Материал4 - std::vector

**std::vector**

vector - стандартный шаблон классов из стандартной библиотеки C++. Неформально говоря, шаблон - это набор похожих типов, параметризованный другим типом или типами. Так, сам по себе std::vector не есть тип, а всего лишь шаблон. Типами(причем, разными) являются std::vector<int>, std::vector<char>, и т.д. (шаблонный аргумент указывается в угловых скобках <>).

Векторы имитируют [массивы](https://www.google.com/url?q=https://docs.google.com/document/pub?id%3D1n9PfTHan6G5d11K7EggmLNP1DLPs-2iwn1plbYXjnPA&sa=D&ust=1529881960053000). Как устроены векторы внутри - неважно. Важно, что их поведение очень похоже на поведение массивов, но функциональность вектора намного шире и обобщеннее, чем у массива.

Схожие характеристики массивов и векторов включают в себя:

* элементы вектора(как и массива) размера N имеют индексы 0, 1, 2, …, N-1
* Поведение программы, которая обращается к несуществующему элементу вектора (как и массива) неопределено.
* Гарантируется, что элементы вектора(как и массива) хранятся в памяти друг за другом, без “пустых” промежутков

Различия массивов и векторов включают в себя:

* Размер вектора может быть любым неотрицательным числом (в случае массива должно быть строго положительное константное выражение)
* Вектор нельзя инициализировать фигурными скобками (в C++03). В C++11 это возможно (но не все компиляторы пока поддерживают этот синтаксис)
* Размер вектора можно изменить во время выполнения.
* Вектору можно присвоить другой вектор с помощью =
* Векторы можно сравнивать на равенство с помощью ==

**Инициализация(конструирование) вектора.**

Вектор можно создать многими способами. Среди них

**- По умолчанию**

**std::vector<double> v;**

Размер вектора - 0. Элементов нет. Вектор пустой. Осторожно! Если написать

**std::vector<double> v();**

то это будет не создание вектора, а объявление функции по имени v, корорая ничего не принимает и возвращает std::vector<double>.

**- Задав начальный размер**

**std::vector<int> v(14);**

Размер вектора - 14. В векторе 14 элементов, каждый из них равен нулю.

**- Задав начальный размер и начальное значение**

**std::vector<char> v(7, ‘e’);**

Размер вектора - 7. В векторе 7 элементов, каждый из них равен ‘e’.

**- Скопировать из другого вектора.**

**std::vector<int> v;**

**v.push\_back(1);**

**v.push\_back(2);**

**v.push\_back(3);**

**std::vector<int> v1(v); //или std::vector<int> v1 = v;**

Вектор v1 есть копия вектора v. То есть его размер равен 3, а элементы - 1, 2, 3. Заметьте, что если теперь изменить v, то на v1 это не повлияет и наоборот. v1 - копия лишь в момент создания.

**Полезные функции-члены шаблона vector:**

 - **size()** - возвращает размер вектора  в виде беззнакового целого

**std::vector<bool> v(10);**

**v.push\_back(true);**

**const int n = v.size(); // n == 11**

- **push\_back(element)** - добавляет элемент в конец вектора. Размер вектора увеличивается на единицу.

**std::vector<char> v(2, ‘a’); //v.size() == 2; {‘a’, ‘a’}**

**v.push\_back(‘b’);  //v.size() == 3; {‘a’, ‘a’ ,’b’}**

**v.push\_back(c); //v.size() == 4; {‘a’, ‘a’ ,’b’, ‘c’}**

- **pop\_back()** - убирает элемент с конца вектора. Уменьшает размер вектора на единицу. Если вектор был пуст, поведение неопределено.

**std::vector<char> v(2, ‘a’); //v.size() == 2; {‘a’, ‘a’}**

**v.pop\_back(); //v.size() == 1; {‘a’}**

**v.pop\_back(); //v.size() == 0; {}**

**v.pop\_back(); //НЕОПРЕДЕЛЕННОЕ ПОВЕДЕНИЕ!!!**

- **resize(newSize) -**изменяет размер вектора. Новый размер становится равен **newSize.** При этом если новый размер меньше предыдущего, то лишние элементы с конца отбрасываются, если же больше, то новые элементы инизиализируются нулем (false в случае bool). Можно добавленные элементы инизиализировать и другим значением, если его явно указать в **resize:** **resize(newSize, value);**

**std::vector<double> v(4, 2.0); //v.size() == 4; {2.0, 2.0, 2.0, 2.0}**

**v[1] = 3.0; //v.size == 4; {2.0, 3.0, 2.0, 2.0}**

**v.resize(2); //v.size() == 2; {2.0, 3.0}**

**v.resize(4); //v.size() == 4; {2.0, 3.0, 0.0, 0.0}**

**v.resize(6, 2.5); //v.size() == 6; {2.0, 3.0, 0.0, 0.0, 2.5, 2.5}**

**- empty() -**проверяет, пуст ли вектор. Возвращает true если пуст, false в противном случае. Эквивалентен **v.size() == 0.**Не путайте с функцией**clear.**

**std::vector<int> v;**

**if(v.empty()) std::cout << “Empty”; //печатает Empty**

**v.push\_back(1);**

**if(v.empty()) std::cout << “Empty”; //ничего не печатает**

**v.resize(0);**

**if(v.empty()) std::cout << “Empty”; //печатает Empty**

**- clear() -**удаляет все элементы из вектора. Размер становится нулем. Эквивалентен **v.resize(0).**Не путайте с функцией**empty.**

**v.clear();**

**const bool b = v.empty(); // b гарантированно равен true**

[Google Диск](https://docs.google.com/)–[Сообщение о нарушении](https://docs.google.com/abuse?id=e/2PACX-1vQy1ZcJ9zW9cqHOB67HF2pitvspFQleZdXQgzdx5_JPHM006dua579f9u_VJzBDWsxPgw3i9akxOagp)–Интервал автоматического обновления в минутах: 5