Кирилл Садретдинов
СКБ 241

Отчет: Лабораторная работа №10

Приоритетная очередь

1. **В ходе** лабораторной работы я создал class с вложенным в него типом struct. Реализовал перегрузку вывода класса в поток.
2. **Сведения о системе**
    1. **Операционная система:** macOS Sequoia 15.0
    2. **Компилятор:** Apple clang version 15.0.0 (clang-1500.3.9.4). Является частью Command line tools, из-за чего имеет нативную поддержку компиляции под ARM. Схож с Clang.
    3. **IDE:** Visual Studio Code 1.93. С самого начала всегда пользовался только VSC, очень удобен, имеет большие количество расширений (pylance, docker, thunderclient, tabnine), поддерживает все современные языки программирования, очень удобная отладка.
    4. **Git** version 2.39.3 (Apple Git-146)
    5. **GitHub Desktop** 3.4.5
3. **Процедура**

3.1. Разработал шаблонный класс PriorityQueue, реализующий приоритетную очередь для хранения элементов с учетом их приоритета.

3.2. Реализовал основные методы класса:

- push(item, priority): добавляет элемент в очередь с учетом приоритета.
- pop(): извлекает элемент с наивысшим приоритетом и удаляет его.
- peek(): извлекает элемент с наивысшим приоритетом без удаления.
- size(): возвращает количество активных элементов в очереди.

3.3. Перегрузил оператор << для удобного вывода содержимого очереди, включая количество элементов, максимальный приоритет и следующий элемент.

3.4. В main проиллюстрировал работу класса, создав объект PriorityQueue<char> и добавив в него несколько элементов с разными приоритетами.

3.5. Обеспечил корректную обработку граничных случаев, таких как попытка извлечения элемента из пустой очереди или добавление элемента при превышении заданной длины очереди.

3.6. Провел тестирование работы основных методов, выявил и исправил ошибки, включая корректное обновление максимального приоритета после удаления элементов.

3.7. Оптимизировал код, используя динамическое выделение памяти для массива элементов очереди и обеспечив корректное освобождение памяти в деструкторе.

4. **Код** (GitHub)

github.com/hosternus/cpp-lab-10

5. **Заключение**

В процессе работы удалось реализовать функционал приоритетной очереди и обеспечить корректную обработку исключительных ситуаций, таких как удаление элементов из пустой очереди и превышение её заданной длины. Основная сложность заключалась в правильном управлении динамической памятью и отслеживании состояния элементов очереди. Программа протестирована на типовых сценариях, ошибок утечек памяти или некорректного поведения не выявлено.