакво в двете посоки

- "abba", "racecar", "risetovotesir", "wasitacaroracatisaw"

$$0 \leftrightarrow n-1$$

$$1 \leftrightarrow n-2$$

$$2 \leftrightarrow n-3$$

.....

$$+ = n-1$$

$$0?$$

$$n-1-i$$

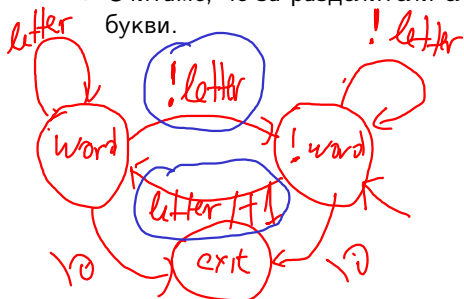
$$m = \left\lfloor \frac{n}{2} \right\rfloor - 1$$

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

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.



Hello, world!

I have 2 apples.

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

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
 - $\langle \text{израз} \rangle ::= \langle \text{число} \rangle \{ \langle \text{операция} \rangle \langle \text{число} \rangle \} =$

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
 - $\langle \text{израз} \rangle ::= \langle \text{число} \rangle \{ \langle \text{операция} \rangle \langle \text{число} \rangle \} =$
 - $\langle \text{число} \rangle ::= \langle \text{цифра} \rangle \{ \langle \text{цифра} \rangle \}$

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
 - $\langle \text{израз} \rangle ::= \langle \text{число} \rangle \{ \langle \text{операция} \rangle \langle \text{число} \rangle \} =$
 - $\langle \text{число} \rangle ::= \langle \text{цифра} \rangle \{ \langle \text{цифра} \rangle \}$
 - $\langle \text{операция} \rangle ::= + \mid - \mid * \mid / \mid \%$

1 2 3

$$((23 + 5) - 14) / 87 =$$

Проблеми при работа с низове

- Излизане извън буфера (buffer overflow)

Проблеми при работа с низове

- Излизане извън буфера (buffer overflow)
 - ~~char a[10] = "Hello, world!"~~

Проблеми при работа с низове

- Излизане извън буфера (buffer overflow)
 - ~~char a[10] = "Hello, world!"~~
 - char b[] = "Hello,", c[] = " world!";

Проблеми при работа с низове

- Излизане извън буфера (buffer overflow)
 - ~~char a[10] = "Hello, world!"~~
 - char b[4] = "Hello,", c[4] = " world!";
 - ~~strcat(b, c);~~

Проблеми при работа с низове

- Излизане извън буфера (buffer overflow)
 - ~~char a[10] = "Hello, world!"~~
 - char b[] = "Hello,", c[] = " world!";
 - ~~strcat(b, c);~~
 - ~~strcpy(b, c);~~

Проблеми при работа с низове

- Излизане извън буфера (buffer overflow)
 - ~~char a[10] = "Hello, world!"~~
 - char b[] = "Hello,", c[] = " world!";
 - ~~strcat(b, c);~~
 - ~~strcpy(b, c);~~
- Нетерминирани низове (non-terminated strings)

Проблеми при работа с низове

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

Проблеми при работа с низове

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

word

H	e	l	l	o		B	t	y		e	l	l	o
---	---	---	---	---	--	---	---	---	--	---	---	---	---

:

Проблеми при работа с низове

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

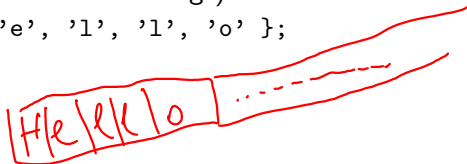
Проблеми при работа с низове

- Излизане извън буфера (buffer overflow)

- ~~char a[10] = "Hello, world!";~~
- char b[] = "Hello,", c[] = " world!";
- ~~strcat(b, c);~~
- ~~strcpy(b, c);~~

- Нетерминирани низове (non-terminated strings)

- char word[5] = { 'H', 'e', 'l', 'l', 'o' };
- ~~cout << strlen(a);~~
- char b[10];
- ~~strcpy(b, a);~~



Ограничени операции

- `strncpy` (<буфер>, <низ>, n)

Ограничени операции

- `strncpy(<буфер>, <низ>, n)`
 - копира първите *n* символа на <низ> в <буфер>

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

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'`!

Handwritten diagram illustrating the behavior of `strncpy`. The word "Hello" is written in red. A red bracket underneath it spans the length of the word and is labeled with the number "10". To the right of "Hello", there are several red dashes representing the rest of the buffer. Below the word "Hello" and the dashes, the code `strncpy = '\0';` is written in red, indicating that the function stops copying when it reaches the end of the source string, even if the buffer is not full.

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>

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**(<низ₁>, <низ₂>, *n*)

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**(<низ₁>, <низ₂>, *n*)
 - конкатенира първите *n* символа на <низ₂> след <низ₁>

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**(`<низ1>`, `<низ2>`, `n`)
 - конкатенира първите `n` символа на `<низ2>` след `<низ1>`
 - винаги поставя `'\0'` на края

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**(<низ₁>, <низ₂>, *n*)
 - конкатенира първите *n* символа на <низ₂> след <низ₁>
 - винаги поставя '\0' на края
 - все още е отговорност на програмиста да осигури достатъчно място в <низ₁>!

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(<низ1>, <низ2>, n)`
 - конкатенира първите n символа на `<низ2>` след `<низ1>`
 - винаги поставя `'\0'` на края
 - все още е отговорност на програмиста да осигури достатъчно място в `<низ1>!`
 - връща `<низ1>`

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`(`<низ1>`, `<низ2>`, `n`)
 - конкатенира първите `n` символа на `<низ2>` след `<низ1>`
 - винаги поставя `'\0'` на края
 - все още е отговорност на програмиста да осигури достатъчно място в `<низ1>!`
 - връща `<низ1>`
- `strncmp`(`<низ1>`, `<низ2>`, `n`)

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**(`<низ1>`, `<низ2>`, `n`)
 - конкатенира първите `n` символа на `<низ2>` след `<низ1>`
 - винаги поставя `'\0'` на края
 - все още е отговорност на програмиста да осигури достатъчно място в `<низ1>!`
 - връща `<низ1>`
- **strncmp**(`<низ1>`, `<низ2>`, `n`)
 - сравнява първите `n` символа на `<низ1>` и `<низ2>`

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**(<низ₁>, <низ₂>, *n*)
 - конкатенира първите *n* символа на <низ₂> след <низ₁>
 - винаги поставя '\0' на края
 - все още е отговорност на програмиста да осигури достатъчно място в <низ₁>!
 - връща <низ₁>
- **strncmp**(<низ₁>, <низ₂>, *n*)
 - сравнява първите *n* символа на <низ₁> и <низ₂>
 - връща -1, 0 или 1, също като strcmp

Многомерни масиви

Масив, чиито елементи...

...са масиви,

наричаме

многомерен масив

Многомерни масиви

Масив, чиито елементи...	наричаме
...са масиви,	многомерен масив
...не са масиви,	едномерен масив

Многомерни масиви

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

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

Многомерни масиви

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

- $\langle \text{тип} \rangle \langle \text{идентификатор} \rangle \left[\left[\langle \text{константа} \rangle \right] \right] \left\{ \left[\langle \text{константа} \rangle \right] \right\}$
 $\left[= \left\{ \langle \text{константа} \rangle \left\{ , \langle \text{константа} \rangle \right\} \right\} \right] ;$
- първата размерност може да бъде изпусната, ако е даден инициализиращ списък

Многомерни масиви

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

- $\langle \text{тип} \rangle \langle \text{идентификатор} \rangle \left[\left[\langle \text{константа} \rangle \right] \right] \left\{ \left[\langle \text{константа} \rangle \right] \right\}$
 $\left[= \left\{ \langle \text{константа} \rangle \left\{ , \langle \text{константа} \rangle \right\} \right\} \right] ;$
- първата размерност може да бъде изпусната, ако е даден инициализиращ списък
- **Примери:**

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

- първата размерност може да бъде изпусната, ако е даден инициализиращ списък

- **Примери:**

- `int a[2][3] = {{1, 2, 3}, {4, 5, 6}};`

```

  ( 1 2 3 )
  ( 4 5 6 )
  
```

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

- 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};` →

↓

$$\begin{pmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0 & 0 \\ 0 & & & & & \end{pmatrix}$$

Многомерни масиви

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

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 3 & 4 & 5 & 6 & 0 \\ 7 & 8 & 9 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 \end{pmatrix}$$

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

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

- 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

$a[2][2][3]$ $b[i]$
 $a[i][j][k]$

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]

$a[0][j]$

$$a[0] = \begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}$$

$a[0][i]$

$a[1][0]$

$$a[1] = \begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}$$

$a[1][k]$

Задачи за многомерни масиви

- Да се въведе от клавиатурата матрица от числа



Задачи за многомерни масиви

- Да се въведе от клавиатурата матрица от числа
- Да се изведе на екрана матрица от числа

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

$$a[i][j]$$


$$a[j][i]$$
~~$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$~~

$$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$
~~$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$~~

Hand-drawn red arrows in the matrix above show the mapping of elements from the original matrix to the transposed matrix. Arrows point from (1,2) to (2,1), (2,3) to (3,2), (3,1) to (1,3), (3,2) to (2,3), and (3,3) to (3,3). The labels 'i < j' and 'i = j' are written at the bottom left and right respectively.

$$a[j][i],$$

$$i < j$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

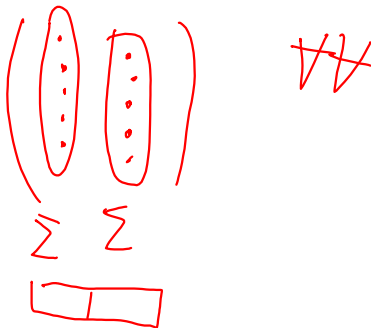
$$\begin{pmatrix} 1 & 4 & ? \\ 2 & 5 & ? \\ 3 & 6 & ? \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & ? \\ 3 & 4 & ? \\ 5 & 6 & ? \end{pmatrix}$$

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \\ \boxed{5 & 6} \end{pmatrix} \begin{matrix} ! \\ 0 \\ ! \\ ! \\ ! \end{matrix}$$

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



Задачи за многомерни масиви

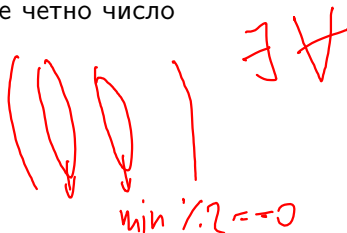
- Да се въведе от клавиатурата матрица от числа
- Да се изведе на екрана матрица от числа
- Да се транспонира правоъгълна матрица от числа
- Да се намерят сумите на всяка колона в матрица от числа
- Да се намерят редовете в матрица от числа, в които се среща x

✓ E

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

сума



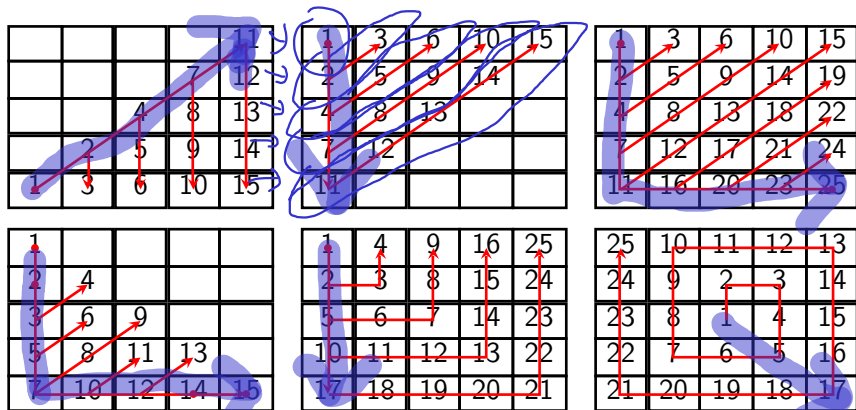
Задачи за многомерни масиви

- 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

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



$a[i][0]$ $a[i-1][1]$ $a[i-2][2]$ $a[0][7]$