**Массивы**

Массив - это совокупность однотипных элементов, которые расположены в определённом порядке и обладают одним именем. Пример: int mas[10] - массив из 10 элементов.

**Элемент массива** - то переменная, имя которой совпадает с именем массива и у которой есть индекс. Чтобы вызвать элемент массива, нужно написать имя массива и указать в квадратных скобках индекс этого элемента. Пример: mas[0] - нулевой элемент массива.

**Индекс - э**то номер элемента в массиве.

**Важно.** Индексация элементов массива начинается с 0, т.е. у тебя никогда не будет в массиве из 5 элементов элемента под номером 5. Если ты возьмёшь номер элемента равный количеству элементов в массиве или больше него, то будет затронута память, которую не очищали от мусора для размещения массива.

**Важно.** В массиве могут быть элементы только того типа, которого и сам массив.

Пример 1: представьте себе улицу. На улице есть дома, расположенные в определённом порядке. У каждого дома есть свой номер. Здесь улица - массив, дома - элементы массива, номера домов - индексы элементов.

Пример 2: представьте себе класс из 20 человек. У классного руководителя есть журнал класса, где записаны все ученики и у каждого есть свой номер. Здесь класс из 20 человек - массив, ученики - элементы массива, а номера в журнале - индексы элементов.

Пример 3 (для тех, кто играет в «геншин» и подобные игры): есть персонаж, которому можно собрать 6 созвездий. Чтобы получить С6, нужно выбить этого персонажа 7 раз: один раз, чтобы получить самого персонажа (С0), и ещё 6 раз, чтобы получить на него 6 созвездий. То есть, у вас есть персонаж, которого вы выбиваете 7 раз, чтобы получить С6. Таким образом, это массив с[7], в котором 7 элементов: c[0], c[1], c[2], c[3], c[4], c[5], c[6].

**Инициализация массива**

Объявить массив можно разными способами, но есть определённые правила:

1. Как и у переменных, у массива должен быть указан тип перед началом работы с этим массивом.
2. Правила наименования массивов совпадают с правилами наименования переменных. **Следите, чтобы не было повторной инициализации**.
3. У массива должно быть указано количество элементов.

Примеры правильного объявления массивов:

* int ar1[3]; - массив из 3 элементов типа int. В дальнейшем мы можем записать нужные нам значения в массив.
* double ar2[3] = {1.2, 3, 7.1}; - массив из 3 элементов типа double. Все значения уже указаны, в случае необходимости мы можем в дальнейшем их изменить.
* char ar3[] = {'a', 'b', 'c'}; - массив из 3 элементов типа char. Все значения указаны.

**Важно.** Когда при инициализации массива мы прописываем в фигурных скобках значения элементов массива, мы можем не указывать размер массива. Компилятор посчитает элементы и поймёт, сколько байт нужно выделить на этот массив.

**Важно.** В качестве размера массива может стоять только число или константа. Если использовать обычные переменные, могут возникнуть ошибки.  
const a = 3;

int ar4[a]; - массив из 3 элементов типа int. Размер массива определён константой.

**Работа с одномерными массивами**.

Задача: создать целочисленный массив из чисел от 5 до 10.

Нам нужно объявить массив. Сколько элементов в этом массиве должно быть? Считаем: 5, 6, 7, 8, 9, 10 - в нашем массиве должно быть 6 элементов. Поскольку индексы элементов начинаются с 0, то номера элементов у нас будут от 0 до 5. Следовательно, нам нужно воспользоваться циклом от 0 до 5, чтобы заполнить массив.

Алгоритм:

1. Начало программы
2. Объявление массива arr[6]
3. Объявление переменной i = 0
4. Пока i <=5:
   1. arr[i] = i + 5
   2. i++
5. Конец программы

Пример реализации:

int main()

{

int arr[6];

for (int i = 0; i <= 5; i++)

{

arr[i] = i + 5;

}

}

Задача: заполнить массив из 6 элементов вручную.

Алгоритм:

1. Начало программы.
2. Объявление массива из 6 элементов.
3. Объявление переменной i = 0.
4. Пока i меньше 6:
   1. Ввод i-го элемента массива
5. Завершение программы

Пример реализации:

#include <iostream>

int main()

{

int arr[6];

for (int i = 0; i < 6; i++)

{

std::cin >> arr[i];

}

}

Задача: вывести на экран массив из 3 чисел. Числа заданы при инициализации массива.

Пример реализации:

#include <iostream>

int main()

{

int ar[3] = {0, 1, 2};

for (int I = 0; I < 3; I++)

{

std::cout << ar[I];

}

}

**Генератор псевдослучайных чисел**

**Виды событий**

Виды событий:

1. Невозможное. То, которое точно не произойдёт.
2. Достоверное. То, которое точно произойдёт.
3. Случайное. То, которое может произойти, а может и не произойти.

Пример. Вы бросаете 2 игральных кубика. На каждом из них есть 6 граней, на которых есть определенное количество точек от 1 до 6.

Варианты чисел, которые выпадут на кубике а: 1, 2, 3, 4, 5, 6. Минимальное возможное число - 1, максимальное возможное число - 6.

Варианты чисел, которые выпадут на кубике b: 1, 2, 3, 4, 5, 6. Минимальное возможное число - 1, максимальное возможное число - 6.

Чтобы узнать, сколько выпало на кубиках вместе, нам нужно сложить результаты броска обоих кубиков. Тогда, складывая минимально возможные числа, мы получаем нижнюю границу возможной суммы - 2. Аналогично для верхней границы мы получаем 12.

Следовательно, сумма очков на обоих кубиках будет всегда принадлежать промежутку от 2 до 12.

Событие, когда сумма выпавших очков на кубиках принадлежит этому промежутку, является *достоверным*.

Событие, когда сумма кубиков больше 12 или меньше 2, является *невозможным*.

Допустим, мы хотим, чтобы сумма выпавших очков на кубиках была равна 4. Это возможно, если:

* На одном из кубиков выпадет 1, на другом - 3
* На каждом из кубиков выпадет 2

В остальных случаях, нужный нам исход не наступит. Следовательно, это событие - *случайное.*

Ещё пример: ты фармишь артефакты. Вероятность того, что у тебя в цветке выпадет главным статом хп - 100%. Это достоверное событие. Вероятность того, что тебе выпадет там главным статом процент атаки - 0%. Это невозможное событие. А вот в подстатах атака может выпасть. А может и не выпасть. Это уже будет случайным событием.

В программировании абсолютной случайности не бывает. Её иллюзию создают сложные алгоритмы, результат выполнения которых сложно предугадать.

**Генератор псевдослучайных чисел**

Это программа, которая принимает некоторое число (стартовое значение), потом проводит с ним сложные математические операции, и так повторяется для каждого результата.

Пример простейшего ГПСЧ:

#include <iostream>

using namespace std;

int main()

{

for (int i = 0; i < 5; i++) //задаём количество итераций цикла. Наш цикл будет от 0 до 99 включительно, т.е., будет 100 итераций

{

static unsigned int r = 16729; //объявляем стартовое значение. Число может быть любым

r = (r \* 8253729 + 2396403); //формула для поиска псевдослучайного числа. Числа в формуле могут быть любыми

cout << r % 100 << endl; //нам нужны числа только от 0 до 99. При делении любого числа на 100, у нас всегда в остатке будет число в этом промежутке.

}

}

Функция **rand** генерирует случайное число (или последовательность чисел) на основе стартового значения. Для каждого стартового значения будет только один вариант последовательности.

Функция **srand** устанавливает стартовое значение для последовательности случайных чисел. В качестве аргумента (того, что в круглых скобках) можно использовать число, функцию, математическое выражение.

Функция **time** позволяет генерировать различные последовательности для одного стартового значения. Это функция возвращает нам, сколько прошло с 00:00:00 1 января 1970 года. Поскольку время не бывает одинаковым, то и результаты будут разными каждый раз. В нашем случае в качестве аргумента будет использоваться nullptr.

**Диапазон генерируемых чисел**

Диапазон задается с помощью операции вычисления остатка от деления. Возьмём игральный кубик. У нас 6 граней, пронумерованных от 1 до 6: 1, 2, 3, 4, 5, 6. Если мы напишем rand() % 6, как делали это в первом примере, то у нас будут выводиться числа от 0 до 5 включительно.

Для задания диапазона используется специальная формула:

верхняя граница - нижняя граница + 1.

Для нашего кубика: 6 - 1 + 1 = 6. Следовательно, диапазон значений - 6. Но нам нужно задать ещё и нижнюю границу, потому что нам нужны числа только от 1 до 6. Тогда мы получаем следующую запись:

rand() % 6 + 1