

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”**

Факультет программной инженерии и компьютерной техники

Направление подготовки (специальность) СППО

ОТЧЕТ

о практической работе №2

По дисциплине: программирование на C++

Обучающийся Розмирский Д.В. Р4114
(Фамилия И.О.) (номер группы)

Санкт-Петербург
2024 г.

Отчет по лабораторной работе №2

Введение

В данной лабораторной работе была поставлена задача спроектировать класс, который представляет собой структуру, хранящую данные в динамической памяти. Для хранения данных был использован пользовательский узел, содержащий значение (value) и приоритет (priority). Основной задачей было изучить и реализовать все типы конструкторов и перегрузки операторов присваивания для данного класса, а также продемонстрировать их работу в различных ситуациях.

Задачи лабораторной работы

1. Реализовать класс, хранящий данные в динамической памяти, включая следующие методы:
 - Конструктор с параметрами.
 - Конструктор копирования и конструктор перемещения.
 - Оператор присваивания (с копированием и перемещением).
 - Деструктор.
 - Методы доступа и изменения значений.
2. Протестировать работу класса в программе, создавая статические и динамические экземпляры класса.
3. Продемонстрировать копирование, перемещение объектов, передачу экземпляров в функции по значению и по ссылке, возврат объектов из функции.
4. Создать вектор и список экземпляров класса и реализовать функции для обработки этих коллекций.

Описание класса myClass

Класс myClass представляет собой объект, который хранит данные о значении (value) и приоритете (priority). Эти данные хранятся в динамической памяти в структуре Node. Основная реализация класса и его методов разделена на следующие файлы:

- myClass.hpp — заголовочный файл, содержащий определение класса и его методов.
- myClass.cpp — файл с реализацией методов класса.
- main.cpp — основной файл, содержащий код для тестирования класса.

Реализация методов класса

Конструкторы

1. Конструктор с параметрами:
 - Создает узел с заданным значением и приоритетом.
 - Сообщает о своем вызове: Constructor was called with param:
2. Конструктор копирования:
 - Создает новый объект путем копирования значения и приоритета из другого экземпляра.
 - Использует глубокое копирование: выделяет новую область памяти и копирует значения.
 - Сообщает о своем вызове: Copy constructor was called with param:
3. Конструктор перемещения:
 - Перемещает данные из другого экземпляра, не выполняя глубокого копирования.
 - Указатель на данные из другого экземпляра присваивается текущему, а указатель у другого экземпляра обнуляется.
 - Сообщает о своем вызове: Move constructor was called with param:

Операторы присваивания

1. Оператор присваивания с копированием:
 - Удаляет существующий узел, если он есть, и создает новый узел путем копирования значений из другого экземпляра.
 - Сообщает о своем вызове: Assignment operator was called with param:
2. Оператор присваивания с перемещением:
 - Удаляет существующий узел и перемещает данные из другого экземпляра, обнуляя указатель у другого экземпляра.
 - Сообщает о своем вызове: Move assignment operator was called with param:

Деструктор

- Удаляет узел, если он существует, и освобождает память.
- Сообщает о своем вызове: Destructor was called.

Пример работы с классом

В main.cpp были выполнены следующие действия:

1. Создание статических и динамических экземпляров класса:
 - Создан статический объект А и динамический объект В.

- Сообщение о вызове конструктора для каждого из объектов.
- 2. Копирование и перемещение объектов:
 - Создан объект C путем копирования объекта A.
 - Создан объект D путем перемещения объекта A.
- 3. Использование операторов присваивания:
 - Объект E присваивается объекту C (копирование).
 - Объект F присваивается объекту C с использованием перемещения.
- 4. Передача объектов в функции:
 - Объект D передается по значению и по ссылке в функции `byValue()` и `byReference()`.
- 5. Возврат объектов из функции:
 - Функция `returnObject()` создает временный объект и возвращает его.
- 6. Работа с контейнерами `vector` и `list`:
 - Созданы вектор `myVector` и список `myList` с экземплярами класса `myClass`.
 - Список и вектор заполняются элементами, после чего проводится обработка элементов.

Вывод результатам работы программы

```
Creating static and dynamic instances of myClass:
Constructor was called with param: 2 3
Constructor was called with param: 12 4

Copying and moving objects:
Copy constructor was called with param: 2 3
Move constructor was called with param: 2 3

Using assignment operators:
Constructor was called with param: 0 0
Assignment operator was called with param : 2 3
Constructor was called with param: 0 0
Copy assignment operator was called with param : 2 3

Passing objects to functions:
Copy constructor was called with param: 2 3
Function received object by value: 2, 3
Destructor was called
Function received object by reference: 2, 3

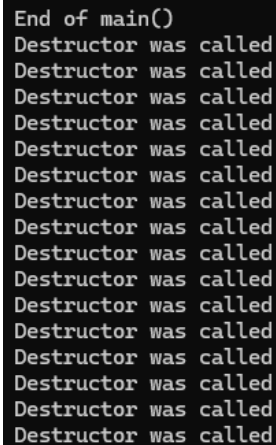
Returning objects from functions:
Constructor was called with param: 42 100
Returning object from function

Creating vector and list of myClass instances:
Constructor was called with param: 0 1
Constructor was called with param: 1 2
Move constructor was called with param: 0 1
Destructor was called
Constructor was called with param: 2 3
Move constructor was called with param: 0 1
Move constructor was called with param: 1 2
Destructor was called
Destructor was called
Constructor was called with param: 3 4
Move constructor was called with param: 0 1
Move constructor was called with param: 1 2
Move constructor was called with param: 2 3
Destructor was called
Destructor was called
Destructor was called
Constructor was called with param: 4 5
Move constructor was called with param: 0 1
Move constructor was called with param: 1 2
Move constructor was called with param: 2 3
Move constructor was called with param: 3 4
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Constructor was called with param: 5 6
Constructor was called with param: 6 7
Constructor was called with param: 7 8
Constructor was called with param: 8 9
Constructor was called with param: 9 10

Processing vector elements:
Processing vector element: 0, 1
Processing vector element: 1, 2
Processing vector element: 2, 3
Processing vector element: 3, 4
Processing vector element: 4, 5

Processing list elements:
Processing list element: 5, 6
Processing list element: 6, 7
Processing list element: 7, 8
Processing list element: 8, 9
Processing list element: 9, 10
Destructor was called
```

Рисунок 1 – Вывод результатов в терминал



```
End of main()
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
Destructor was called
```

Рисунок 2 – Окончание работы `main()` и вызов деструкторов

Анализ программы

1. Корректное использование динамической памяти:
 - Класс корректно использует динамическую память для хранения данных.
 - Реализованы все необходимые методы управления памятью, в том числе деструктор и операторы присваивания.
2. Перемещение и копирование:
 - Конструкторы перемещения и копирования, а также соответствующие операторы позволяют эффективно управлять ресурсами и избегать лишнего копирования.
3. Работа с контейнерами:
 - Вектор и список используются для хранения экземпляров класса `myClass`. Это позволило увидеть работу конструкторов и операторов присваивания при добавлении и удалении элементов из контейнеров.
4. Основные недостатки:
 - В текущей версии программы нет проверки на корректность выделения памяти (`new`). В случае ошибки выделения памяти (`nullptr`) программа не сможет корректно продолжить выполнение.
 - Использование сырых указателей для динамической памяти увеличивает риск ошибок. В будущем можно улучшить реализацию, применяя умные указатели (`std::unique_ptr` или `std::shared_ptr`) для управления динамической памятью.

Вывод

В ходе лабораторной работы №2 была изучена и реализована работа с динамической памятью в C++ через проектирование и реализацию класса `myClass`. Были рассмотрены основные аспекты управления динамической памятью, включая конструкторы, операторы присваивания и деструктор. Работа также позволила на практике применить знания о копировании и перемещении объектов, а также продемонстрировать работу с контейнерами стандартной библиотеки (`vector` и `list`). Реализация программы показала важность управления ресурсами и предотвращения утечек памяти при использовании динамической памяти в C++.

Исходный код:

https://github.com/RozmiDan/cpp_itmo_labs/tree/main/scnd_lab_cpp