**Arrays / Массивы**

**Arrays is defined as a collection of similar data elements. If you have some sets of integers or some sets of floats, you can group them under one name as an array.**

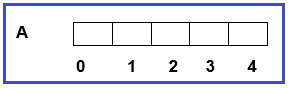
**Массивы** определяются как набор похожих элементов данных. Если у вас есть наборы целых чисел или наборы чисел с плавающей запятой, вы можете сгруппировать их под одним именем в виде массива.

**Способ объявления массива**

Если вам нужен массив целочисленного типа,

* В С++ запишим, int A[5];
* в Golang, **var** A [5] **int**

Таким образом мы получим 5 пять целочисленных элементов с индексом от 0 до 4 и имя массива «A».

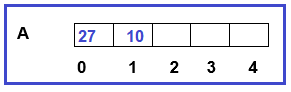


Массив представляется цельным куском памяти, размер которого вычисляется по следующей формуле: количество элементов \* количество памяти под каждый элемент

Размер массива - фиксированная величина. Те динамические массивы, с которыми мы имеем дело во многих языках, реализованы уже внутри языка, а не на уровне железа.

Все элементы массива имеют один тип и занимают одно и то же количество памяти. Благодаря этому появляется возможность простым умножением (по формуле, описанной выше) получить адрес той ячейки, в которой лежит нужный нам элемент. Именно это происходит под капотом, при обращении к элементу массива под определённым индексом.

Если предположить, что целое число занимает 4 байта, то массив А будит занимать в памяти 20 байтов.

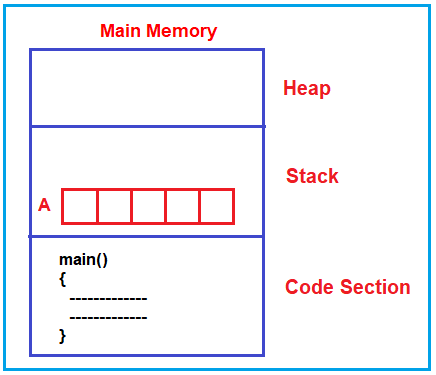


Чтобы записать значение в массив «A» с индексами 0, можно использовать следующую запись A[0]=27; так мы получаем доступ к каждому местоположению.

**Как объявить и инициализировать массив?**

|  |  |
| --- | --- |
| C/C++ | Golang |
|  |  |

Когда программа работает, она выполняется в основной памяти. Основная память разделена на 3 раздела: раздел кода (Code section), раздел стека (Stack section) и раздел кучи (Heap section), как показано на рисунке ниже.



Как вы можете видеть на изображении выше, любой код, который мы напишем, будет внутри секции кода. Вам нужно понять, что переменные, которые мы объявили, будут созданы внутри секции Stack. Итак, здесь массив также будет создан внутри стека, поскольку массив также является переменной.

Те переменные, которые создаются внутри секции стека, напрямую доступны методу Main из секции кода. Таким образом, массив напрямую доступен основной функции и может напрямую хранить значения в массиве.

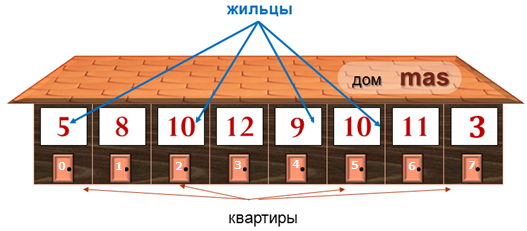
**Как получить доступ к массиву?**

|  |  |
| --- | --- |
| C/C++ | Golang |
|  |  |



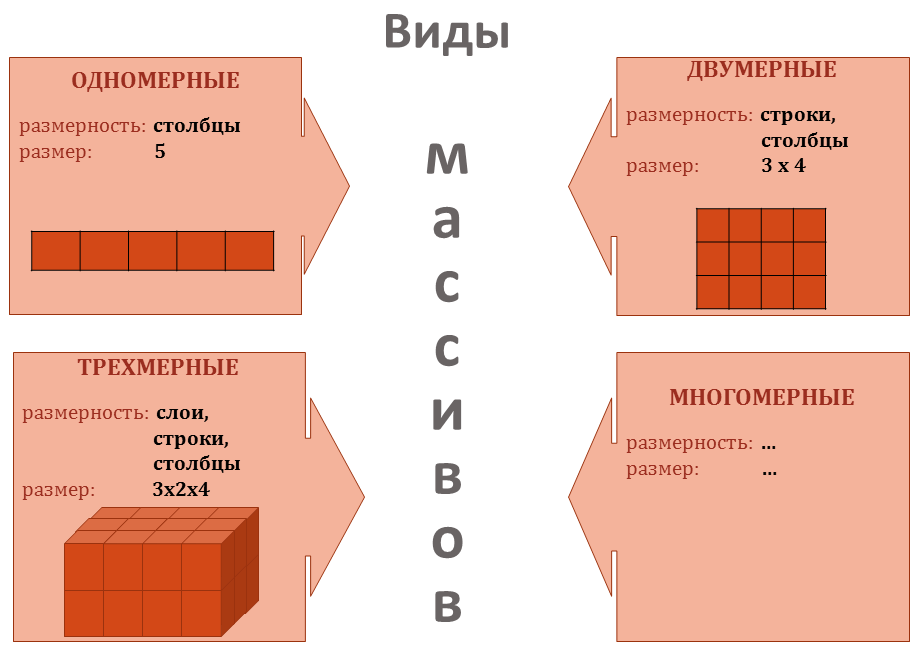
**МАССИВ можно сравнить и с домом:**

* название массива - имя дома (адрес)
* индексы элементов массива  - номера квартир
* элементы массива - жильцы дома.



**ВИДЫ МАССИВОВ**

  Все массивы можно разделить на две большие категории: **ОДНОМЕРНЫЕ** и **МНОГОМЕРНЫЕ**.

****

* ОДНОМЕРНЫЕ      - одна мера - СТОЛБЦЫ
* ДВУМЕРНЫЕ         - две   меры - СТРОКИ, СТОЛБЦЫ
* ТРЕХМЕРНЫЕ       - три   меры - СЛОИ, СТРОКИ, СТОЛБЦЫ
* МНОГОМЕРНЫЕ   - много мер  - ...

**ОДНОМЕРНЫЙ** массив можно сравнить с ОДНОЭТАЖНЫМ домом, в котором мера - квартиры

**ДВУМЕРНЫЙ** массив можно сравнить с МНОГОЭТАЖНЫМ домом, в котором мера - этажи, на этаже - квартиры

**ТРЕХМЕРНЫЙ** массив можно сравнить с КОРПУСАМИ МНОГОЭТАЖНОГО дома, в котором мера - корпуса, в корпусе - этажи, на этаже - квартиры.

<https://sites.google.com/site/programmirovaniepaskal/razdel-6-massivy>

<https://ru.hexlet.io/courses/php-arrays/lessons/implementation/theory_unit>