

# Программирование на языке C++

Занятие 6

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a, b;
    cin >> a;
    b = rand();
    if (a >= b)
        cout << "1";
    else cout << "2";
    system("pause");
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int n, p=1;
    cin >> n;
    for(int i=1; i<=n; i++){
        p=p*i;
    }
    cout << p;
    system("pause");
    return 0;
}
```

```
#include <iostream>
#include <cstdlib>
#include <cmath>
using namespace std;
int main()
{
    const int N=10;
    int a[N];
    for(int i=0; i<N; i++){
        a[i]= rand()%101;
    }
    for(int i=0; i<N; i++){
        cout << pow(a[i], 3) << endl;
    }
    system("pause");
}
```

```
#include <iostream>
using namespace std;
int main()
{
    const int n=20;
    int a[n], c;
    for(int i=0; i<n; i++){
        cin >> a[i];
    }
    for(int i=0; i<n-1; i++){
        for(int j=n-2; j>=i; j--){
            if(a[j]>a[j+1]){
                c=a[j+1]; a[j+1]=a[j]; a[j]=c;
            }
        }
    }
}
```

```
for(int i=0; i<n; i++){  
    cout << a[i] << " ";  
}  
  
    system("pause");  
    return 0;  
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int a[10], b[10], sum;
    for(int i=0; i<10; i++) {
cin >> a[i];
    }
    for(int i=0; i<10; i++) {
        sum=0;
        while(a[i]>0) {
            sum=sum+a[i]%10;
            a[i]=a[i]/10;
        }
        b[i]= sum;
    }
}
```

```
for(int i=0; i<10; i++) {  
    cout << b[i] << " ";  
}  
system("pause");  
    return 0;  
}
```



# План занятия

- Статические двумерные массивы  
(матрицы)
- Динамические одномерные массивы  
(vector)

# Что такое матрица?

	○	×
	○	×
○	×	

строка 1,  
столбец 2

**Матрица** — это прямоугольная таблица, составленная из элементов одного типа (чисел, строк и т.д.). Каждый элемент матрицы имеет два индекса — номера строки и столбца.

# Объявление матриц

```
const int N=3, M=4;  
int A[N][M];  
double X[N][12];
```

строки

столбцы

строки

столбцы



Нумерация строк и столбцов с нуля!

# Простые алгоритмы

## Заполнение случайными числами:

```
for ( i = 0; i < N; i++ ) {  
    for ( j = 0; j < M; j++ ) {  
        A[i][j] = 20 + rand() % 61;  
        cout << A[i][j] << " ";  
    }  
    cout << endl;  
}
```



Вложенный цикл!

## Суммирование:

```
sum = 0;  
for ( i = 0; i < N; i++ )  
    for ( j = 0; j < M; j++ )  
        sum += A[i][j];
```

# Задачи

**«А»:** Напишите программу, которая заполняет квадратную матрицу случайными числами в интервале  $[10,99]$ , и находит максимальный и минимальный элементы в матрице и их индексы.

**Пример:**

**Матрица А:**

12 14 67 45

32 87 45 63

69 45 14 11

40 12 35 15

**Максимальный элемент  $A[2,2]=87$**

**Минимальный элемент  $A[3,4]=11$**

# Задачи

«В»: Яркости пикселей рисунка закодированы числами от 0 до 255 в виде матрицы. Преобразовать рисунок в черно-белый по следующему алгоритму:

- 1) вычислить среднюю яркость пикселей по всему рисунку
- 2) все пиксели, яркость которых меньше средней, сделать черными (записать код 0), а остальные – белыми (код 255)

**Пример:**

**Матрица А:**

12	14	67	45
32	87	45	63
69	45	14	11
40	12	35	15

**Средняя яркость 37.88**

**Результат:**

0	0	255	255
0	255	255	255
255	255	0	0
255	0	0	0

# Чем плох обычный массив?

```
const int N = 100;  
int A[N];
```

статический  
массив

- память выделяется при трансляции
- нужно заранее знать размер
- изменить размер нельзя

*Задача.* В файле записаны фамилии (сколько – неизвестно!). Вывести их в другой файл в алфавитном порядке.

- выделить заранее большой блок (с запасом)
- выделять память во время работы программы (динамически!)

# Динамические структуры данных

... **ПОЗВОЛЯЮТ**

- создавать новые объекты в памяти
- изменять их размер
- удалять из памяти, когда не нужны

*Задача.* Ввести с клавиатуры целое значение  $N$ , затем –  $N$  целых чисел, и вывести на экран эти числа в порядке возрастания.



# Тип `vector` (библиотека STL)

STL = *Standard Template Library*



Вектор – это массив переменного размера!

Заголовочный файл:

```
#include <vector>
```

Объявление:

```
vector <int> A;
```

пустой массив  
типа `int`

Размер:

```
cout << A.size();
```

Заполнение (добавление в конец):

```
for ( i = 0; i < N; i++ )  
    A.push_back ( i + 1 );
```

# Тип `vector` (библиотека STL)

Обработка :

```
for ( i = 0; i < A.size(); i++ )  
    cout << A[i] << " " ;
```



Так же, как с обычным массивом!

# Расширение массива

*Задача.* С клавиатуры вводятся натуральные числа, ввод заканчивается числом **0**. Нужно вывести на экран эти числа в порядке возрастания.



Какой размер массива нужен?

**Ввод данных:**

```
cin >> x;  
while ( x != 0 )  
{  
    A.push_back(x);  
    cin >> x;  
}
```

автоматическое  
расширение

# Операции с векторами

## Объявление

```
vector <int> A;
```

```
vector <int> vector_first(3);
```

```
vector <int> vector_second;
```

```
vector_second.reserve(3);
```

# Методы `size()` и `empty()`

Если нам требуется узнать длину вектора, понадобится функция — `size()`. Эта функция практически всегда используется вместе с циклом [for](#).

Также, если нам требуется узнать пуст ли стек, мы можем использовать функцию — `empty()`.

При отсутствии в ячейках какого-либо значения это функция возвратит — `true`.

В противном случае результатом будет — `false`.

# Методы `push_back()` и `pop_back()`

- С помощью функции `push_back()` мы можем добавить ячейку в конец вектора.
- А функция `pop_back()` все делает наоборот — удаляет одну ячейку в конце вектора.

Использование функции `push_back()` без указания значения ячейки — невозможно. Так или иначе, придется это значение указать!