ТЕХНОЛОГИЧНО УЧИЛИЩЕ “ЕЛЕКТРОННИ СИСТЕМИ”
КЪМ ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

МАТЕМАТИЧЕСКИ ОСНОВИ НА ПРОГРАМИРАНЕТО

ПРАКТИКА

МАТРИЦИ

Определение. Видове – вектор-стълб; вектор-ред; квадратна; единична; нулева; транспонирана, равенство, събиране

Дефиниране на масиви

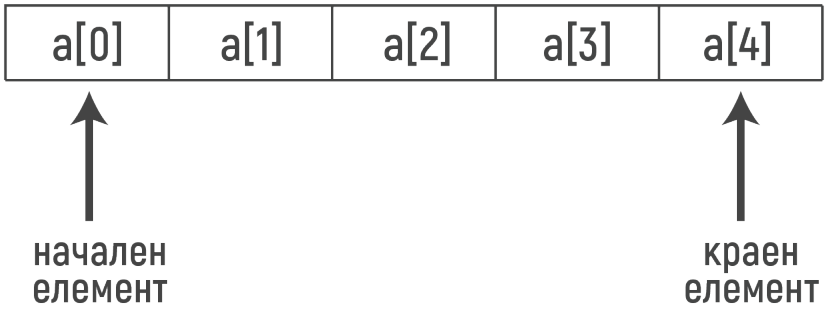
Масивът е структура от данни, състояща се от множество последователно наредени елементи от един и същи тип.

За даден тип T изразът a[size] е масив от size елемента от тип T.

```
1 int a[5];
2 char* b[40];
```

Дефиниране на масиви

Елементите на масива се индексират (номерират) от 0 до size – 1.



Дефиниране на масиви

Многомерните масиви се дефинират като масиви от масиви.

```
1 int d1[10][10];
2 int d2[10][20];
3 int d3[40][40][40];
```

Двумерни масиви

За съхранение и обработка на матрици (таблици) се използват двумерни масиви.

```
1 int m[3][4]; // двумерен масив с 3 реда и 4 колони
```

m[0][0]	m[0][1]	m[0][2]	m[0][3]
m[1][0]	m[1][1]	m[1][2]	m[1][3]
m[2][0]	m[2][1]	m[2][2]	m[2][3]

Двумерни масиви - инициализация

За инициализирането на двумерни масиви се използва списък от списъци за инициализация.

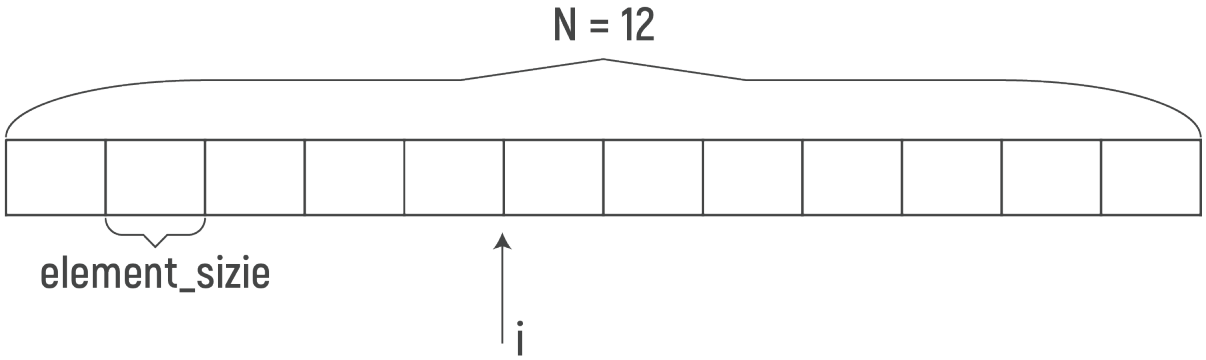
```
1 int m[][4] = { {1, 3, 5, 7}, {2, 4, 6, 8}, {1, 4, 5, 8} };
```

Двумерни масиви – предаване като аргумент на функция

Когато двумерен масив се предава като аргумент на функция, декларацията на масива задължително трябва да съдържа броя на колоните.

```
1 void f( int m[][4]) {
2   ...
3 }
```

Моделът на паметта в C++ за масиви

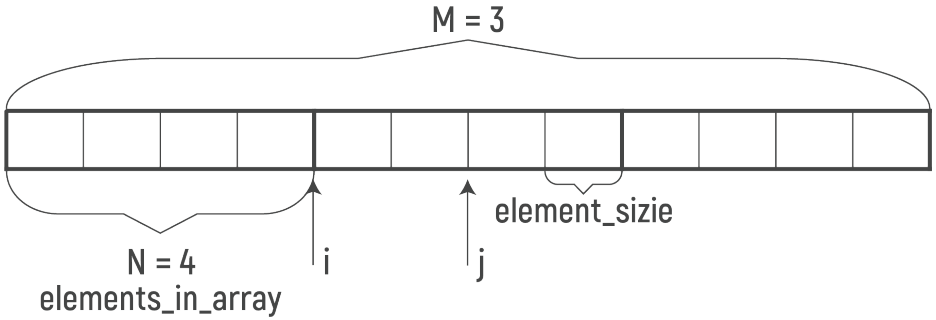


Моделът на паметта в C++ за масиви е непрекъснат в паметта блок от елементи.

Всеки блок е ***i*element_size*** от началото на масива, където:
i е индексът на елемента в масива;
element_sizie е размерът на един произволен елемент.

Двумерни масиви – предаване като аргумент на функция

Проблемът, който представят пред нас многомерните масиви е, че всеки техен елемент е друг масив, с фиксирана памет.



Размерът на тази памет е ***j*element_size*elements_in_array***, където:

j е индексът на едномерен масив, който се съдържа в двумерен масив;

element_sizie е размерът на един елемент;

elements_in_array е броят на елементите, които има в един от едномерните масиви, които които се съдържат в двумерния масив.

Двумерни масиви – предаване като аргумент на функция

elements_in_array е неизвестно и трябва да бъде указано някъде - затова се налага да се подава размера на масива по едното направление, когато се подава двумерен масив като аргумент на функция.

В C++ може да се създаде константа N, на която да се зададе фиксирана стойност, която да се използва за размер на колоните в двумерните масиви, които ще се подават като аргументи на функциите.

Това налага ограничение върху размера на колоните на всички матрици, които ще се използват, но това се налага единствено, за да може да работи аритметиката на указателите в двумерния масив. Може да се използва по-малко колони от зададената стойност на N.

Задача 0

Напишете функция, която да извежда матрица на стандартния изход. Функцията трябва да е със следната сигнатура:

```
1 void print(int A[][N], int m, int n)
```

A е матрици с максимален размер 50x50 (нека **N** е константа, **N=50**);
m и ***n*** са размера на матрица A;

Задача 0 структура на програмата

```cpp
#include <iostream>
using namespace std;

const int N = 50;

void print(int A[][N], int n, int m) {
  ...
}

int main() {
  int A[3][N] = {{12, 64, -34},
                 {-234, 12, 823},
                 {657, 283, -123}};

  print(A, 3, 3);

  return 0;
}
```

Задача 1

Напишете функция, която да проверява две матрици дали са равни. Функцията трябва да е със следната сигнатура:

```
1 bool equal( int A[][N], int m1, int n1, int B[][N], int m2, int n2 )
```

A и **B** са матрици с максимален размер 50x50 (нека **N** е константа, **N=50**);
m1 и ***n1*** са размера на матрица A;
m2 и ***n2*** са размера на матрица B.
Функцията да връща ***true***, ако двете матрици са равни и ***false***, ако не са.

Задача 2

Напишете функция, която да сумира две матрици и да записва сумата в трета. Функцията трябва да е със следната сигнатура:

```
1 void sum( int A[][N], int B[][N], int C[][N], int m, int n )
```

A, ***B*** и **C** са матрици с максимален размер 50x50 (нека **N** е константа, **N=50**);
m и ***n*** са размера на матриците.

Задача 2 примерен main

```
1 int main() {
2   int A[3][N] = {{12, 64, -34},
3                  {-234, 12, 823},
4                  {657, 283, -123}};
5   int B[3][N] = {{13, 46, 67},
6                  {81, 19, 34},
7                  {0, 3, -12}};
8   int C[3][N] = {{0}};
9
10  sum(A, B, C, 3, 3);
11  print(C, 3, 3);
12
13  return 0;
14 }
```

Задача 3

Напишете функция, която да транспонира матрица, резултата ѝ да се запише в друга. Функцията трябва да е със следната сигнатура:

```
1 void transpose( int A[][N], int B[][N], int m, int n )
```

A, B са матрици с максимален размер 50x50 (нека **N** е константа, **N=50**);
m и ***n*** са размера на матриците.

Задача 4

Напишете функция, която да умножава матрица с число, резултата от умножението да се запише в друга матрица. Функцията трябва да е със следната сигнатура:

```
1 void sMult(int A[][N], int R[][N], int m, int n, int s)
```

A, R са матрици с максимален размер 50x50 (нека **N** е константа, **N=50**);
m и ***n*** са размера на матриците;
s е числото с което се умножава.

Задача 5 домашна работа

Използвайки функциите ***sum*** и ***sMult***, напишете функция със следната сигнатура:

```
1 void sub( int A[][50], int B[][50], int C[][50], int m, int n )
```

Функцията записва в **C** резултатът от **A – B**.
A, B, C са матрици с максимален размер 50x50 (нека **N** е константа, **N=50**);
m и ***n*** са размера на матриците;