Масиви и низове

Трифон Трифонов

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

15 ноември–6 декември 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> } }] ;

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 };`

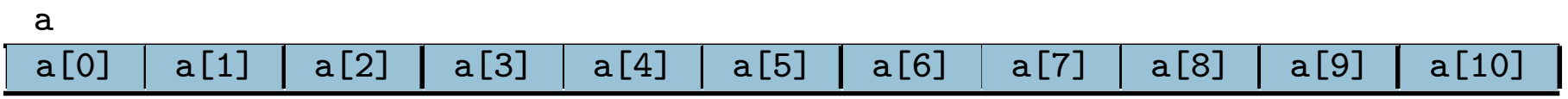
Дефиниция на масив

<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



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

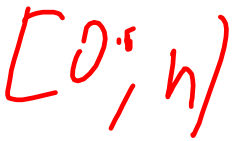
- Да се въведе масив от числа

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

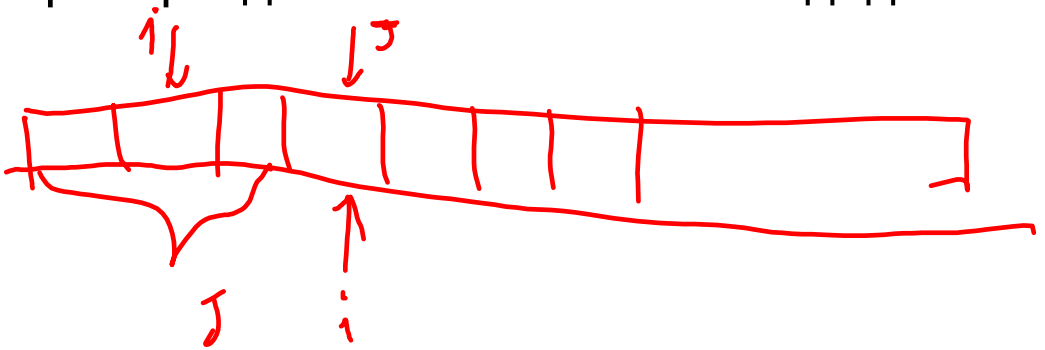
Задачи за масиви

- 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

$\forall i \quad a[i] \leq a[i+1]$

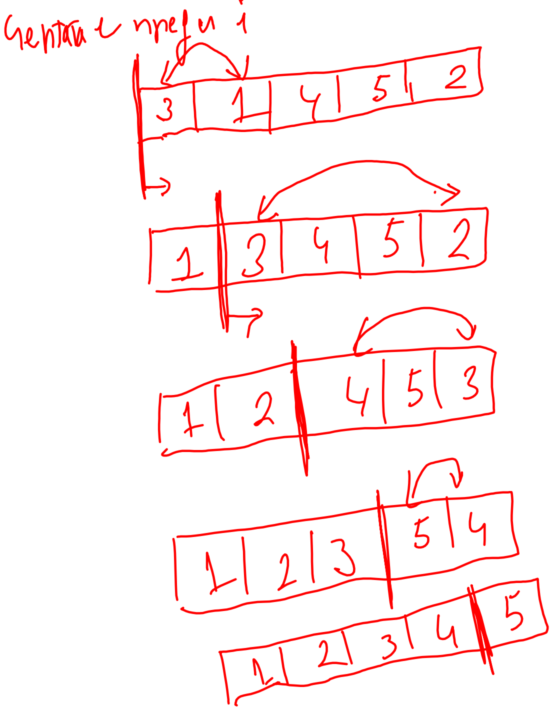
Задачи за масиви

- 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

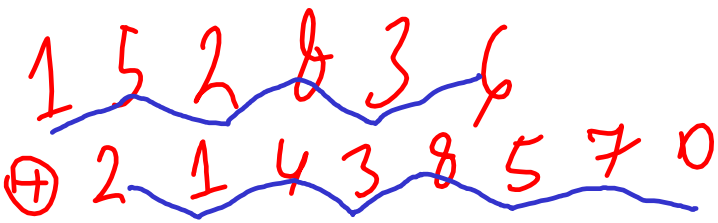


Задачи за масиви

- 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

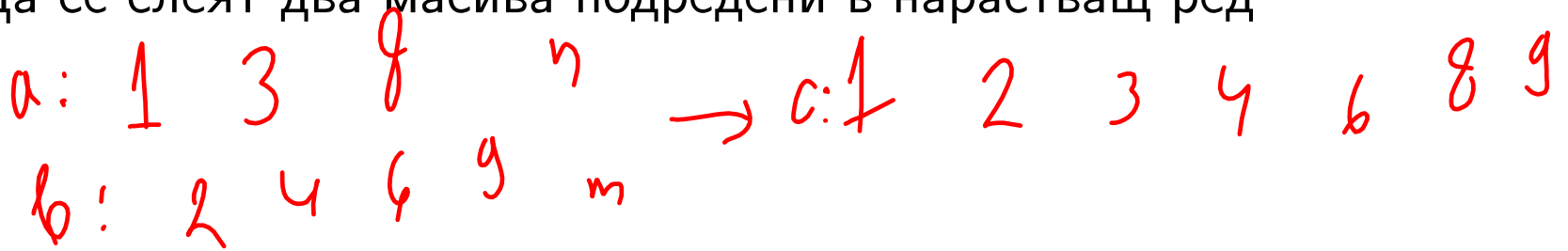


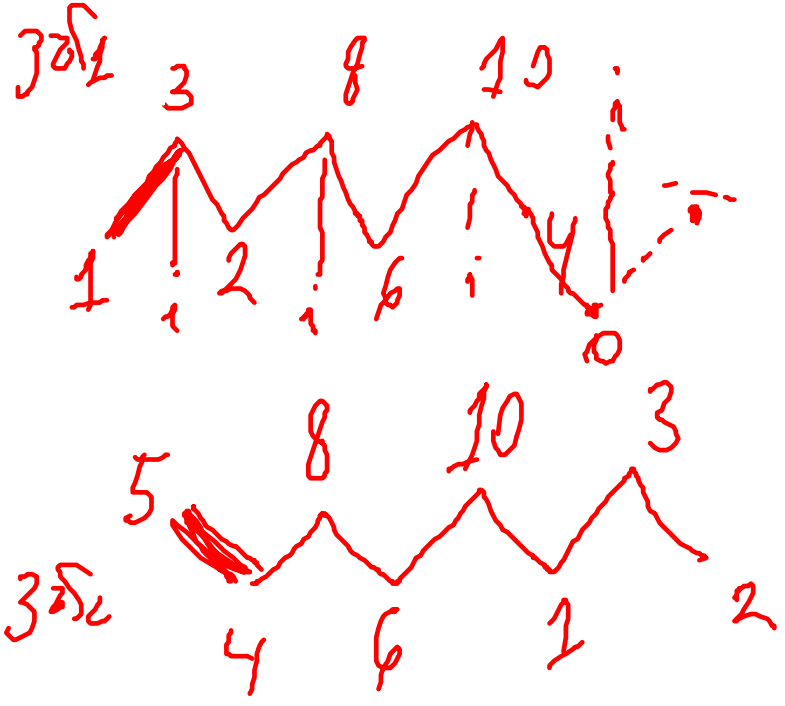
Задачи за масиви



Вярно ли е, че числата (в ⊕) в масив образуват „грион"

- 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





$\forall x (3\sigma_1(x)) \vee \forall x (3\sigma_2(x))$

$\Downarrow$

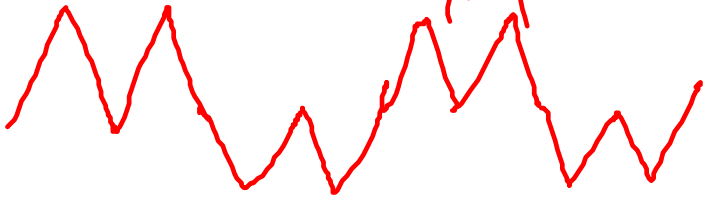
$\forall x (3\sigma_1(x) \vee 3\sigma_2(x))$

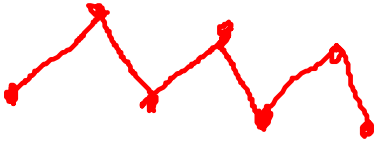
$\Downarrow$

max $x \leftarrow 1$!

$\forall x\, p(x) \Leftrightarrow p(a_1) \,||\, p(a_2) \,||\, \cdots \,||\, p(a_n)$

$\exists x\, p(x) \Leftrightarrow p(a_1) \,||\, p(a_2) \,||\, \cdots \,||\, p(a_n)$





a: 1 3 8 n

b: 2 4 6 9 m

→ c: 1 2 3 4 6 8 9

1 2 3 4 6 8 9

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

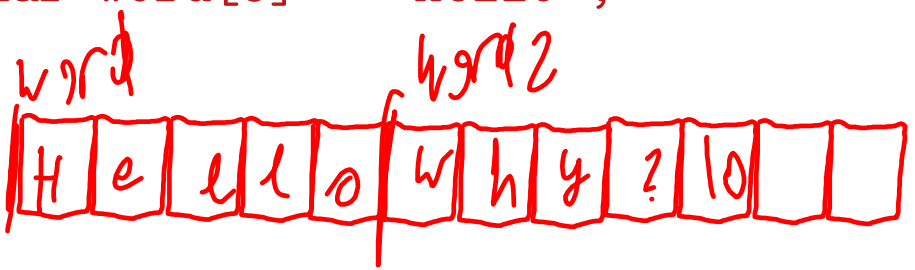
- **Описание:** Низ наричаме последователност от символи
  - последователност от 0 символи наричаме празен низ
- **Представяне в C++:** Масив от символи (`char`), в който след последния символ в низа е записан терминиращият символ `'\0'`
  - `'\0'` е първият символ в ASCII таблицата, с код 0
- **Примери:**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

```
char a[]; // !!!
```

```
char *a;
```

```
char a[N], b[M];
```

```
i<N && i<M
```

Операции за работа с низове

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

Операции за работа с низове

- Вход (`>>`, `cin.getline(`<низ>, <число>`)`)
  - `>>` въвежда до разделител (интервал, табулация, нов ред)

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

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 (`<<`)

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 (`[]`)

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`~~)

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`~~)

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...

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>)`

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

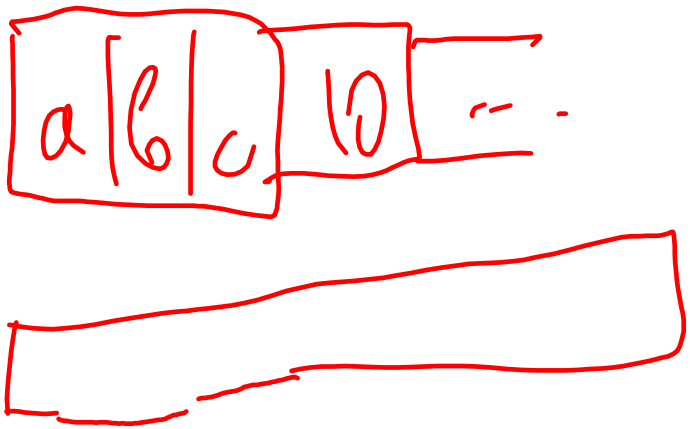
```
#include <cstring>
```

- `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>
```

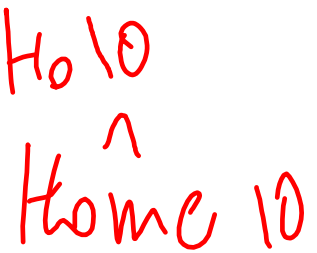
- `strlen(<низ>)`
  - връща дължината на <низ>, т.е. броя символи без '\0'
- `strcpy(<буфер>, <низ>)`
  - прехвърля всички символи от <низ> в <буфер>
  - връща <буфер>
  - отговорност на програмиста е да осигури, че в <буфер> да има достатъчно място да поеме всички символи на <низ>



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 число < 0, 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 число < 0, 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 число < 0, ако <низ$_1$> е преди <низ$_2$>
  - връща 0, ако <низ$_1$> съвпада с <низ$_2$>
  - връща число > 0, 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 число < 0, ако <низ$_1$> е преди <низ$_2$>
  - връща 0, ако <низ$_1$> съвпада с <низ$_2$>
  - връща число > 0, 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 лексикографски (речникова наредба)
  - връща число < 0, ако <низ$_1$> е преди <низ$_2$>
  - връща 0, ако <низ$_1$> съвпада с <низ$_2$>
  - връща число > 0, 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>

Задачи за низове

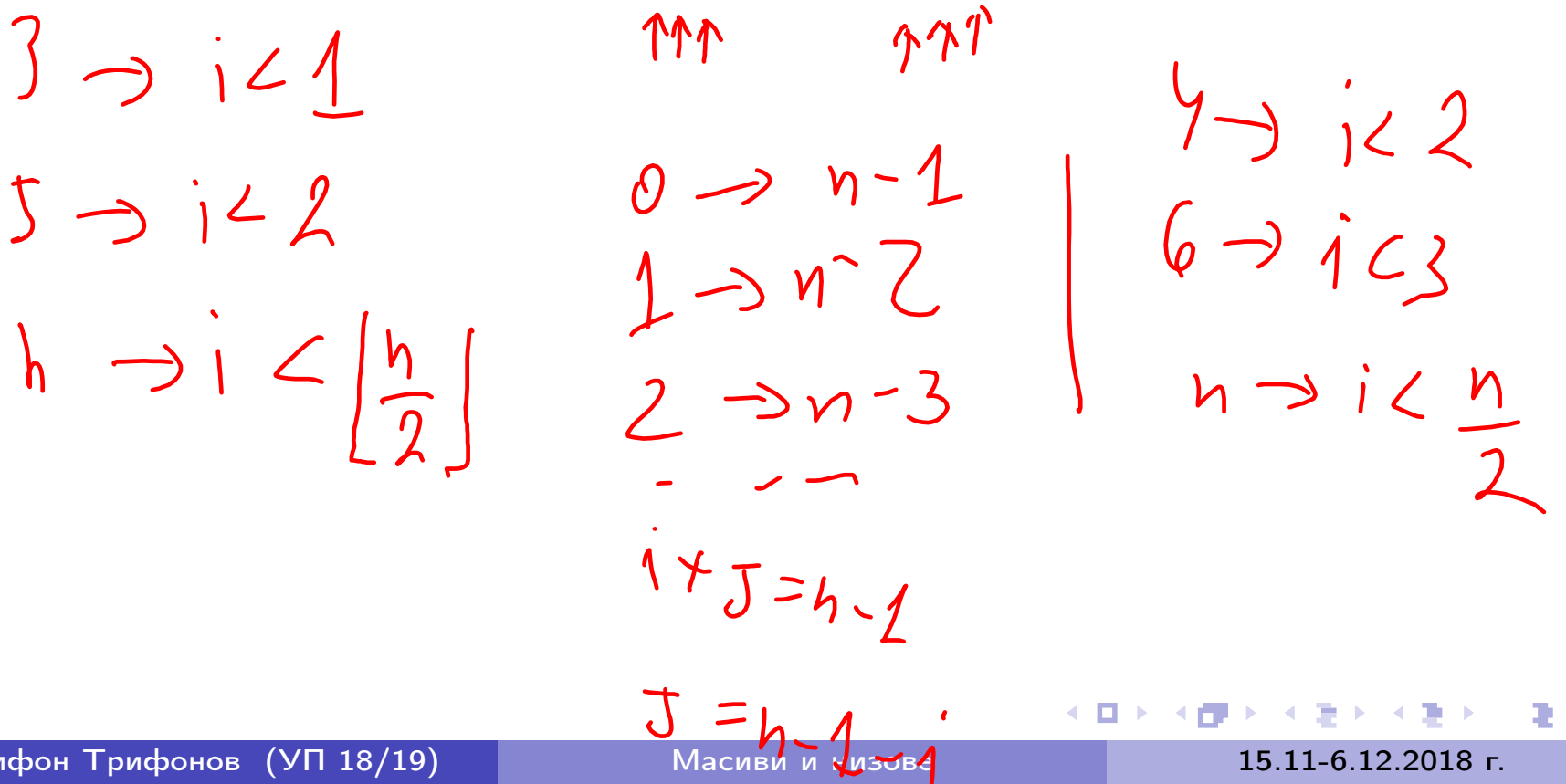
- Да се провери дали даден низ е палиндром

Задачи за низове

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

Задачи за низове

- Да се провери дали даден низ е палиндром
  - чете се еднакво в двете посоки
  - "abba", "racecar", "risetovotesir", "wasitacaroracatisaw"

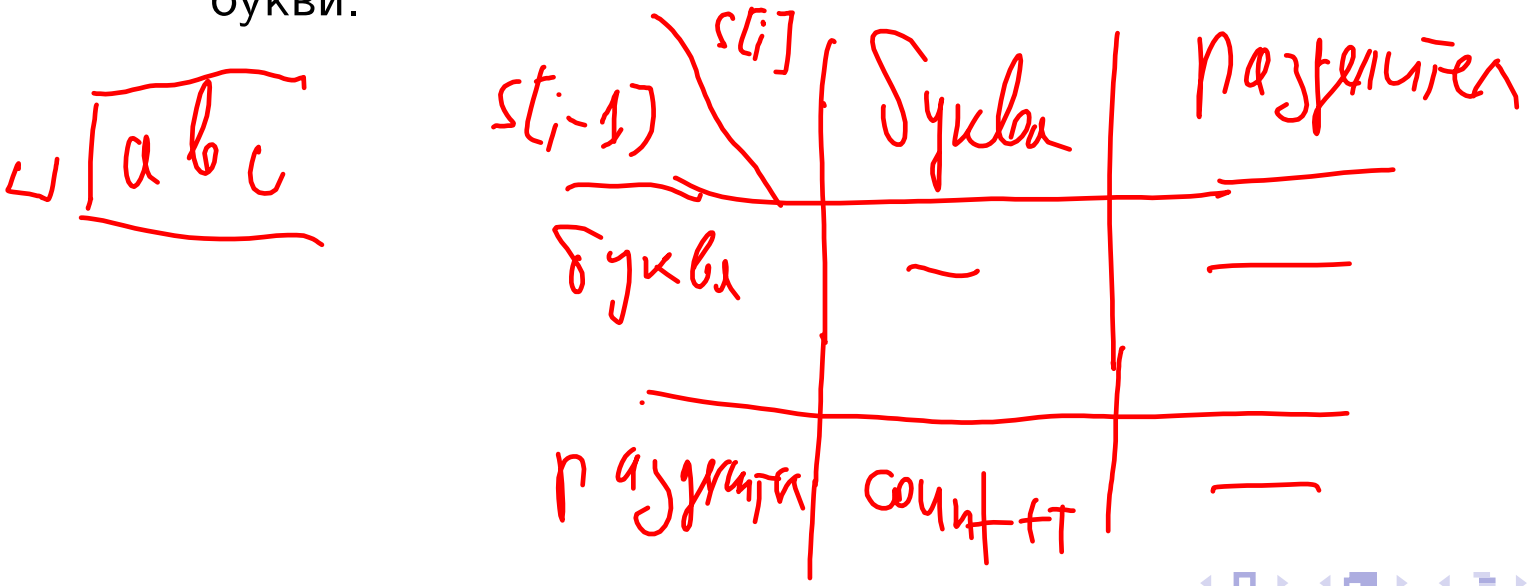


Задачи за низове

- 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.



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

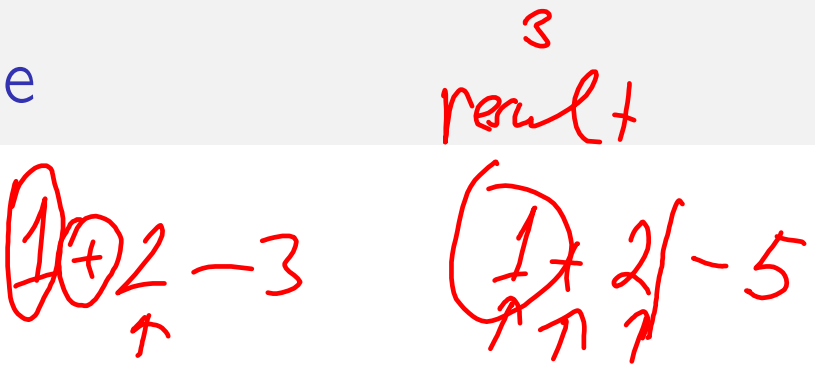
Задачи за низове

- 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
  - <израз> ::= <число>{<операция><число>}=

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
  - <израз> ::= <число>{<операция><число>}=
  - <число> ::= <цифра>{<цифра>}

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
  - <израз> ::= <число>{<операция><число>}=
  - <число> ::= <цифра>{<цифра>}
  - <операция> ::= + | - | * | / | %

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[] = "Hello,", c[] = " world!";`
  - ~~`strcat(b, c);`~~

Проблеми при работа с низове

- Излизане извън буфера (buffer overflow)
  - ~~`char a[10] = "Hello, world!"`~~
  - `char b[7] = "Hello,", c[8] = " 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(word);`~~

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(word);`~~
  - `char word2[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(word);`~~
  - `char word2[10];`
  - ~~`strcpy(word2, word);`~~

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'!

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(`<низ$_1$>, <низ$_2$>, 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(`<низ$_1$>, <низ$_2$>, n`)`
  - конкатенира първите *n* символа на <низ$_2$> след <низ$_1$>

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$_1$>, <низ$_2$>, n`)`
  - конкатенира първите *n* символа на <низ$_2$> след <низ$_1$>
  - 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(`<низ$_1$>, <низ$_2$>, n`)`
  - конкатенира първите *n* символа на <низ$_2$> след <низ$_1$>
  - винаги поставя '\0'на края
  - все още е отговорност на програмиста да осигури достатъчно място в <низ$_1$>!

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$_1$>, <низ$_2$>, n`)`
  - конкатенира първите *n* символа на <низ$_2$> след <низ$_1$>
  - винаги поставя '\0'на края
  - все още е отговорност на програмиста да осигури достатъчно място в <низ$_1$>!
  - връща <низ$_1$>

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$_1$>, <низ$_2$>, n`)`
  - конкатенира първите *n* символа на <низ$_2$> след <низ$_1$>
  - винаги поставя '\0'на края
  - все още е отговорност на програмиста да осигури достатъчно място в <низ$_1$>!
  - връща <низ$_1$>
- `strncmp(`<низ$_1$>, <низ$_2$>, 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(`<низ$_1$>, <низ$_2$>, n`)`
  - конкатенира първите *n* символа на <низ$_2$> след <низ$_1$>
  - винаги поставя '\0'на края
  - все още е отговорност на програмиста да осигури достатъчно място в <низ$_1$>!
  - връща <низ$_1$>
- `strncmp(`<низ$_1$>, <низ$_2$>, n`)`
  - сравнява първите *n* символа на <низ$_1$> и <низ$_2$>

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$_1$>, <низ$_2$>, n`)`
  - конкатенира първите *n* символа на <низ$_2$> след <низ$_1$>
  - винаги поставя '\0'на края
  - все още е отговорност на програмиста да осигури достатъчно място в <низ$_1$>!
  - връща <низ$_1$>
- `strncmp(`<низ$_1$>, <низ$_2$>, n`)`
  - сравнява първите *n* символа на <низ$_1$> и <низ$_2$>
  - връща < 0, 0 или > 0, също като `strcmp`

Многомерни масиви

Масив, чиито елементи...	наричаме
...са масиви,	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

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

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

Многомерни масиви

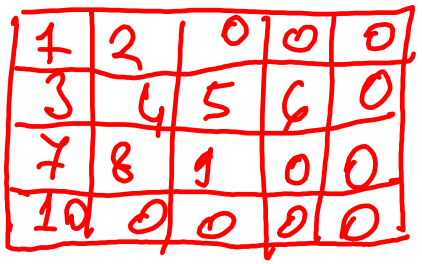
Масив, чиито елементи...	наричаме
...са масиви,	многомерен масив
...не са масиви,	едномерен масив
...са n-мерни масиви,	$n+1$-мерен масив

- <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};`

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}};`



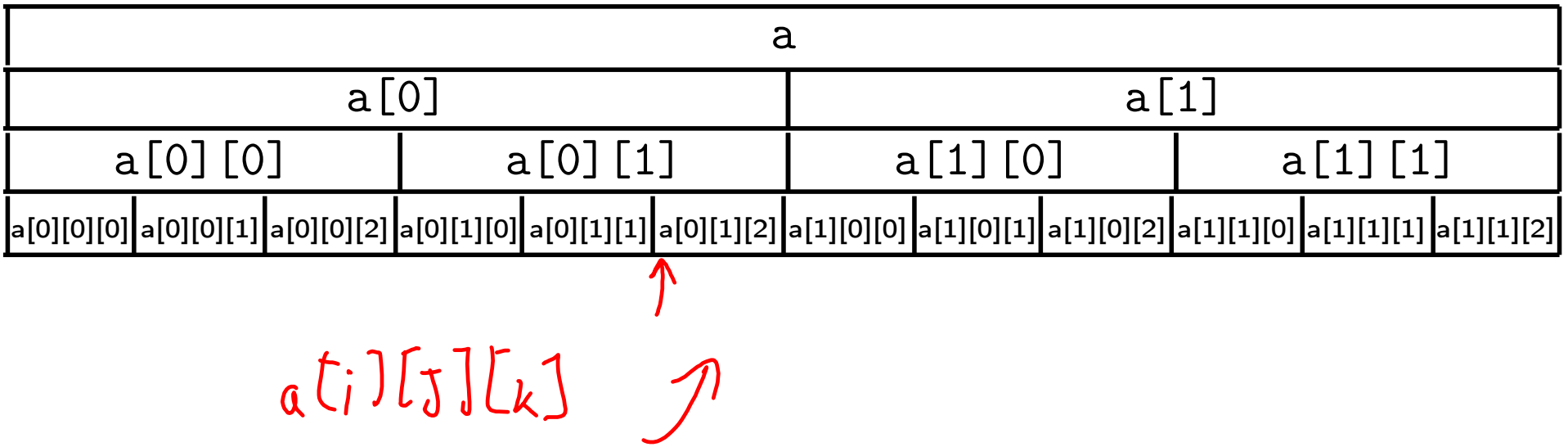
Многомерни масиви

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

```
int a[2][2][3];
```



Задачи за многомерни масиви

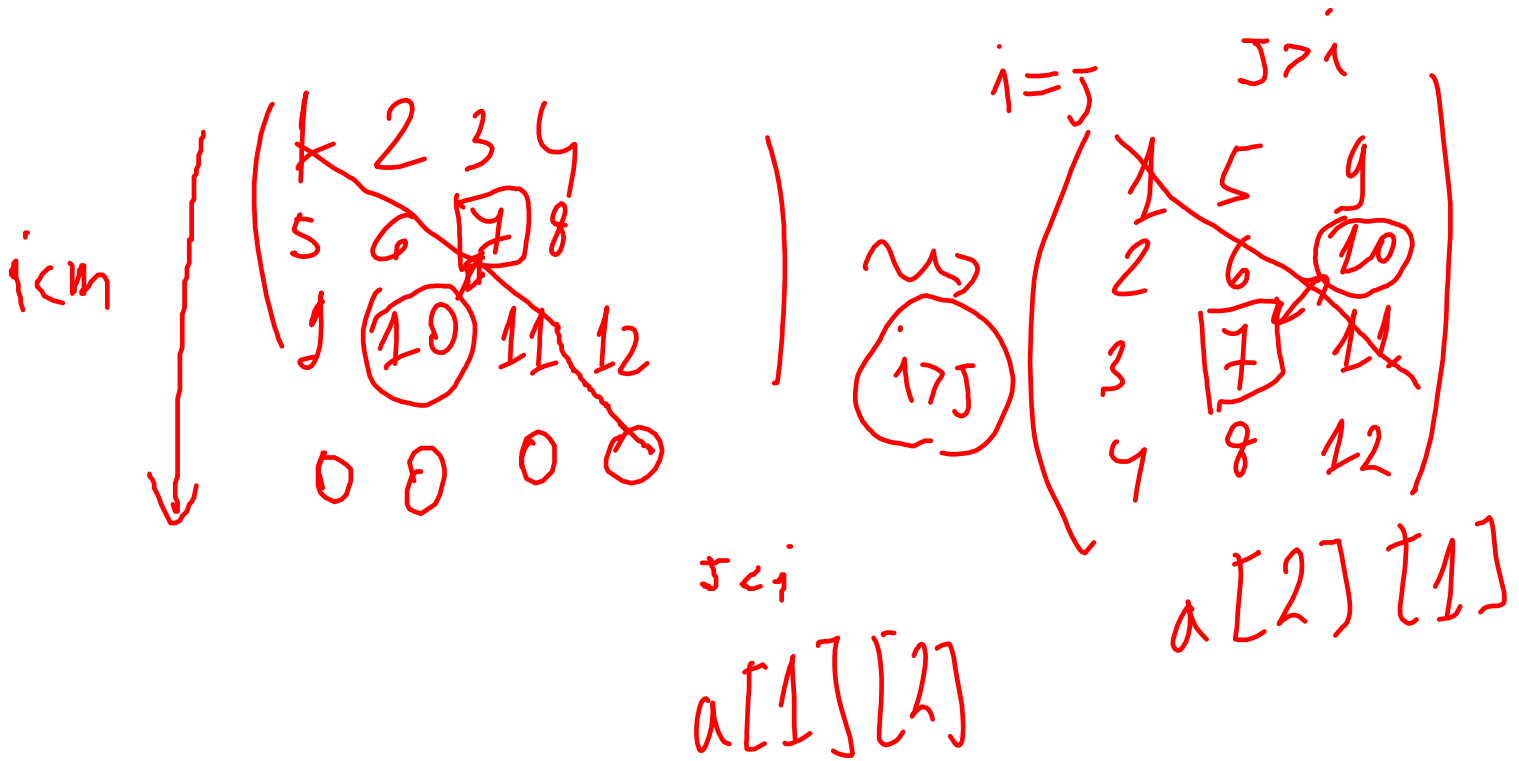
- Да се въведе от клавиатурата матрица от числа

Задачи за многомерни масиви

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

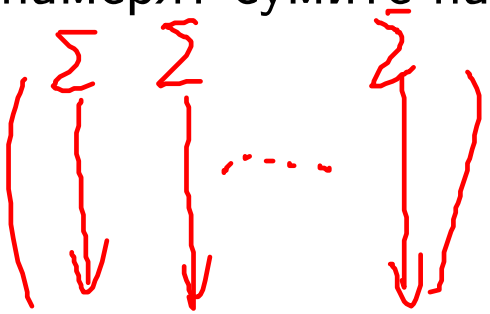
Задачи за многомерни масиви

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



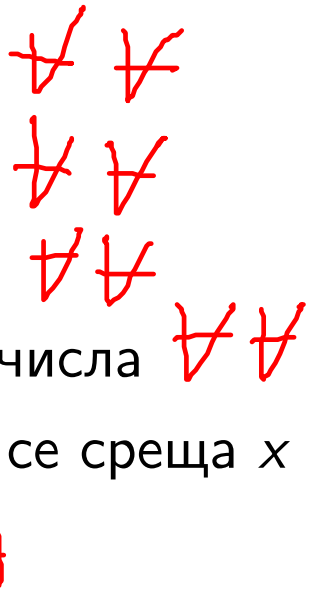
Задачи за многомерни масиви

- 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



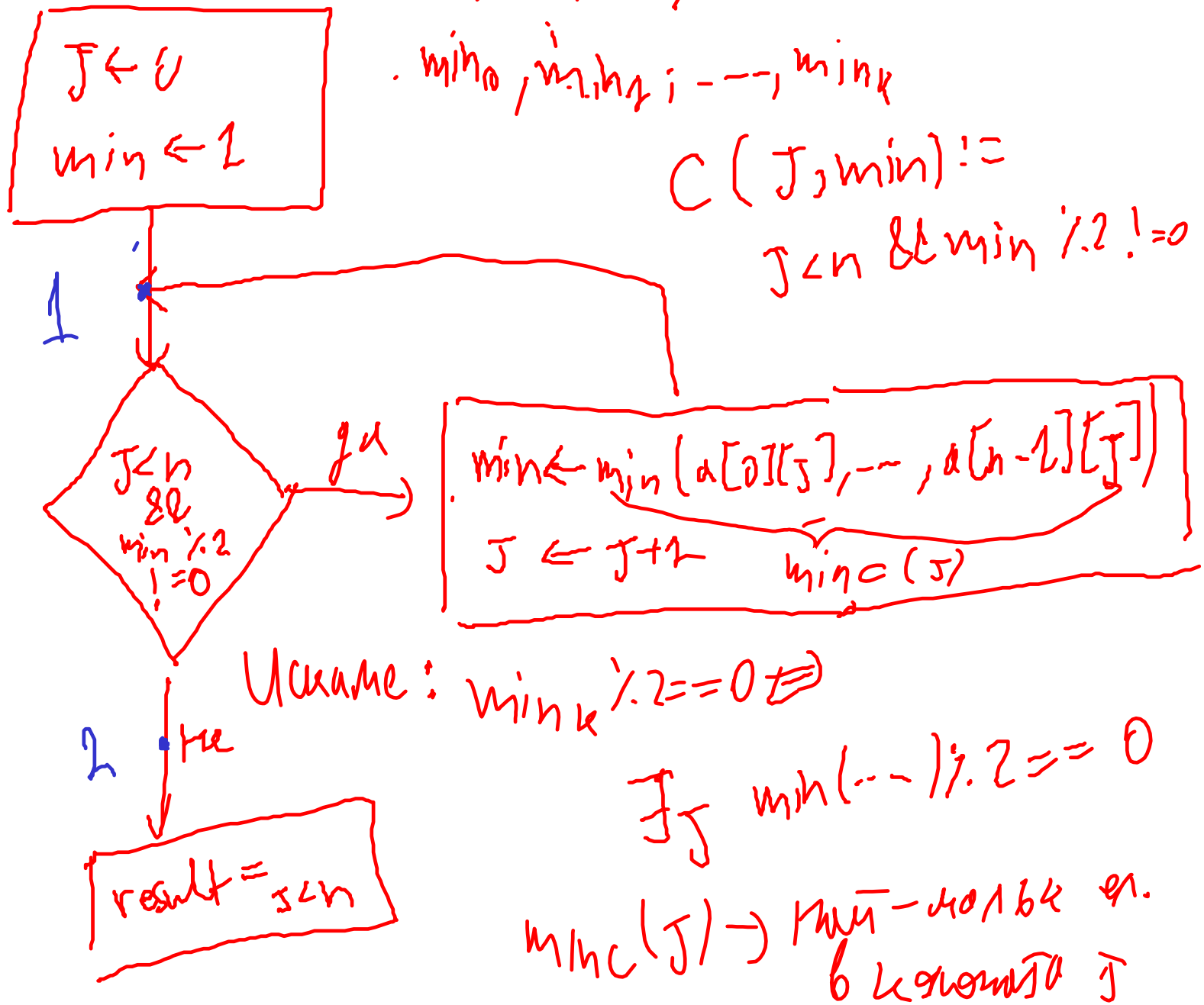
Задачи за многомерни масиви

- 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



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



$J_0, J_1, \dots, J_k$

$min_0, min_1, \dots, min_k$

C(j, min) :=
j<n && min %2 != 0

1

2

Имаме: $min_k \%2 == 0 \Rightarrow$

$\exists j \; min(\dots) \%2 == 0$

minc(j) → най-малък ел. в колоната j

$C(j, min) = j < n \;\&\&\; min \,\%\, 2 != 0$

$((j_0, min_0), \ldots, (j_{k-1}, min_{k-1})$

$\neg C(j_k, min_k)$

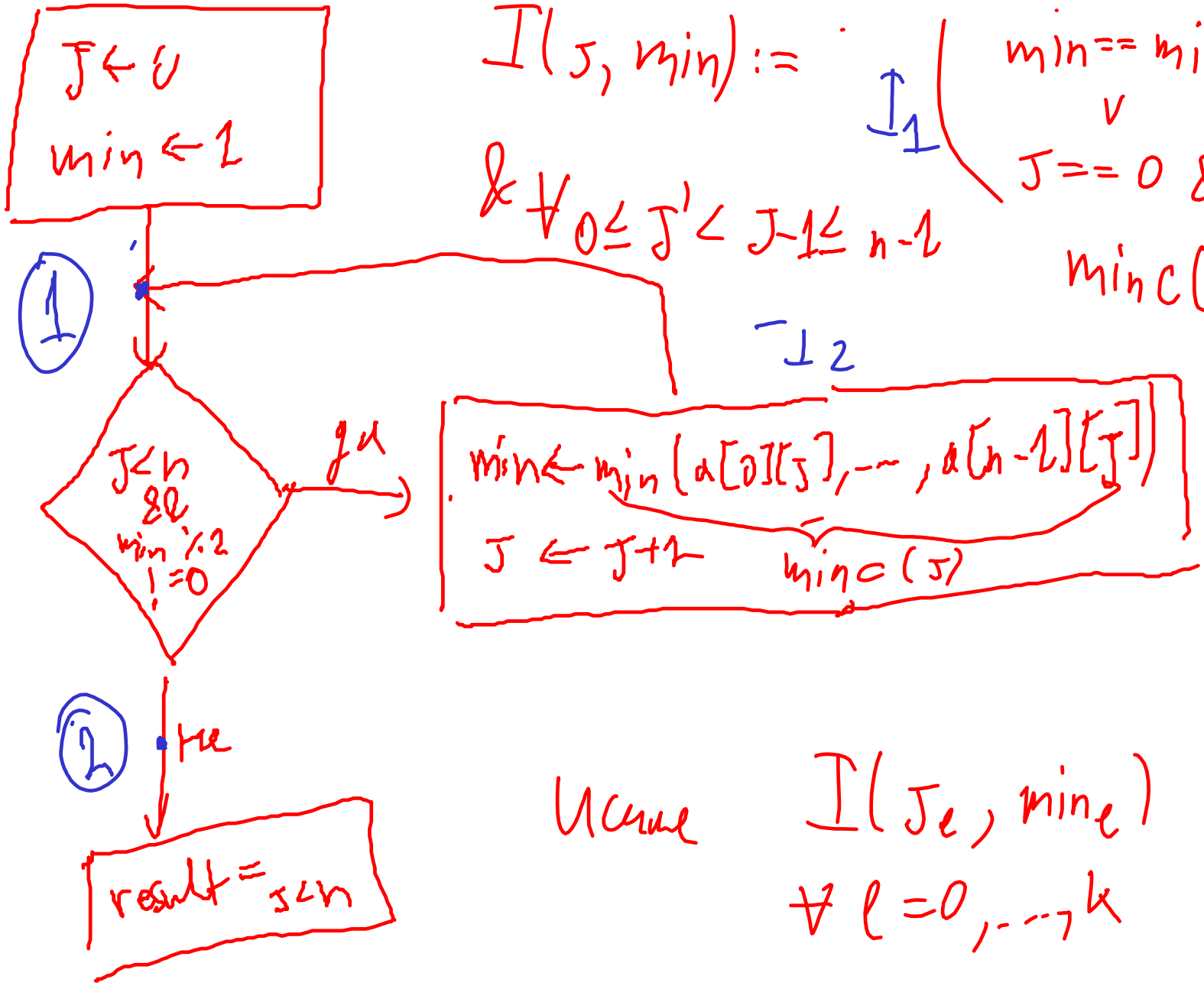
$I(j, min)$ → вижте вече в (1).

$I(j_k, min_k) \;\&\; \neg C(j_k, min_k) \rightarrow$

$min_k \,\%\, 2 == 0 \leftrightarrow \exists j \; min(j) \,\%\, 2 == 0$

$I(j_k, min_k) = true$ е инвариант!

не е безполезен



$I(j, min) :=$

$I_1 \left(\begin{array}{c} min == minc(j-1) \\ \vee \\ j == 0 \;\&\; min == 1 \end{array} \right)$

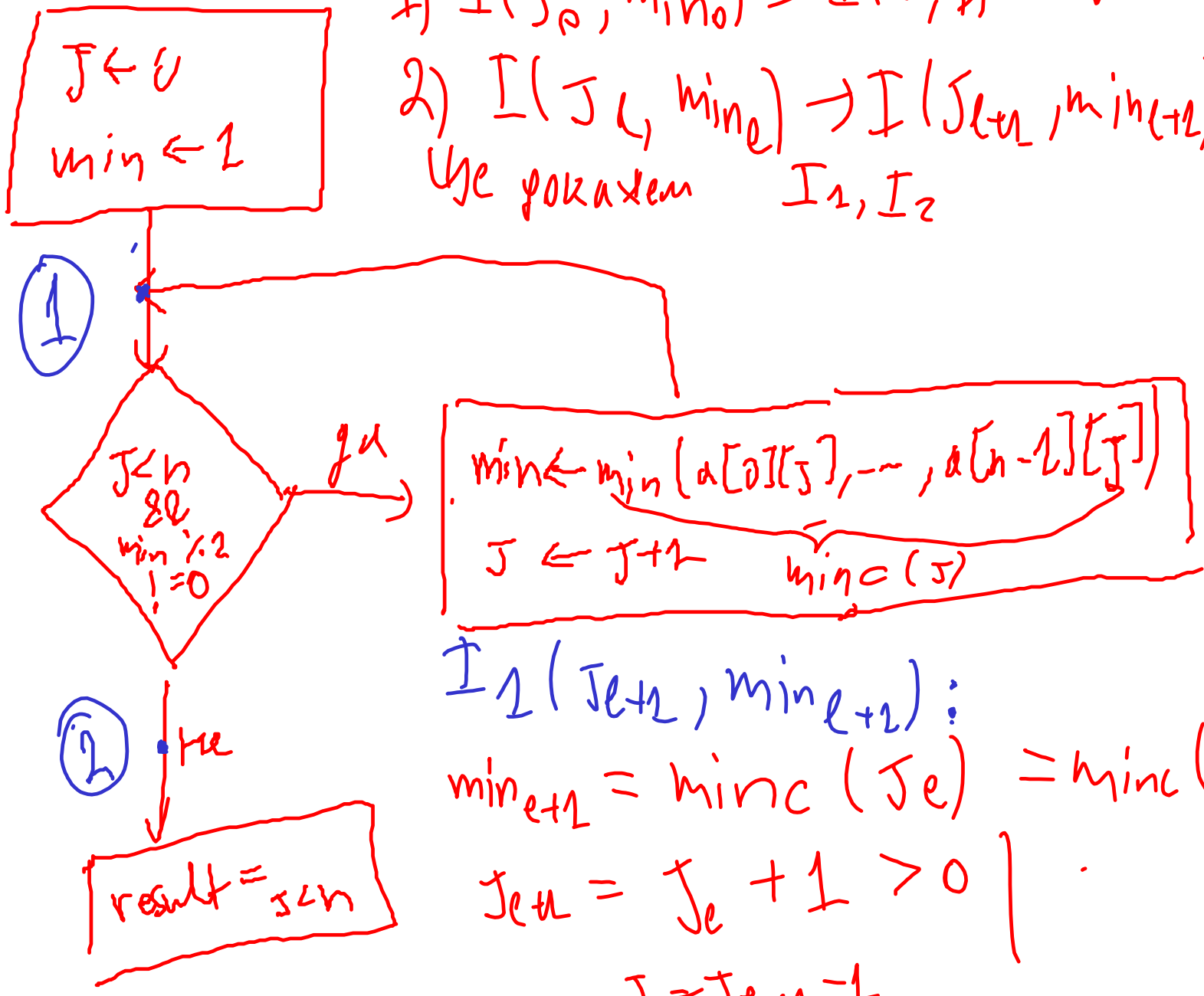
$\& \forall 0 \le j' < j-1 \le n-1$

$minc(j') \,\%\, 2 == 1$

I_2

Цикл $I(j_\ell, min_\ell)$

$\forall \ell = 0, \ldots, k$



1) $I(J_0, min_0) = I(0,1)$ V

2) $I(J_l, min_l) \to I(J_{l+1}, min_{l+1})$

Уже доказали I_1, I_2

$I_1(J_{l+1}, min_{l+1})$:

$min_{l+1} = minc(J_l) = minc(J_{l+1} - 1)$

$J_{l+1} = J_l + 1 > 0$

$J_l = J_{l+1} - 1$

$(min_\ell == min_C(J_\ell - 1) \vee J_\ell == 0 \;\&\; min_\ell == 1)$ $\&\; J_\ell \le n$

$\&\; \forall 0 \le j' < J_\ell - 1 \; min_C(j') \,\%\, 2 == 1$

$I_2(J_{\ell+1}, min_{\ell+1})$

Или $J_\ell == 0 \rightarrow$ тривиално верно

J_ℓ =

Иначе $\forall 0 \le j' < (J_{\ell+1} - 1) \; min_C(j') \,\%\, 2 == 1$

$\Updownarrow$

$\forall 0 \le j' < J_\ell - 1 \; min_C(j') \,\%\, 2 == 1$ ✓

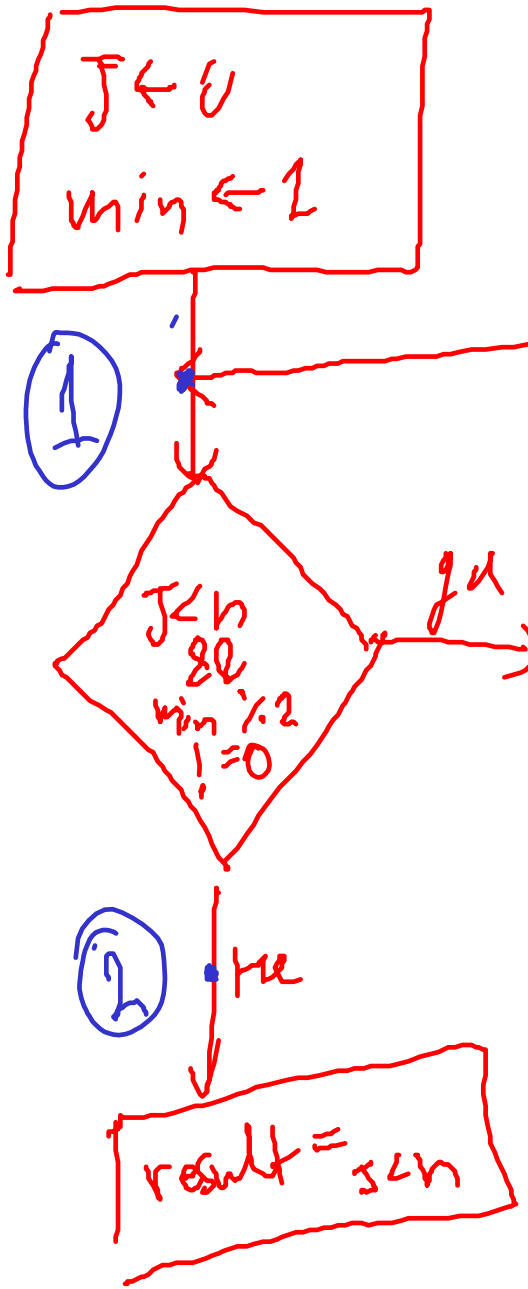
$\&$

$min_C(J_\ell - 1) \,\%\, 2 == 1$ ✓

$C(J_\ell, min_\ell)$

$J_\ell < n \;\&$

$min_\ell \,\%\, 2 == 1$



$I(J_k, min_k)$ е верно, т.е.

$(J_k == 0 \;\&\; min_k == 1 \;\vee\; min_k == minc(J_k - 1))$ | I_1

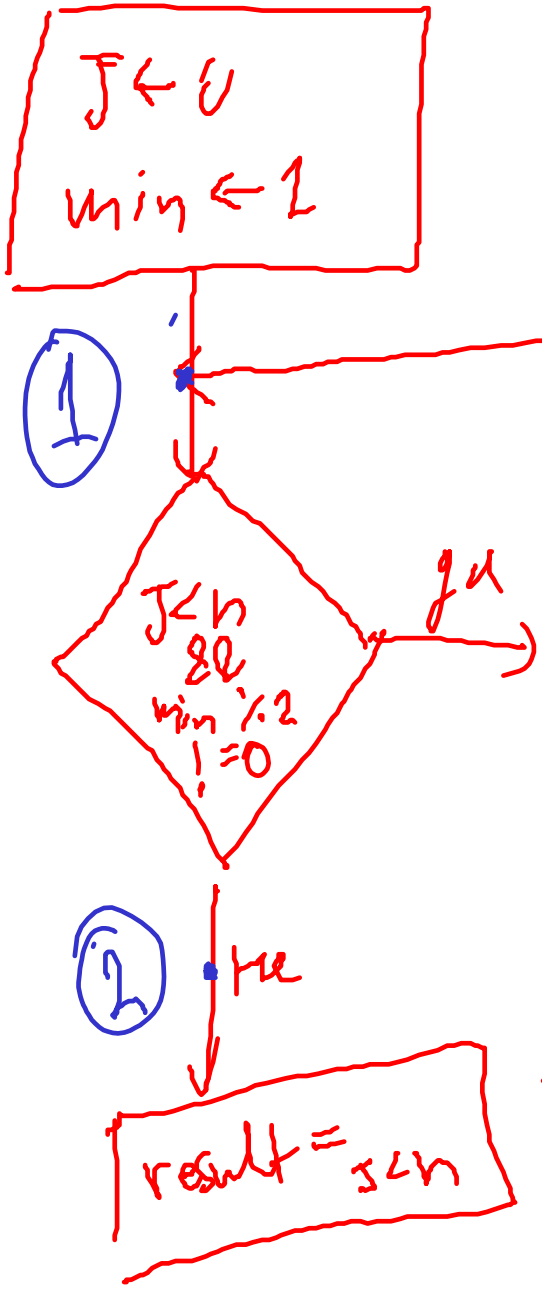
I_2 | $\& \; \forall 0 \le J' < J_k - 1 \;\; minc(J') \% 2 == 1$

$J_k \ge n \;\vee\; min_k \% 2 == 0$ | $\neg C$

Искане: $min_k \% 2 == 0 \Leftrightarrow \exists_J \; minc(J) \% 2 == 0$

→) Нека $min_k \% 2 == 0$ Тогава от I_2 $\checkmark$

$min_k == minc(J_k - 1)$.



$I(J_k, min_k)$ – цикл, т.е.

$(J_k == 0 \;\&\; min_k == 1 \;\vee\; min_k == minc(J_k - 1)) \;\&\; \forall 0 \le J' < J_k - 1 \; minc(J') \,\%\, 2 == 1$ | I

$J_k \ge n \;\vee\; min_k \,\%\, 2 == 0$ | ¬C

→) Цикл $\exists J \; minc(J) \,\%\, 2 == 0$

Условие: $min_k \,\%\, 2 == 0$

Дано е $\min c(J_k - 1) \,\%\, 2 == 1$. Тогава $J_k \geq h$ (от 1С)

$\Updownarrow$

$\min_k$

Тогава от I_2 $\Updownarrow$

с доказ, че $\exists j < h \;\; \min c(j) \,\%\, 2 == 0$

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

I над главния диагонал

$k=0$ $(0,0)$ $(1,1)$ $(2,2)$ ---

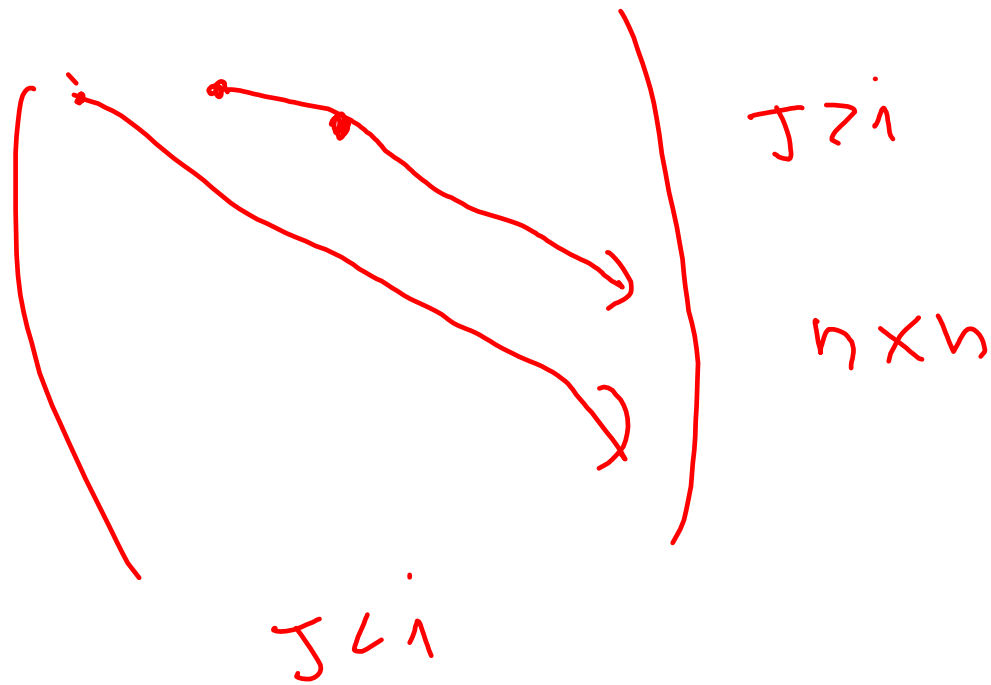
$k=1$ $(0,1)$ $(1,2)$ $(2,3)$ ---

$k=2$ $(0,2)$ $(1,3)$ $(2,4)$ ....

$k=n-1$ $(0,n-1)$

$k := j-i$

$i := j-k$



$k=0$ $(0,0)$ $(1,1)$ $(2,2)$ ---

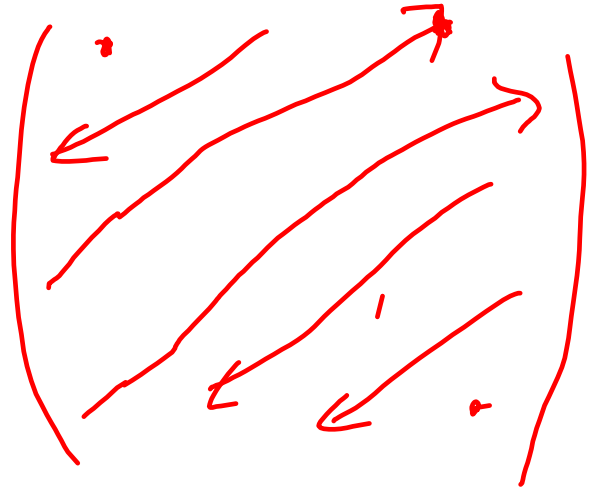
$k=1$ $(1,0)$ $(2,1)$, $(3,2)$, ---

$k=2$ $(2,0)$ $(3,1)$, $(4,2)$, ---

$k=n-1$ $(n-1,0)$

$k := i-j$

$i+j < n-1$



$i+j == n-1$

$i+j > n-1$

$(n-1, 0)$ $(n-2, 1)$ $\cdots$ $(0, n-1)$

$k = n-1$

$k := i+j$

$k = n-2$ $(n-2, 0)$ $(n-3, 1) \cdots (0, n-2)$

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 слеят “шахматно” две матрици от числа с еднакви размерности

Обхождане на матрици