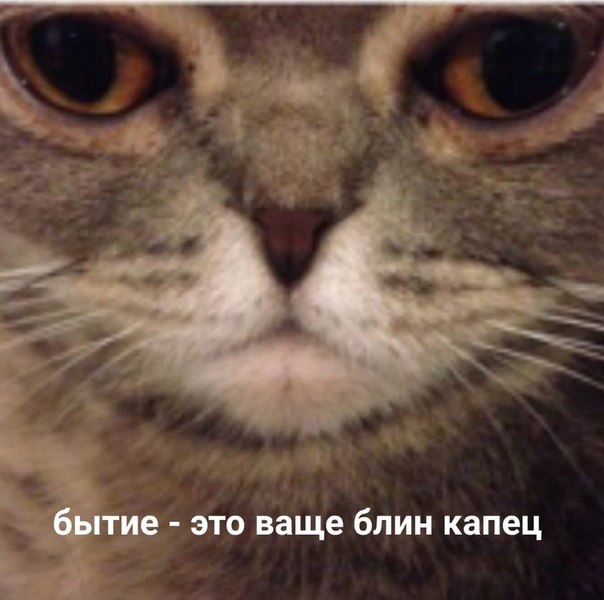
В современных проектах, написанных на с++, очень часто появляются такие понятия как “контейнер”, “STL”, “Алгоритмическая сложность”, “Итератор” и многие другие страшные для новичков слова. Попробуем описать эти понятия на примерах:  
  
Начнем с того, **что такое STL?**

**S**tandard **T**emplate **L**ibrary - это набор согласованных обобщённых алгоритмов, контейнеров, средств доступа к их содержимому и различных вспомогательных функций в C++.



Если сравнивать это с реальной жизнью можно привести в пример огромный гараж какого-то очень богатого человека, у которого есть велосипеды, машины, вертолеты, грузовики, лыжи и вообще все виды транспорта :)



Когда человеку нужно доставить **много груза и не важно сколько времени** это займет, он берет **грузовик** (если умеет им управлять)

Когда человеку нужно отправить **небольшую посылку в соседний дом** он может доехать на **машине** или **велосипеде**

Когда человеку нужно **доставить срочно, много, далеко и не важно сколько это отнимет ресурсов** он берет **самолет**

Библиотека STL это огромный гараж, только транспортом здесь служат контейнеры, которые были написаны для максимальной эффективности в конкретных задачах.

И тут встает важный вопрос: Разумно ли посылать самолет с маленькой посылкой в соседний дом? Или ехать на велосипеде, волоча за собой 100 кг груза, в другую страну?



От программиста зависит правильность применения контейнеров в проектах. Конечно в маленьких проектах будет незаметное падение скорости или засорение памяти, но как только проект станет больше уже будет поздно что-то менять.

Итак, наиболее часто используются следующие контейнеры:

* **vector — коллекция элементов, сохраненных в массиве, изменяющегося по мере необходимости размера (обычно, увеличивающегося).** Все любят векторы, все их постоянно используют в проектах, и это действительно удобный контейнер. Им можно заменить обычный или динамический массив, и он даже лучше в плане добавления новых элементов в любое место. Но как дело доходит до скорости на форумах начинаются скандалы, интриги, расследования.

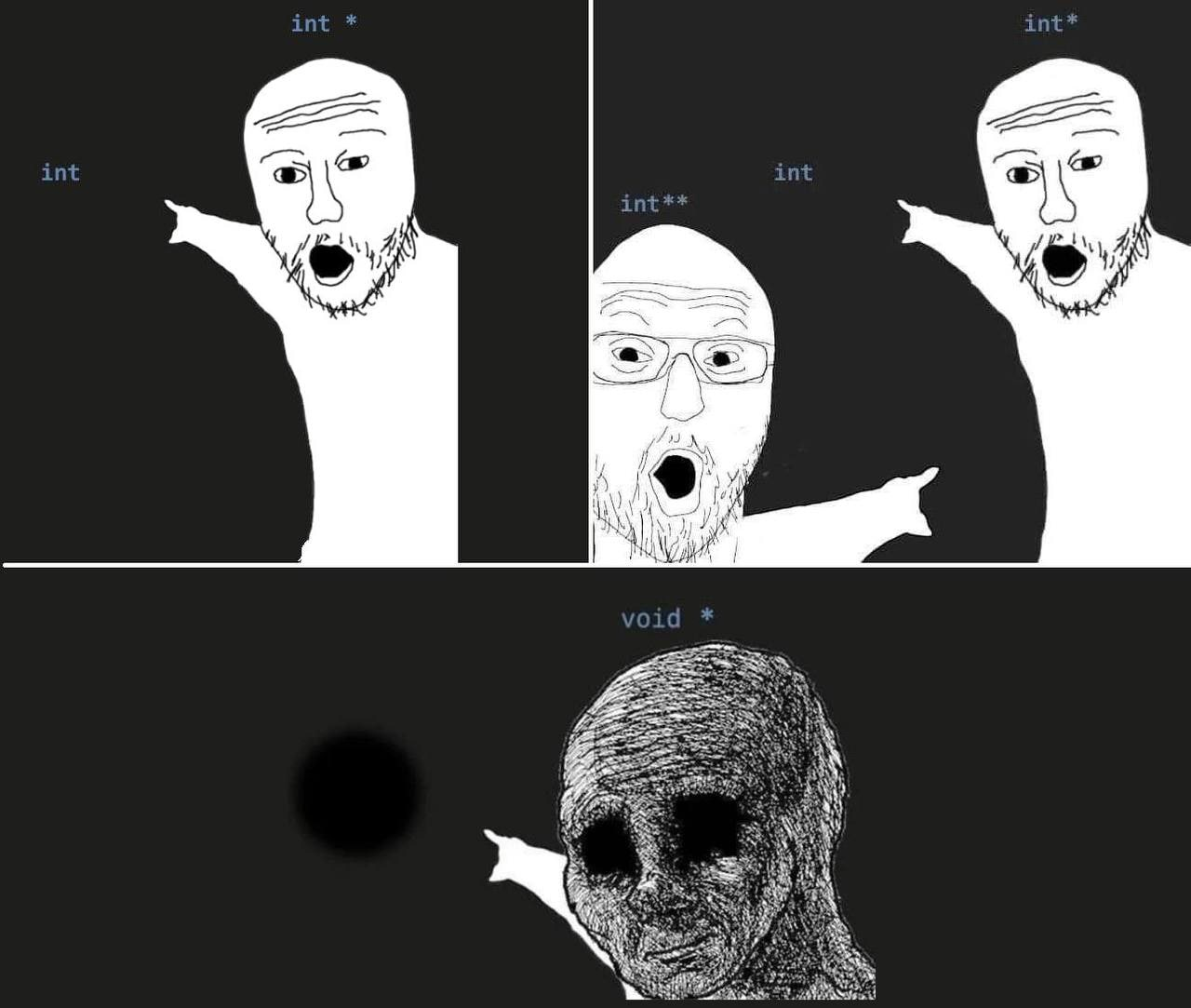


О всех плюсах и минусах можно написать очень много “интересного текста”, но я напишу одно:

Используйте вектор когда не знаете сколько элементов в нём будет храниться. С этой задачей он справляется идеально и не нужно придумывать “велосипед” с постоянным выделением памяти под новые динамические массивы, ведь вектор в своей реализации уже все сделал за вас.

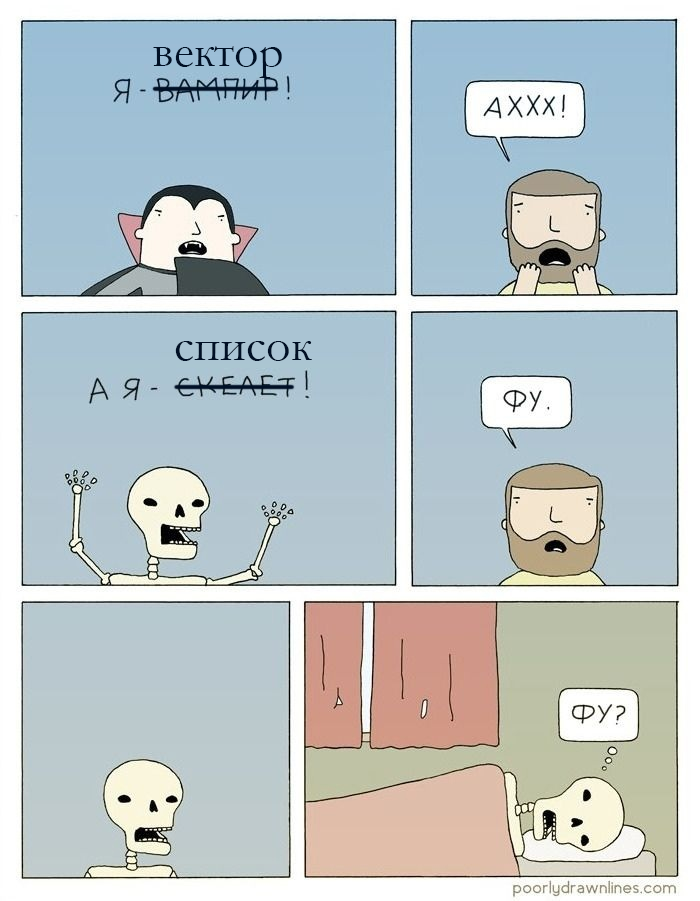
Если элементов действительно очень много, то вам как программисту достаточно указать приблизительный начальный размер вектора, чтобы не так сильно терять в скорости по сравнению с массивами

* **list — коллекция, хранящая элементы в виде двунаправленного связанного списка.** Не самый частый гость многих проектов, сильно недооценен программистами, которые просто не могут найти списку применение. Самое частое обвинение: Долгий поиск элемента и отсутствие индексов. Но почему-то списки мало кто хвалит за максимальную скорость удаления и добавления элемента в любое место.



Все элементы в списке хранятся в виде указателей друг на друга. первый указывает на второго, второй на первого и третьего, третий на второго и четвертого и так далее. При удалении третьего мы меняем только 2 указателя и это происходит настолько быстро в отличия от вектора, который пересобирает целый, возможно огромный блок в памяти когда ты удалил например только второй элемент.

Используйте list в С++ когда вам нужно часто удалять или добавлять элементы, и не нужен поиск, а тем более индексы



* **map — коллекция, сохраняющая пары вида <ключ, значение>**, где ключ — некоторая характеризующая значение величина, для которой применима операция сравнения. Пары хранятся в отсортированном виде, что позволяет осуществлять быстрый поиск по ключу, но за это, естественно, придется заплатить: придется так реализовывать вставку, чтобы условие отсортированности не нарушилось.

Достаточно уникальный контейнер, у которого есть масса применений. Вы знаете что у вас есть уникальный объект, но не уникальное значение, которое ему соответствует. Например: посчитать количество каждой буквы в тексте.

Порой бывает что map вообще не следует использовать. Например если существует объект какого то класса и он может сам хранить поле, которое вы могли записать в качестве значения. А контейнер таких объектов можно заменить на наш следующий контейнер

* **set — это отсортированная коллекция одних только ключей**, т.е. значений, для которых применима операция сравнения, при этом уникальных — каждый ключ может встретиться во множестве (от англ. set — множество) только один раз;

Идеальный вариант для того чтобы использовать только уникальные объекты в проекте

* **multimap — map, в котором отсутствует условие уникальности ключа**, т.е. если вы произведете поиск по ключу, то получите не единственное значение, а набор элементов с одинаковым значением ключа
* **multiset** — **коллекция** с тем же отличием от **set**, что и multimap от map, т.е. **с отсутствием условия уникальности ключа**

****

**Подробнее об использовании можете почитать** [**здесь**](https://codelessons.ru/category/cplusplus/kontejnery) **и** [**здесь**](https://metanit.com/cpp/tutorial/7.1.php)