# Задание по программированию: Пишем свой вектор

### Условие

В лекциях 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:

template <typename T>

class SimpleVector {

public:

  SimpleVector() = default;

  explicit SimpleVector(size\_t size);

  ~SimpleVector();

  T& operator[](size\_t index);

  T\* begin();

  T\* end();

  size\_t Size() const;

  size\_t Capacity() const;

  void PushBack(const T& value);

private:

  ...

};

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 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

[simple\_vector.h](https://d3c33hcgiwev3.cloudfront.net/q-OL4qX_EeilzRLZf2WxfA_ac4e8270a5ff11e89fd0455a8819d387_simple_vector.h?Expires=1617062400&Signature=YCDKOopfd9-gzTFMsAQLOpc~KGqqQSvQhcKPB0lgm2hsAnqetbSBq2ZJfxoU0ZBgtso0UDKsuoL75HNubnjFodn4Re0RHF81S3qHLyvUFzS-olqtiVESdFGqzOSTyjHQjrYyKhBMkFzdBT-h60buw1rr~6b-HhDY~vuY7evCdxM_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

[simple\_vector.cpp](https://d3c33hcgiwev3.cloudfront.net/uoPvEoauEeianAr0yIdmDg_bae6cec086ae11e88d9327752d64e780_simple_vector.cpp?Expires=1617062400&Signature=fxKBfl~4eGeKZL~Xpt~fB1UNepZUsJsXdCQT1YNBNdQ56yNopMEHJkg-CGKrR8Yybmz1-CqSsgz~cgRyGeVk96Ii-u1my~Zn2e8J7arKiiuDDqrG2WJ8sQQ9KOOH9WtYSxwomuVRZoDOoyk4iL7YHgNN4N0htoyo7Hs78rgOT3w_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

### 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:

int\* p = nullptr;

string\* q = nullptr;

map<string, vector<int>>\* r = nullptr;

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