

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

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

Capacity увеличивается приблизительно на 30% от первого значения, которое вышло за пределы capacity.

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

void print_vector_info(std::vector<int> vec) {
    std::cout << vec.size() << " ";
    int cap = vec.capacity();
    std::cout << cap << 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);

    for (int i = 0; i < 50; i++) {
        std::cout << "Size / Capacity:  " << vec.size()
        << " ";
        std::cout << vec.capacity() << std::endl;
        vec.push_back(1 * (i + 1)); }
    /*print_vector_info(vec);

    for (int i = 0; i < 5; i++) { vec.push_back(111 * (i
+ 1)); }
    print_vector_info(vec);
    vec.erase(vec.begin() + 4);
    print_vector_info(vec);
    for (int i = 0; i < vec.size(); i++) {
        vec.erase(vec.begin());
    }
    print_vector_info(vec);*/
    system("pause");
    return 0;
}
```

C:\Users\user\source\repos\...

```
Size / Capacity:  1 1
Size / Capacity:  2 2
Size / Capacity:  3 3
Size / Capacity:  4 4
Size / Capacity:  5 6
Size / Capacity:  6 6
Size / Capacity:  7 9
Size / Capacity:  8 9
Size / Capacity:  9 9
Size / Capacity: 10 13
Size / Capacity: 11 13
Size / Capacity: 12 13
Size / Capacity: 13 13
Size / Capacity: 14 19
Size / Capacity: 15 19
Size / Capacity: 16 19
Size / Capacity: 17 19
Size / Capacity: 18 19
Size / Capacity: 19 19
Size / Capacity: 20 28
Size / Capacity: 21 28
Size / Capacity: 22 28
Size / Capacity: 23 28
Size / Capacity: 24 28
Size / Capacity: 25 28
Size / Capacity: 26 28
Size / Capacity: 27 28
Size / Capacity: 28 28
Size / Capacity: 29 42
Size / Capacity: 30 42
Size / Capacity: 31 42
Size / Capacity: 32 42
Size / Capacity: 33 42
Size / Capacity: 34 42
Size / Capacity: 35 42
Size / Capacity: 36 42
Size / Capacity: 37 42
Size / Capacity: 38 42
Size / Capacity: 39 42
Size / Capacity: 40 42
Size / Capacity: 41 42
Size / Capacity: 42 42
Size / Capacity: 43 63
Size / Capacity: 44 63
Size / Capacity: 45 63
```

\*схема представления вектора в памяти\*

## Вставка.

При добавлении элементов адреса всех ячеек меняются.

## Удаление.

При удалении элементов адреса ячеек не меняются. Элементы «сдвигаются» вперед.

```
C:\Users\user\source\repos\Projects_1k2s\64\Debug\Sandbox.exe
15 15
1(000001B16D076320) 2(000001B16D076324) 3(000001B16D076328) 4(000001B16D07632C) 5(000001B16D076330) 6(000001B16D076334)
7(000001B16D076338) 8(000001B16D07633C) 9(000001B16D076340) 10(000001B16D076344) 11(000001B16D076348) 12(000001B16D07634C)
13(000001B16D076350) 14(000001B16D076354) 15(000001B16D076358)
20 20
1(000001B16D0738A0) 2(000001B16D0738A4) 3(000001B16D0738A8) 4(000001B16D0738AC) 5(000001B16D0738B0) 6(000001B16D0738B4)
7(000001B16D0738B8) 8(000001B16D0738BC) 9(000001B16D0738C0) 10(000001B16D0738C4) 11(000001B16D0738C8) 12(000001B16D0738CC)
13(000001B16D0738D0) 14(000001B16D0738D4) 15(000001B16D0738D8) 111(000001B16D0738DC) 222(000001B16D0738E0) 333(000001B16D0738E4)
444(000001B16D0738E8) 555(000001B16D0738EC)
19 19
1(000001B16D0738A0) 2(000001B16D0738A4) 3(000001B16D0738A8) 4(000001B16D0738AC) 6(000001B16D0738B0) 7(000001B16D0738B4)
8(000001B16D0738B8) 9(000001B16D0738BC) 10(000001B16D0738C0) 11(000001B16D0738C4) 12(000001B16D0738C8) 13(000001B16D0738CC)
14(000001B16D0738D0) 15(000001B16D0738D4) 111(000001B16D0738D8) 222(000001B16D0738DC) 333(000001B16D0738E0) 444(000001B16D0738E4)
555(000001B16D0738E8)
Для продолжения нажмите любую клавишу . . .
```

## Сравнение с TVector. ...

```
C:\UNN\Teaching\Classworks\Classworks\64\Debug\TestAll.exe
12 12
17 18
0(0000017A5EF76530) 0(0000017A5EF76534) 0(0000017A5EF76538) 0(0000017A5EF7653C)
0(0000017A5EF76540) 0(0000017A5EF76544) 0(0000017A5EF76548) 0(0000017A5EF7654C)
0(0000017A5EF76550) 0(0000017A5EF76554) 0(0000017A5EF76558) 0(0000017A5EF7655C)
111(0000017A5EF76560) 222(0000017A5EF76564) 333(0000017A5EF76568) 444(0000017A5EF7656C)
555(0000017A5EF76570)
16 18
0(0000017A5EF76530) 0(0000017A5EF76534) 0(0000017A5EF76538) 0(0000017A5EF7653C)
0(0000017A5EF76540) 0(0000017A5EF76544) 0(0000017A5EF76548) 0(0000017A5EF7654C)
0(0000017A5EF76550) 0(0000017A5EF76554) 0(0000017A5EF76558) 111(0000017A5EF7655C)
222(0000017A5EF76560) 333(0000017A5EF76564) 444(0000017A5EF76568) 555(0000017A5EF7656C)
Для продолжения нажмите любую клавишу . . .
```

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);
    print_vector_info(vec);
    for (int i = 0; i < 5; i++) { vec.push_back(111 * (i + 1)); }
    print_vector_info(vec);
    vec.erase(vec.begin() + 4);
    print_vector_info(vec);
    system("pause");
    return 0;
}
```