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. 7381 _____ Аженилок В.А.

Преподаватель _____ Жангиров Т.М.

Санкт-Петербург

2019

Цель работы.

Изучить стандартные контейнеры vector и list языка C++.

Задание.

Необходимо реализовать умный указатель разделяемого владения объектом (shared_ptr). 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.

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).

Ход работы.

Shared_ptr – умный указатель, с разделяемым владением объектом через его указатель. Несколько указателей shared_ptr могут владеть одним и тем же объектом; объект будет уничтожен, когда последний shared_ptr, указывающий на него, будет уничтожен или сброшен. Реализуемый класс имеет два поля: указатель на объект и указатель на счётчик указателей на этот объект.

Были реализованы две вспомогательные функции: inc_counter для инкриментирования счётчика умных указателей и deg_counter для декрементирование счётчика и удаления объекта, если счётчик достигает нуля. Конструктор, принимающий С-указатель на объект, для которого инициализируется

новый счётчик, или ссылку на другой shared_ptr, копирую его поля и увеличиваю счётчик на единицу. Деструктор вызывает функцию deg_counter.

Также были реализованы функции get (возвращающая указатель на объект), use_count (возвращающая значение счётчика), swap (обменивающая поля двух умных указателей), reset (заменяющая объект, которым владеет указатель) и перегружены операторы =, ==, !=, <, >, <=, >=, *, -> и bool аналогично обычным указателям.

Реализация класса представлена в приложении А.

Вывод.

В ходе выполнения данной работы был реализован класс shared_ptr, аналогичный классу std::shared_ptr из стандартной библиотеки.

Приложение А. Файл shared_ptr.h.

```
namespace stepik
{
  template <typename T>
class shared_ptr
  {   public:     explicit
shared_ptr(T *ptr = 0)
     {
        this->ptr = ptr;
if(ptr != nullptr)
count = new long(1);
else            count =
nullptr;
        }

    ~shared_ptr()
{
      if(ptr != nullptr){
if((*count) == 1){
delete ptr;           delete
count;
        }
else
          (*count)--;
      }
    }
    shared_ptr(const shared_ptr & other)
    : ptr(other.ptr)
    , count(other.count)
    {
if(ptr)
        (*count)++;
```

```
    }
    shared_ptr& operator=(const shared_ptr & other)
    {
      shared_ptr<T>(other).swap(*this);
return *this;
    }
    template <class Derived>
friend class shared_ptr;

    template <class Derived>
    shared_ptr(const shared_ptr<Derived> & other) : ptr(other.ptr),
count(other.count)
    {
if(ptr)
        (*count)++;
    }
    template <class Derived>
    shared_ptr& operator=(const shared_ptr<Derived> & other)
    {
      shared_ptr<T>(other).swap(*this);
return *this;
    }
    template <class Derived>
    bool operator==(const shared_ptr<Derived> & other) const
    {
      return ptr == other.ptr;
    }
    explicit operator bool() const
    {
      return ptr != nullptr;
    }
```

```
    T* get() const
{       return
ptr;
    }
    long use_count() const
    {
      return ptr == nullptr ? 0 : *count;
    }

    T& operator*() const
    {       return
*ptr;
    }

    T* operator->() const
    {
return ptr;
    }
    void swap(shared_ptr& x) noexcept
    {
      std::swap(ptr, x.ptr);       std::swap(count, x.count);
    }      void reset(T
*ptr = 0)
    {       shared_ptr
temp(ptr);
swap(temp);
    }    private:
T * ptr;
long * count;
};
} // namespace stepik
```