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

ОТЧЕТ

по лабораторной работе № 4

по дисциплине Объектно-ориентированное программирование

Тема: “shared_ptr”.

Студентка гр.7304 _____ Каляева А.В

Преподаватель _____ Размочаева Н.В

г. Санкт-Петербург

2019 г.

Цель работы:

Изучить shared_ptr – умный указатель с разделяемым владением объектом и реализовать его на языке программирования C++.

Задание:

Реализовать умный указатель разделяемого владения объектом (shared_ptr). Поведение реализованных функций должно быть аналогично функциям std::shared_ptr.

Ход работы:

Был реализован класс shared_ptr. Данный класс содержит в себе два приватных поля: T *ptr – указатель на объект типа T и long *count – счетчик, который хранит в себе количество указателей, указывающих на данный объект. Для полного функционирования класса shared_ptr были реализованы следующие методы:

- Конструктор explicit shared_ptr (T *ptr = 0), который принимает указатель на объект и записывает в поле count значение равное 1.
- Были реализованы два конструктора копирования, один из которых для поддержания полиморфизма. Реализованные конструкторы копируют данные объекта other и увеличивают счетчик на 1.
- Были реализованы два оператора присваивания, один их которых для поддержания полиморфизма. Реализованные операторы перемещают данные объекта other и увеличивают счетчик на 1.
- Был реализован деструктор, который уменьшает значение счетчика на 1. Если значение счетчика стало равным 0, то объект удаляется.
- Были реализованы операторы == для сравнения указателей.
- Был реализован метод bool, который возвращает true, если указатель не равен nullptr.
- Был реализован метод use_count, который возвращает количество указателей для данного объекта.
- Был реализован метод swap, который обеспечивает обмен содержимым.
- Был реализован метод reset, который заменяет управляемый объект объектом, на который указывает ptr.

Примеры работы программы:

```
count ptr1: 2
count ptr2: 2
count ptr3: 1
VALUE:
10 10 20
```

Заключение:

В ходе выполнения лабораторной работы были изучены умные указатели и реализован указатель shared_ptr – умный указатель с разделяемым владением объектом . Для заданного указателя были реализованы основные функции для работы с ним. Поведение реализованных функций соответствует классу std::shared_ptr.

Приложение А
Исходный код файла lr4.hpp

```
namespace stepik
{
template <typename T>
class shared_ptr
{
   template <typename T1> friend class shared_ptr;
public:
  explicit shared_ptr(T *ptr = 0):ptr(ptr), count(new long(1))
  {
  }

  ~shared_ptr()
  {
   delete_ptr();
  }

  shared_ptr(const shared_ptr & other):ptr(other.ptr), count(other.count)
  {
   if(count){
      (*count)++;
   }
  }
   template <typename T1>
   shared_ptr(const shared_ptr<T1> & other):ptr(other.ptr), count(other.count)
  {
   if(count){
      (*count)++;
   }
  }

  shared_ptr& operator=(const shared_ptr & other)
  {
   if(ptr!=other.ptr){
      delete_ptr();
      ptr=other.ptr;
      count=other.count;
      (*count)++;
   }
   return *this;
  }

   template <typename T1>
   shared_ptr& operator=(const shared_ptr<T1> & other)
  {
   if(ptr!=other.ptr){
      delete_ptr();
      ptr=other.ptr;
      count=other.count;
      (*count)++;
   }
```

```
  return *this;
 }
  template <typename T1>
  bool operator == (const shared_ptr<T1>& other) const{
     return ptr==other.ptr;
  }

 explicit operator bool() const
 {
  return ptr!=nullptr;
 }

 T* get() const
 {
  return ptr;
 }

 long use_count() const
 {
  return (ptr)?(*count):0;
 }

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

 T* operator->() const
 {
  return ptr;
 }

 void swap(shared_ptr& x) noexcept
 {
  shared_ptr tmp(x);
  x=*this;
  *this=tmp;
 }

 void reset(T *ptr = 0)
 {
  shared_ptr tmp(ptr);
  swap(tmp);
 }

private:
 T *ptr;
 long *count;
 void delete_ptr(){
    if((*count)>0){
       (*count)--;
    }
```

```
        if((*count)==0){
            delete ptr;
            delete count;
        }
    }
  };
} // namespace stepik
```