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 _____ Петруненко Д.А

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2019

Цель работы

Изучить реализацию умного указателя, разделяемого владения объектом в языке программирования c++.

Задача

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.

Ход работы

1. В классе shared_ptr создаются два приватных поля. T* pointer – указатель на объект T. И unsigned * counter – счетчик, который показывает сколько shared_prt указывают на данный объект.
    1.1. Реализуется основной конструктор, который принимает указатель на объект, по умолчанию равен 0. А так же записывает в счетчик 1.
    1.2. Реализуются конструкторы копирования, один обычный, другой для поддержания полиморфизма. Перемещают данные с объекта other. И увеличиваю счетчик на 1.
    1.3. Реализуются операторы перемещения, один обычный, другой для поддержания полиморфизма. Удаляет старый объект, то есть уменьшает счетчик, если он станет равный 0, то удаляет указатель. И дальше перемещает данные с объекта other и увеличивает счетчик на 1.
    1.4. Реализуется деструктор, который уменьшает счетчик на 1, если счетчик будет равен 0, то удаляет объект.
    1.5. Реализуются операторы сравнения для хранимых указателей.

1.6. Реализуется оператор bool(), который выводит true,если указатель не NULL.

1.7. Реализуется метода get(), который возвращает указатель на объект.

1.8. Реализуется метод use_count, который возвращает число указатель на объект.

1.9. Реализуется метод swap, который меняет два shared_ptr местами.

1.10. Реализуется метод reset, который удаляет старый указатель и создает новый.

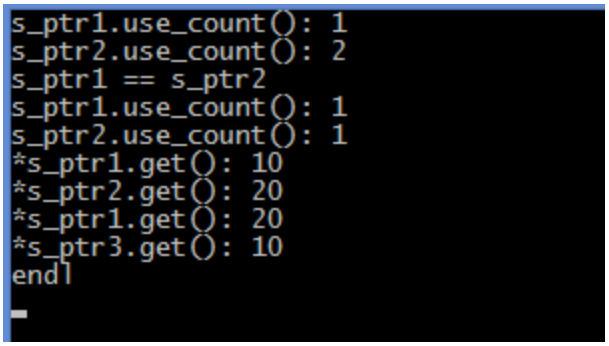
Результат работы

Входные данные

```
int main()
{
    int* ptr1 = new int;
        *ptr1 = 10;
        shared_ptr<int> s_ptr1(ptr1);
        cout << "s_ptr1.use_count(): " << s_ptr1.use_count() << endl;
        int* ptr2 = new int;
        *ptr2 = 20;
        shared_ptr<int> s_ptr2(s_ptr1);
        cout << "s_ptr2.use_count(): " << s_ptr2.use_count() << endl;
        if(s_ptr1 == s_ptr2)
            cout << "s_ptr1 == s_ptr2" << endl;
        else
            cout << "s_ptr1 != s_ptr2" << endl;
        s_ptr2.reset(ptr2);
        cout << "s_ptr1.use_count(): " << s_ptr1.use_count() << endl;
        cout << "s_ptr2.use_count(): " << s_ptr2.use_count() << endl;
        cout << "*s_ptr1.get(): " << *s_ptr1.get() << endl;
        cout << "*s_ptr2.get(): " << *s_ptr2.get() << endl;
        shared_ptr<int> s_ptr3(ptr2);
        s_ptr1.swap(s_ptr3);
        cout << "*s_ptr1.get(): " << *s_ptr1.get() << endl;
        cout << "*s_ptr3.get(): " << *s_ptr3.get() << endl;
        cout<<"endl\n";

    return 0;
}
```

Выходные данные



Вывод

В ходе выполнения данной лабораторной работы была изучена реализация умного указателя, разделяемого владения объектом, и были реализованы основные функции, поведение которых полностью аналогично функциям из стандартной библиотеки. Использование таких указателей сильно облегчает деятельность программиста. Умные указатели помогают избежать множества проблем: «висячие» указатели, «утечки» памяти и отказы в выделении памяти.