Принципы работы контейнера std::vector из библиотеки <vector>

Представление в памяти. У вектора есть метод .size(), который определяет количество элементов в векторе; и метод .capacity(), который определяет количество элементов, под которые нужно зарезервировать память.

F001	F002	F003	F004	F005	F006	F007
1	2	3	4	5782	4848	1654

vector.size() = 4; vector.capacity() = 7;

Рис. 1. Схема представления вектора в памяти.

Вставка. Вставка в середину/начало сдвигает все элементы вправо и вставляет элемент. Например, если в начало вставить элемент со значением 0, то в F001 (на рис. 1.) поместится значение 0, а в F002 перейдет значение 1 и последующие элементы также сдвинутся на одну ячейку.

Вставка в конец добавляет элементы в хвост вектора (на рис.1. хвост вектора – ячейки F005-F007). Например, если вставить элемент со значением 5, то в ячейку F005 (на рис. 1.) поместится значение 5.

Удаление. Удаление происходит сдвигом элементов, так как все элементы после удаленного, сдвигаются на один по адресу. Например, если удалить элемент со значением 3, то тогда в ячейку F003 (на рис. 1.) перейдет значение 4 и т.д.

Сравнение с TVector.

Представление в памяти и вставки в TVector будут осуществляться так же как и в std::vector. Удаление в TVector будет отличаться от удаления в std::vector. В TVector у ячеек будут статусы их состояние и если значение в ячейке будет удалено, то смещения элементов происходить не будет, только лишь измениться статус ячейки с busy на delete.

Приложение А: проведение эксперимента.

```cpp
#include <iostream>
#include <vector>

void print_vector_info(std::vector<int> vec) {
    std::cout << vec.size() << " " << vec.capacity() << std::endl;
    for (int i = 0; i < vec.size(); i++) {
        std::cout << vec[i] << "(" << &vec[i] << ")";
    }
    std::cout << std::endl;
}

int main() {
    std::vector<int> vec({ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 });
    print_vector_info(vec);
    for (int i = 0; i < 3; i++) {
        vec.push_back(111 * (i + 1));
    }
    print_vector_info(vec);
    int c = 2;
    for (int i = 0; i < 3; i++) {
        vec.insert(vec.begin() + c, (i + 1) * 5);
        c++;
    }
    print_vector_info(vec);
    vec.erase(vec.begin() + 4);
    print_vector_info(vec);
    system("pause");
    return 0;
}
```