Содержание

[ОБЩАЯ ИНФОРМАЦИЯ 2](#_heading=h.gjdgxs)

[Тема занятия](#_heading=h.30j0zll) 2

[Цели и задачи 2](#_heading=h.1fob9te)

[Ожидаемый результат 2](#_heading=h.3znysh7)

[Структура занятия 3](#_heading=h.tyjcwt)

[ОПИСАНИЕ ЗАНЯТИЯ 4](#_heading=h.3dy6vkm)

# ОБЩАЯ ИНФОРМАЦИЯ

### Тема занятия

Standard Template Library.

Структуры данных.

### Цели и задачи

* Познакомиться со структурами данных (динамический массив и список)
* Научиться работать с объектами класса vector и list
* Познакомиться с понятием вычислительной сложности
* Узнать о возможных обобщений уже устоявшихся понятий в С++ (итераторов)

### Ожидаемый результат

По результатам занятия Слушатель должен знать:

* Что собой представляет стандартная библиотека шаблонов
* Почему существуют несколько структур данных
* В каких случаях лучше использовать определённую СД
* Что такое итератор

### Структура занятия

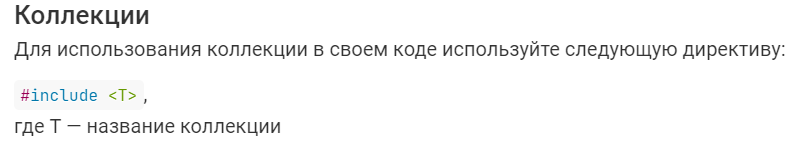
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Тайминг занятия** | |  |  |  | Таблица 1 |
| № | Этапы | Что делает преподаватель | Что делает Слушатель | Время | Общее время |
| 1 | Приветственное слово преподавателя |  |  | 5 мин. | 5 мин. |
| 2 | Повторение пройденного |  |  | 5 мин. | 40 мин. |
| 3 | Теоретическая часть |  |  | 30 мин. |
| 4 | Вопросы по теоретической части |  |  | 5 мин. |
|  | *Перерыв* |  |  | 15 мин. | 15 мин. |
| 5 | Практическая часть |  |  | 40 мин. | 45 мин. |
| 6 | Рефлексия и вопросы |  |  | 5 мин. |

# ОПИСАНИЕ ЗАНЯТИЯ

**Теоретическая часть**

Механизм шаблонов встроен в компилятор C++, чтобы дать возможность программистам делать свой код короче за счет обобщенного программирования. Естественно, существуют и стандартные библиотеки, реализующие этот механизм. STL является самой эффективной библиотекой C++ на сегодняшний день.

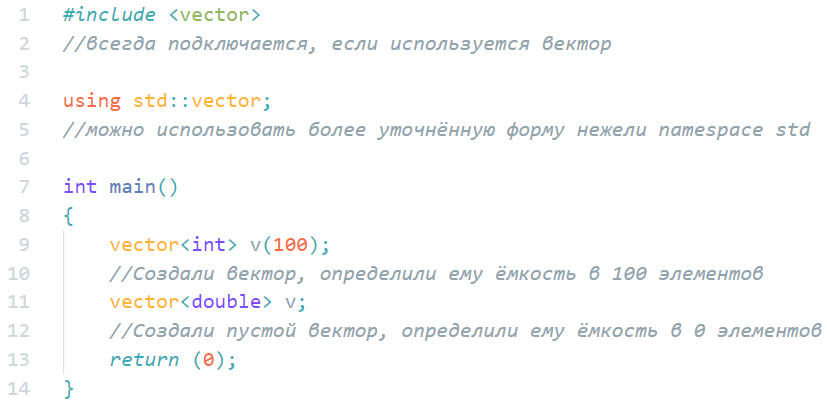
Для начала рассмотрим самые популярные коллекции из библиотеки. Каждая из них имеет собственный набор шаблонных параметров, чтобы быть максимально удобной для как можно большего спектра решаемых задач.



**Vector** — коллекция элементов, сохраненных в массиве, изменяющегося по мере необходимости размера (обычно, увеличивающегося);

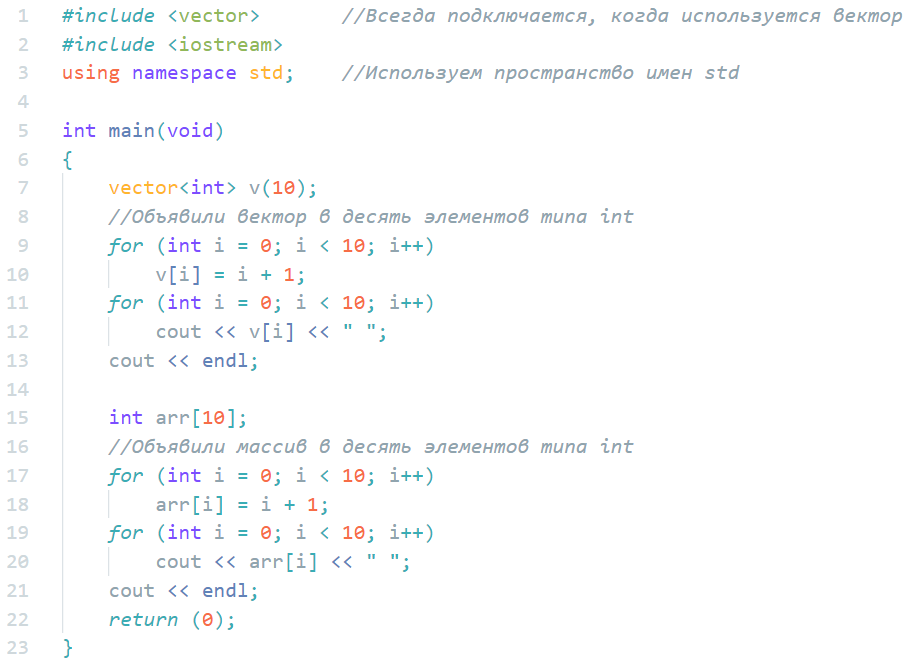
**Динамический массив** — это такой массив, который может и расширять и сужать свою фактическую вместимость.

* + Векторы являются частью STL и относятся к последовательным контейнерам.
  + Последовательные контейнеры — это упорядоченные коллекции, в которых каждый элемент занимает определенную позицию.
* Поскольку вектор прежде всего массив (массив с расширенной функциональностью), то с объектами-векторами можно работать как с обычным массивом. Вектор всегда может вычислить свою действительную вместимость с помощью функции **size()**. В разные моменты времени вместимость может изменяться (о чём говорит определение вектора как динамического массива) и всегда можно узнать последнюю актуальную.

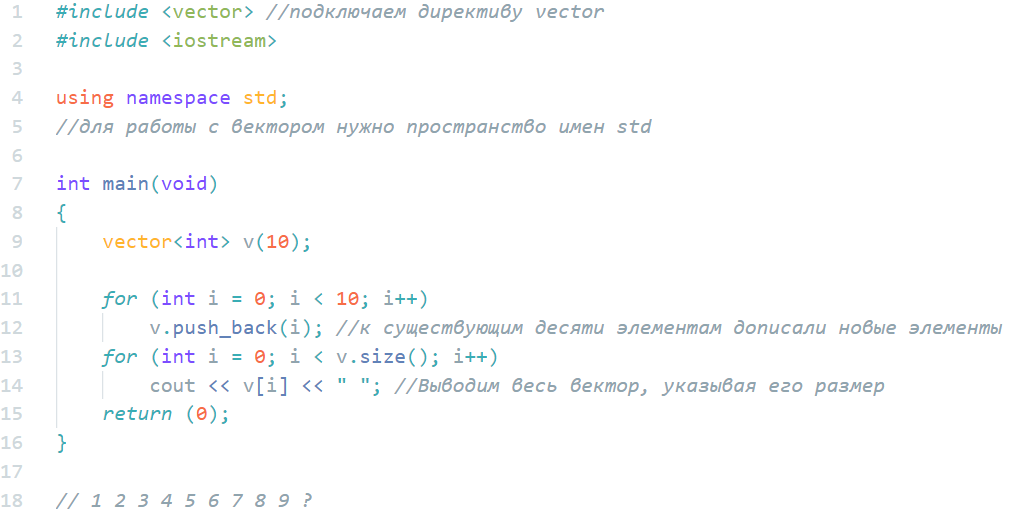


**Вектор в C++ — это замена стандартному** [**динамическому массиву**](https://code-live.ru/post/cpp-dynamic-arrays/)**, память для которого выделяется вручную, с помощью оператора new.**

**Разработчики языка рекомендуют в использовать именно vector вместо ручного выделения памяти для массива. Это позволяет избежать утечек памяти и облегчает работу программисту.**



В отличие от обычных массивов векторы обладают свойством дополнительной функциональности. Поскольку векторы — это объекты класса, а в том классе описаны специальные функции для объектов, то мы можем использовать те описанные функции для всех объектов-векторов.



В отличие от обычных массивов, где мы или используем прямое указание номера позиции или двигаем указатель, по элементам векторов можно ходить с помощью так называемых итераторов.

* Итератор — это специальный элемент, подобный указателю, предназначенный для навигации по элементам коллекции.

Итераторы — это не указатели как таковые, а объекты обобщённых классов, выполняющие роль указательных переменных, подходящих обобщённому классу.

Итераторы дают нам доступ к элементам коллекций (коллекции — это структуры данных, умеющие хранить множество элементов, кроме массива существуют и другие структуры).

Любой контейнер стандартной библиотеки шаблонов (а вектор — это контейнер такой библиотеки) всегда содержит методы **begin()** и **end()**, по которым всегда можно отследить начало расположения объекта-вектора в памяти и конец. Эти методы возвращают соответствующие указательные переменные на адреса. И эти адреса можно использовать, используя итераторы в качестве бегунков.

Удаляемые элементы оставляют после себя использованную собой память в резерве объекта. Связано это с тем, что очистка памяти наряду с выделением — операции дорогостоящие, т. е. времязатратные, поэтому в угоду эффективности остаётся резерв ячейки памяти за объектом. Так не придётся перевыделять и переуничтожать многократно, что благоприятно сказывается на скорости работы программы. Но при желании мы своими силами можем насильно подчистить память, оказавшуюся в резерве объекта-вектора.

Из-за динамичности вектора следует различать понятия ёмкости и размера вектора. Ёмкость вектора предполагает собой общую вместимость элементов, а размер вектора представляет собой количество элементов, расположившихся внутри вектора.

size() — функция объект-вектора, позволяющая узнать количество элементов, расположившихся внутри объекта-вектора.

capacity() отображает общее число ячеек, которые используются и забронированы вектором.

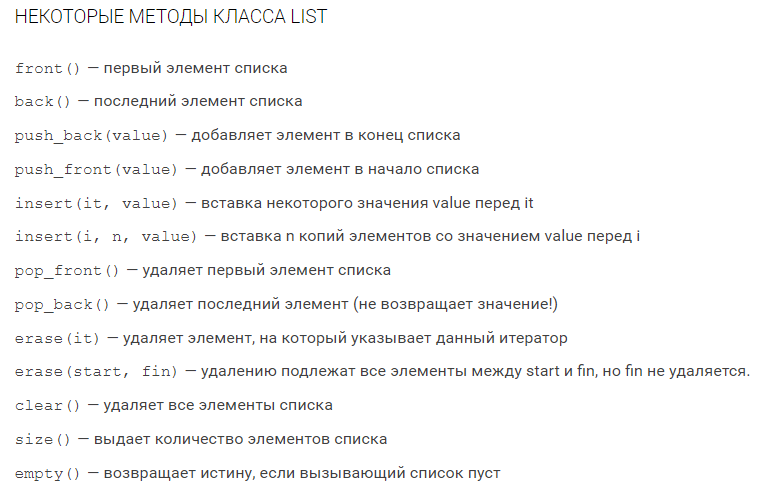
Для того, чтобы расширить объект-вектор вручную, нужно использовать метод resize(). Метод resize() легко расширит фактическую вместимость вектора, но не сможет убрать излишки резерва.

Чтобы убрать излишки резерва, нужно использовать метод shrink\_to\_fit()

Контейнер **list** реализует структуру данных "двусвязный список" с возможностью добавления (и удаления) элементов и в начало, и в конец, и в середину списка за время О(1)

Минусом двусвязного списка является отсутствие возможности произвольного доступа, которое есть у массива и вектора.

Двусвязный список являет собой сущность, где каждый отдельный внутренний элемент имеет прямую связь со своими соседствующими элементами, при этом любые не соседствующие элементы друг с другом никаких связей не имеют и ничего друг о друге не знают.



Итератор list не является итератором прямого доступа. Поэтому для него неприменимы операции +,  –, +=, -=, а для контейнера недопустима операция индексации. Итераторы для этого контейнера перемещаются последовательно операциями  ++ и — .

**Практическая часть**

**Задание 1**

Задача о порождении всех подмножеств множества из n элементов.

**Задание 2**

Задача о перестановках элементов множества из n элементов.

(Файл с кодом лежит в папке code/)