

*«Функция capacity() возвращает размер текущего выделенного пространства для хранения вектора. Эта ёмкость **не обязательно** равна размеру вектора. Она может быть равна или больше, при этом дополнительное пространство позволяет учитывать рост без необходимости перераспределения при каждом вставке.»*

1. Push back - добавление элементов в конец

```
initial values
size: 1, capacity: 1
1(000002665BEAA1D0)

push back
size: 7, capacity: 9
1(000002665BEB01A0)
2(000002665BEB01A4)
3(000002665BEB01A8)
4(000002665BEB01AC)
5(000002665BEB01B0)
6(000002665BEB01B4)
7(000002665BEB01B8)
```

2. Push selected position - добавление элементов с выбранной позиции

```
push selected position
size: 8, capacity: 9
1(0000028C3E57FDE0)
111(0000028C3E57FDE4)
111(0000028C3E57FDE8)
111(0000028C3E57FDEC)
2(0000028C3E57FDF0)
3(0000028C3E57FDF4)
4(0000028C3E57FDF8)
5(0000028C3E57FDFC)
```

3. Push begin – добавление элементов в начало

```
push begin
size: 10, capacity: 13
222(000001F0B0E7DE00)
222(000001F0B0E7DE04)
1(000001F0B0E7DE08)
111(000001F0B0E7DE0C)
111(000001F0B0E7DE10)
111(000001F0B0E7DE14)
2(000001F0B0E7DE18)
3(000001F0B0E7DE1C)
4(000001F0B0E7DE20)
5(000001F0B0E7DE24)
```

4. Pop back – удаление элементов с конца

```
push begin
size: 10, capacity: 13
222(00000226EEBED940)
222(00000226EEBED944)
1(00000226EEBED948)
111(00000226EEBED94C)
111(00000226EEBED950)
111(00000226EEBED954)
2(00000226EEBED958)
3(00000226EEBED95C)
4(00000226EEBED960)
5(00000226EEBED964)

Pop back
size: 6, capacity: 13
222(00000226EEBED940)
222(00000226EEBED944)
1(00000226EEBED948)
111(00000226EEBED94C)
111(00000226EEBED950)
111(00000226EEBED954)
```

Заметим, что при удалении четырех элементов с конца `size()` уменьшился на 4. `Capacity()` не изменился, а это значит, что вектор не перераспределял память. Также адреса элементов остались неизменными, это тоже говорит о том, что память не перераспределялась.

5. Pop selected position – удаление элементов с выбранной позиции

```
pop back
size: 6, capacity: 13
222(0000019C90A9DFC0)
222(0000019C90A9DFC4)
1(0000019C90A9DFC8)
111(0000019C90A9DFCC)
111(0000019C90A9DFD0)
111(0000019C90A9DFD4)

pop selected position
size: 3, capacity: 13
222(0000019C90A9DFC0)
111(0000019C90A9DFC4)
111(0000019C90A9DFC8)
```

Заметим, что при удалении трёх элементов с выбранной позиции `size()` уменьшился на 3. `Capacity()` опять же не изменился, а это значит, что вектор не перераспределял память. Я удалила три элемента, на их место сдвинулись оставшиеся элементы. Адреса не изменились.

6. `Pop begin` – удаление элементов в начале

```
pop selected position
size: 3, capacity: 13
222(0000019C90A9DFC0)
111(0000019C90A9DFC4)
111(0000019C90A9DFC8)

pop begin
size: 1, capacity: 13
111(0000019C90A9DFC0)
```

Здесь такая же ситуация: на место удаленных элементов сдвинулись оставшиеся.

Удаление элементов не приводит к перераспределению памяти. Удаленный объект будет уничтожен, но память по-прежнему будет принадлежать вектору.

Если после удаления элементов я снова начну добавлять новые элементы, перераспределение памяти не произойдет до тех пор, пока остаются свободные ячейки. Новые элементы могут быть размещены в уже выделенной памяти, и только когда количество добавляемых элементов

превысит текущую емкость вектора, произойдет перераспределение памяти.

В случае, когда я добавила 12 элементов, в `capacity()` было достаточно свободных ячеек, и, следовательно, не потребовалось перераспределение памяти.

Если `size()` и `capacity()` возвращают одинаковые значения, значит, в контейнере не осталось свободного места, и следующая вставка (`insert`, `push_back` и т. д.) вызовет процедуру перераспределения памяти.

Теперь, когда я добавила 13 элементов, это привело к увеличению `capacity()`, что свидетельствует о том, что произошло перераспределение памяти.

push back	push back
size: 14, capacity: 19	size: 13, capacity: 13
111(000002AD124D6130)	111(0000029EEB6005F0)
2(000002AD124D6134)	2(0000029EEB6005F4)
3(000002AD124D6138)	3(0000029EEB6005F8)
4(000002AD124D613C)	4(0000029EEB6005FC)
5(000002AD124D6140)	5(0000029EEB600600)
6(000002AD124D6144)	6(0000029EEB600604)
7(000002AD124D6148)	7(0000029EEB600608)
8(000002AD124D614C)	8(0000029EEB60060C)
9(000002AD124D6150)	9(0000029EEB600610)
10(000002AD124D6154)	10(0000029EEB600614)
11(000002AD124D6158)	11(0000029EEB600618)
12(000002AD124D615C)	12(0000029EEB60061C)
13(000002AD124D6160)	13(0000029EEB600620)
14(000002AD124D6164)	