

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: «Умные указатели»

Студент гр. 7303

Романенко М.В.

Преподаватель

Размочаева Н.В.

Санкт-Петербург

2019

Цель работы.

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

Задание.

Реализовать базовые методы умного указателя разделяемого владения объектом, после чего модифицировать созданный `shared_ptr` так, чтобы он был пригоден для полиморфного использования. Должны быть обеспечены следующие возможности:

- копирование указателей на полиморфные объекты
- сравнение `shared_ptr` как указателей на хранимые объекты.

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`).

Ход работы.

Реализация `shared_ptr`. В качестве полей были заведены: указатель на объект типа `T`, и счетчик для подсчета количества умных указателей, которые ссылаются на один объект. Также была заведена функция `count_dec()`, осуществляющая уменьшение счетчика, и при необходимости удаляющая поля указателя.

Были реализованы: конструктор, принимающий указатель, и деструктор. В конструкторе поле, отвечающее за указатель, задается переданным указателем, а счетчик инициализируется значением 1. Деструктор вызывает функцию `count_dec()`, которая удалит данные при

разрушении умного указателя только в том случае, если данный умный указатель – единственный.

Были реализованы: оператор присваивания, оператор `bool()` (проверяет, указывает ли указатель на объект), методы `get()` (позволяет получить хранимый указатель на данные), `use_count()` (возвращает счетчик), операторы `*` и `->`. Были реализованы методы `swap()` и `reset()` (для замены объекта, которым владеет умный указатель).

Для полиморфного использования внутри класса был объявлен friend-класс `shared_ptr` другого типа (связанного с типом реализуемого указателя посредством наследования). Были переписаны конструктор копирования и оператор присваивания с использованием `template`, чтобы иметь возможность конструировать, например, умные указатели на объекты базового класса через указатели на объекты наследников. Также были добавлены операторы `!=` и `==`, чтобы умные указатели можно было сравнивать как указатели на хранимые объекты.

Выводы.

В ходе выполнения лабораторной работы был реализован класс, аналогичный классу `std::shared_ptr` из стандартной библиотеки. Данный умный указатель с разделяемым владением позволяет не заботиться об освобождении памяти для объекта, доступ к которому прекращён, поскольку это происходит автоматически.