Контейнерные адаптеры

Контейнерные адаптеры

Контейнерный адаптер - обертка над контейнером, предоставляющая ограниченный интерфейс для работы с данными.

В стандартный библиотеке контейнерные адаптеры представлены тремя шаблонными классами:

- `std::stack`
- `std::queue`
- `std::priority_queue`

`std::stack`

`std::stack<T, Container = std::deque<T>>` - предоставляет интерфейс стека.

Хранит данные типа `T` в контейнере `Container` и использует методы `Container` для реализации своих.

- *Конструкторы*: по умолчанию, copy/move, конструктор от контейнера
- *Методы*: `top()`, `empty()`, `size()`, `push(const T&)`, `push(T&&)`, `emplace(Args&&...)`, `pop()`, `swap()`

`Container` должен содержать методы `back`, `push_back`, `emplace_back`, `pop_back`, `empty`, `size`, а также типы-члены `value_type`, `size_type`, `reference`, `const_reference`.

std::stack

```
std::stack<int> s;
s.push(1);
s.emplace(2);
s.top();  // 2
s.pop();
s.top();  // 1
```

```
std::stack<int, std::vector<int>> s(std::vector<int>{1, 2, 3});
s.top();  // 3
```

std::stack

Возможная реализация:

```
template <class T, class Container = std::deque<T>>
class stack {
  Container container_;

 public:
  using const_reference = typename Container::const_reference;
  // ... other member types

  stack() = default;
  stack(const Container& container) : container_(container) {
  }

  void push(const T& value) { container_.push_back(value); }
  const_reference top() const { return container_.back(); }
  // ... other methods
};
```

`std::queue`

`std::queue<T, Container = std::deque<T>>` - предоставляет интерфейс очереди.

Хранит данные типа `T` в контейнере `Container` и использует методы `Container` для реализации своих.

- *Конструкторы*: по умолчанию, copy/move, конструктор от контейнера
- *Методы*: `front()`, `back()`, `empty()`, `size()`, `push(const T&)`, `push(T&&)`, `emplace(Args&&...)`, `pop()`, `swap()`

`Container` должен содержать методы `back`, `front`, `push_back`, `emplace_back`, `pop_front`, `empty`, `size`, а также типы-члены `value_type`, `size_type`, `reference`, `const_reference`.

`std::priority_queue`

`std::priority_queue<T, Container = std::vector<T>, Compare = std::less<T>>` - предоставляет интерфейс очереди с приоритетами.

Хранит данные типа `T` в контейнере `Container` и использует методы `Container` для реализации своих. На вершине очереди находится наибольший элемент с точки зрения `Compare`.

- *Конструкторы*: по умолчанию, copy/move, конструктор от контейнера, конструктор от компаратора, конструктор от компаратора и контейнера
- *Методы*: `top()`, `empty()`, `size()`, `push(const T&)`, `push(T&&)`, `emplace(Args&&...)`, `pop()`, `swap()`

`Container` должен содержать методы `front`, `push_back`, `emplace_back`, `pop_back`, `empty`, `size`, `[]` а также типы-члены `value_type`, `size_type`, `reference`, `const_reference`.