Задача 3 (Пишем свой вектор)

Условие

В методических материалах мы уже начали реализовывать свой вектор. В этой задаче вам надо его развить: добавить методы Size, Capacity и PushBack. Пришлите на проверку заголовочный файл simple_vector.h, содержащий объявление и определение шаблона класса SimpleVector:

```
1  template <typename T>
2  class SimpleVector {
3  public:
4  SimpleVector() = default;
5  explicit SimpleVector(size_t size);
6  ~SimpleVector();
7
8  T& operator[](size_t index);
9
10 T* begin();
11 T* end();
12
13 size_t Size() const;
14 size_t Capacity() const;
15 void PushBack(const T& value);
16
17 private:
18 ...
19 };
```

Требования:

- метод Capacity должен возвращать текущую ёмкость вектора — количество элементов, которое помещается в блок памяти, выделенный вектором в данный момент
- метод Size должен возвращать количество элементов в векторе
- метод PushBack добавляет новый элемент в конец вектора; если в текущем выделенном блоке памяти не осталось свободного места (т.е. Size() == Capacity()), вектор должен выделить блок размера 2 * Capacity(), скопировать в него все элементы и удалить старый.
- первый вызов метода PushBack для вновь созданного объекта должен делать ёмкость, равной единице
- метод PushBack должен иметь амортизированную константную сложность
- методы begin и end должны возвращать итераторы на текущие начало и конец вектора
- в деструкторе должен освобождаться текущий блок памяти, выделенный вектором
- также см. дополнительные требования к работе SimpleVector в юнит-тестах в приложенном шаблоне решения

Заготовка решения

Файл: simple_vector.h
Файл: simple_vector.cpp

Замечание

Заголовочный файл, который вы пришлёте на проверку, не должен подключать файлы <vector>, <list>, <forward_list>, <deque>, <map>. Если у вас будет подключен один из этих файлов, вы получите ошибку компиляции.

Подсказка

Наверняка в вашей реализации шаблона класса SimpleVector будет поле, являющееся указателем. В конструкторе по умолчанию вам надо будет его чем-нибудь проинициализировать. В лекциях мы рассматривали только один способ инициализации указателей — с помощью оператора new. В C++ есть специальное значение, означающее указатель, который ни на что не указывает — nullptr:

```
1  int* p = nullptr;
2  string* q = nullptr;
3  map<string, vector<int>>* r = nullptr;
```

Вы можете использовать nullptr для инициализации указателя в конструкторе по умолчанию.