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

Эксперимент производился кодом, приложенным в приложении А.

Представление в памяти.

У вектора есть метод .size(), это количество реальных объектов; и .capacity(), это количество объектов, под которые зарезервирована память.

Резервировать память при необходимости можно самостоятельно методом .reserve().

При разных действиях (вставка, удаление) и если не резервировать память самому, то вектор не выделяет под себя память заранее, а пересчитывает её каждый раз при изменении, что может привести к замедлению работы.

Позиция в векторе определяется не числом от 0,1,2 и т.д. А итератором, обычно, vector.begin() + <позиция>.

00001E……10	00001E……14	00001E……18	00001E……11C	00001E……14
0	0	0	0	0

Схема представления в памяти

Вставка.

Вставка происходит методами .insert() и .push_back(). Они вставляют значение/значения (если передается массив) и увеличивают size и capacity (если не было выделено заранее) вектора. Адреса всех ячеек меняются. Валидация всех ячеек происходит если у вектора изменился capacity, если нет, то ячеек после той, в которую что-то вставили.[1]

Удаление.

Удаление происходит с помощью методов .erase() и .pop_back(). Первый очищает элемент/ы как и по позиции, так и по принципу: «с какой позиции, до какой позиции». Второй же очищает последний элемент. Изменения адресов ячеек не происходит. Удаленная ячейка заполняется следующей, а следующая следующей и т.д. (Последовательно, смещение всех непустых ячеек в пустые)

Также, для очистки всего вектора есть метод .clear(), он очищает все ячейки и не сохраняет их адреса и но оставляет capacity вектора.

Замена значений.

Заменять значения в векторе можно методом .emplace() и методом .emplace_back().

Первый заменяет значение ячейки по позиции на значение, указанное вторым параметром.

А второй заменяет значение последней ячейки на значение, указанное в параметре.

Адреса не меняются.

Сравнение с TVector.

- Нет статусов у ячеек;
- Адресация происходит в методах происходит по схеме: итератор + <число>, а не просто [<число>]

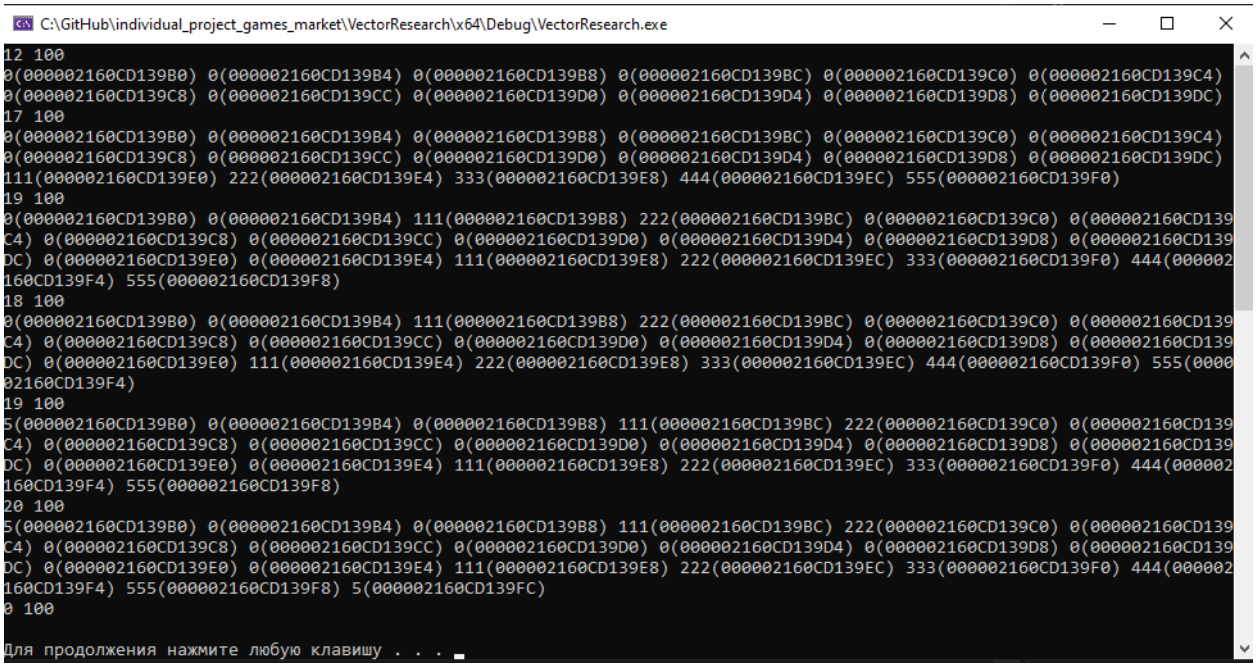

Рис. 1 - Запуск тестовой программы с выводом адресов

Приложение А: проведение эксперимента

```
#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(12);
    vec.reserve(100);
    print_vector_info(vec);
    for (int i = 0; i < 5; i++) { vec.push_back(111 * (i + 1)); }
    print_vector_info(vec);
    vec.insert(vec.begin() + 2, { 111, 222 });
    print_vector_info(vec);
    vec.erase(vec.begin() + 4);
    print_vector_info(vec);
    vec.emplace(vec.begin(), 5);
    print_vector_info(vec);
    vec.emplace_back(5);
    print_vector_info(vec);
    vec.clear();
    print_vector_info(vec);
    system("pause");
    return 0;
```

```
}
```

Сноски:
[1] - std::vector - cppreference.com