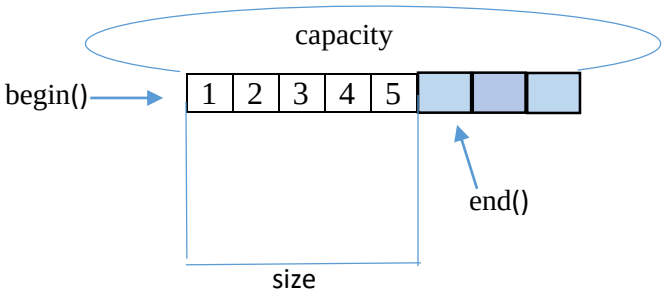
Принципы работы контейнера std::vector из библиотеки <vector>

Представление в памяти: у вектора есть метод .size() (т.е. фактическое количество реальных объектов в нём) и .capacity() (т.е. то количество объектов, под которые зарезервирована память); capacity – это размер (количество элементов) вектора + некоторый резерв памяти.



begin() и end() – это методы вектора, возвращающие итераторы – объекты, похожие на указатели. begin() вернёт итератор, указывающий на первый элемент вектора, end() – итератор, указывающий на позицию после последнего элемента

Вставка: у вектора есть метод .push_back(), который вставляет заданный элемент в конец. Если capacity переполнено (т.е. size вектора становится больше capacity), происходит так называемая реаллокация, или перераспределение памяти.

Новый блок памяти увеличивается в 1,5 раза (см. рис. 1 ниже: создаётся вектор с size = capacity = 10, при вставке новых элементов происходит реаллокация, size = capacity = 10 * 1,5 = 15, а после удаления элемента size становится 14, capacity же не изменяется и так же равно 15).

Также из рис. 1 видно, что после перевыделения памяти адреса элементов, лежащих в векторе меняются (из-за того, что при реаллокации мы создаём новый блок памяти и перемещаем туда существующие элементы, а не просто увеличиваем старый блок; соотвественно, в новом блоке памяти у элементов будут новые адреса)

Удаление: осуществляется при помощи метода .erase(); элементы, следующие за удалённым, сдвигаются влево, но при этом удаление элементов не приводит к перераспределению памяти: удаленный из вектора элемент будет уничтожен, но память останется принадлежать вектору

Сравнение с Tvector:

1) В Tvector мы реализуем подход вставки элемнтов достаточно схожий с подходом в std::vector: при создании массива выделяем память под хранение элементов с запасом, и когда он переполняется, создаём новый массив с увеличенным размером памяти.

2) в TVector, в отличие от std::vector, при удалении элементов у нас происходит перераспределение памяти (при условии, что количество удаленных элементов стало превышать 15% от размера массива, производится перераспределение памяти: создаётся новый массив с размером памяти на 15 элементов больше

количества реально заполненных элементов, копируются в новый массив элементы старого массива только с состоянием busy, очищается память старого массива, указателю на старый массив присваивается адрес памяти нового массива, обнуляется счётчик «мнимо удалённых» элементов, обновляется счётчик количества заполненных элементов).

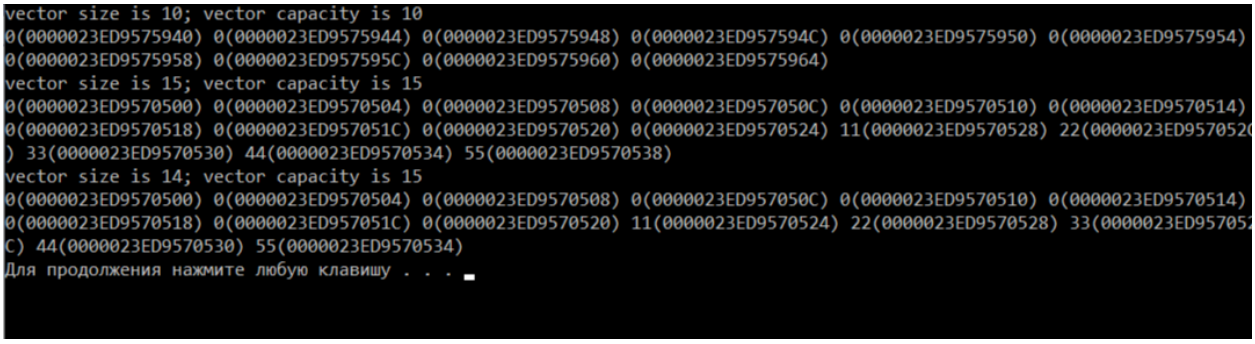


рис.1

Приложение: проведение эксперимента

```
#include <iostream>
#include <vector>
void print_info_about_vector(std::vector<int>& v) {
	std::cout << "vector size is " << v.size() << "; " << "vector capacity is " << v.capacity()
<< "\n";
	for (int i = 0; i != v.size(); ++i) {
		std::cout << v[i] << "(" << &v[i] << ") ";

	}
	std::cout << std::endl;
}
int main() {
	std::vector<int> v(10);
	print_info_about_vector(v);
	for (int i = 0; i < 5; ++i) {
		v.push_back(11 * (i+1));
	}
	print_info_about_vector(v);
	v.erase(v.begin() + 4);
	print_info_about_vector(v);
	system("pause");
	return 0;
}
```