МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Объектно-ориентированное программирование»

Тема: Умные указатели

Студент гр. 7304 _____ Нгуен К.Х.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2019

Цель работы.

Исследование умных указателей на языке программирования C ++.

Задание

Необходимо реализовать умный указатель разделяемого владения объектом (shared_ptr).

Для того, чтобы shared_ptr можно было использовать везде, где раньше использовались обычные указатели, он должен полностью поддерживать их семантику. Модифицируйте созданный на предыдущем шаге shared_ptr, чтобы он был пригоден для полиморфного использования. Должны быть обеспечены следующие возможности:

- копирование указателей на полиморфные объекты

```
stepik::shared_ptr<Derived> derivedPtr(new Derived);
stepik::shared_ptr<Base> basePtr = derivedPtr;
```

- сравнение shared_ptr как указателей на хранимые объекты.

Поведение реализованных функций должно быть аналогично функциям std::shared_ptr (http://ru.cppreference.com/w/cpp/memory/shared_ptr).

Требования к реализации: при выполнении этого задания вы можете определять любые вспомогательные функции. Вводить или выводить что-либо не нужно. Реализовывать функцию main не нужно. Не используйте функции из cstdlib (malloc, calloc, realloc и free).

Экспериментальные результаты.

explicit shared_ptr(T *ptr = 0)
Create a smart pointer from a pointer of type T

~shared_ptr()
Деструктор, используйте для удаления этого экземпляра интеллектуального указателя. Если счетчик указателя достиг 0, то есть это последний умный указатель, указывающий на объект, объект уничтожается. Иначе уменьшает счетчик указателя на единицу.

shared_ptr(const shared_ptr<U> & other)
Конструктор копирования, этот и созданный умный указатель совместно используют владельца объекта, на который в данный момент указывает other. Эффективно увеличить счетчик указателей на один.

shared_ptr<T>& operator=(const shared_ptr<U>& other)
Оператор присваивания. Скопируйте значение с правой стороны на левую сторону. Увеличивает счетчик указателей на 1.

bool operator==(const shared_ptr<U>& rhs) const
Сравнивает 2 умных указателя. Возвращает true, если 2 указателя указывают на один и тот же объект

explicit operator bool() const
Возвращает true, если умный указатель управляет объектом

T* get() const
Возвращает указатель на управляемый объект

long use_count() const
Возвращает счетчик указателей (количество умных указателей, указывающих на управляемый объект)

T& operator*() const

T* operator->() const

void swap(shared_ptr& x)
Меняет местами указатели

void reset(T *ptr = 0)
Заменяет управляемого объекта с объектом, на который указывает ptr.

Выводы.

В результате лабораторной работы изучил работу умных указателей на языке C ++. Реализована простая версия умного указателя.