# Стек

**Принцип работы**

Стек (Stack) — структура данных, представляющая собой набор данных, доступ к которым реализуется по принципу «последний пришел — первый ушел» (LIFO).

До первого элемента в стеке можно добраться только после того, как будут последовательно сняты позже добавленные элементы.

Вершина стека — последний добавленный в стек элемент! Примеры из реальной жизни: стопка книг/тарелок/etc.

**Интерфейс структуры**

Интерфейс структуры (возможные операции со структурой):

1) push — положить элемент на вершину стека;

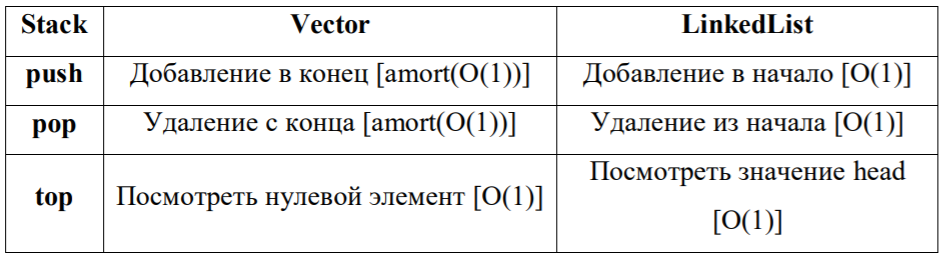
2) pop — снять элемент с вершины;

3) top — посмотреть элемент на вершине.

Также могут реализовываться дополнительные методы: size, isEmpty.



**Сложность**



(size и isEmpty O(1))

**Реализация**

Суть паттерна «Bridge» — разделение классов на две отдельные иерархии — иерархию абстракции и иерархию реализации. Благодаря такому разделению, можно изменять иерархии независимо друг от друга. Шаблон используется, когда нужно иметь несколько версий одного класса в программе. В нашем случае абстракцией будет являться класс стека, а реализациями — конкретные реализации на базовых структурах. Это позволит нам в будущем удобно добавлять новые способы реализации стека без изменения класса самого стека.

Клиентский код будет видеть только абстракцию (стек), реализация же будет развиваться в иерархии реализации. Реализовывается данный паттерн с помощью композиции объекта иерархии реализации в классе абстракции.

Идиома Pimpl (Pointer to Implementation) реализует композицию так, чтобы реализация была делегирована объекту-части, причем таким образом, чтобы в хедере «целого» не было включений файлов с конкретными реализациями: это позволяет увеличить скорость компиляции, а также скрыть от клиента особенности реализации.

Отрицательное качество данной идиомы — ухудшение производительности за счет обращения к объекту на куче. Иерархия абстракции будет представлена всего одним классом — стеком (Stack). Иерархия реализации будет содержать интерфейс StackImplementation, содержащий чистые виртуальные функции стека, которые будут реализовывать классы StackVector и StackList. Stack в приватной секции будет содержать указатель на StackImplementation, а реализация стека будет просто дергать методы StackImplementation.

При таком подходе, для добавления новой реализации нужно будет всего лишь добавить новое условие в конструктор класса Stack и для нового контейнера реализовать интерфейс StackImplementation.

# Очередь

**Принцип работы**

Очередь (Queue) — структура данных, представляющая собой набор данных, доступ к которым реализуется по принципу «первый пришел — первый ушел» (FIFO).

До последнего элемента в очереди можно добраться только после того, как будут последовательно сняты ранее добавленные элементы.

Голова очереди (head, front) — первый элемент очереди. Хвост очереди (tail, back) — последний элемент очереди. Пример из жизни: очередь в магазине.

**Интерфейс структуры**

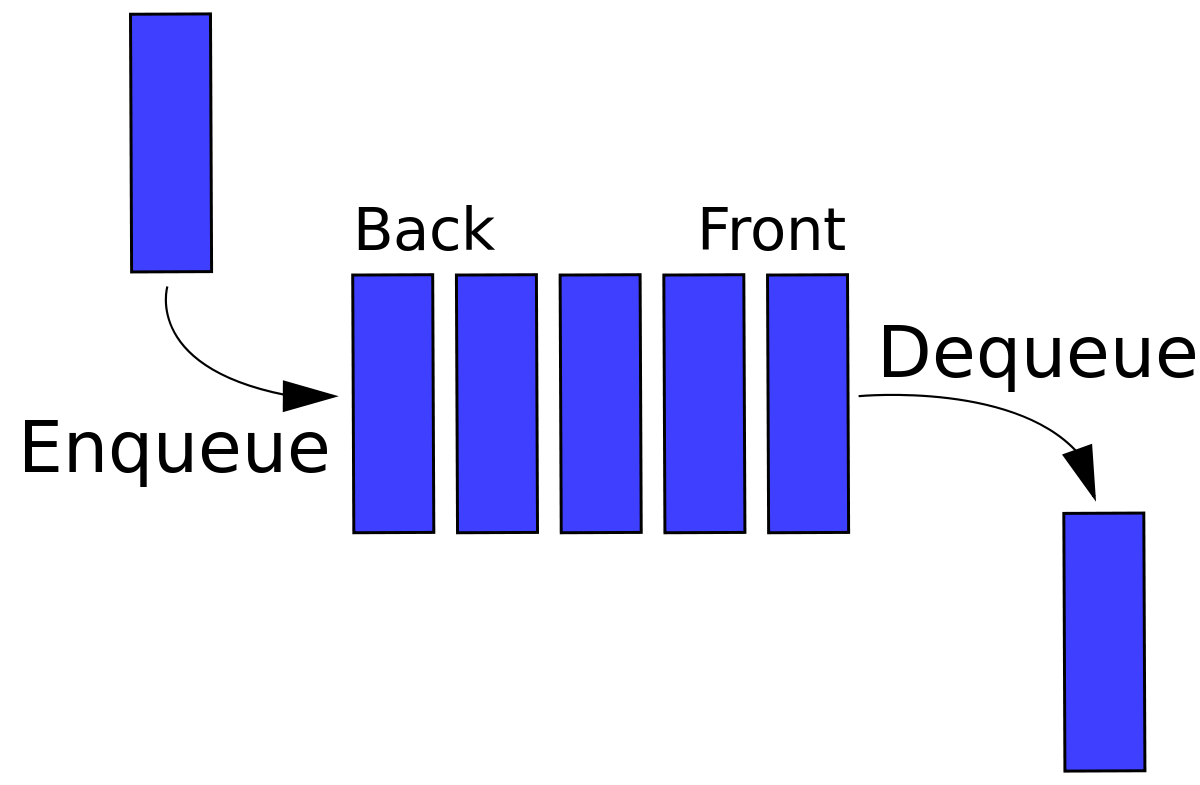
Интерфейс структуры (возможные операции со структурой):

1) enqueue — поставить элемент в очередь;

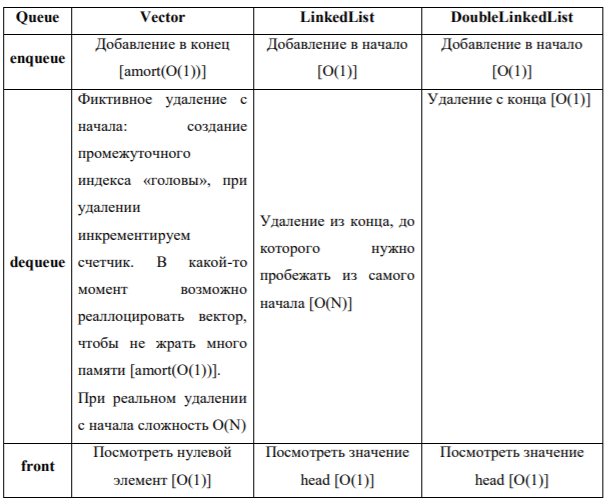
2) dequeue — достать первый элемент из очереди;

3) front — посмотреть элемент в голове очереди.

Также могут реализовываться дополнительные методы: size, isEmpty.



**Сложность**



**Реализация**

Аналогично стеку

# Дек

**Принцип работы**

Дек (Deque, двухсторонняя очередь) — обобщение очереди и стека(Можем делать все, что можем в очереди и стеке).

**Интерфейс структуры**

Интерфейс структуры (возможные операции со структурой):

1) pushFront — добавить элемент в начало;

2) popFront — достать первый элемент;

3) pushBack — добавить элемент в конец;

4) popBack — достать последний элемент;

5) back — посмотреть элемент в хвосте;

6) front — посмотреть элемент в голове.

Также могут реализовываться дополнительные методы: size, isEmpty, итераторы, доступ к элементам по индексу (в некоторых реализациях).



**Сложность**

Смотреть стек и очередь

**Реализация**

Популярные реализации деки достаточно удобно и часто применяются, так как структура получается вполне себе универсальной. Классическая реализация — двусвязные списки, реализация в STL — Chunked Vector (список массивов фиксированного размера)