

**Міністерство освіти і науки України
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА
ФРАНКА
Факультет прикладної математики та інформатики**

Кафедра програмування

**ЛАБОРАТОРНА РОБОТА № 7
Черга та черга з пріоритетом
з курсу “Алгоритми та структури даних”**

Виконала:

**Студентка групи ПМІ-13
Демко Сніжана Іванівна**

Львів - 2024

Мета: ознайомитися з поняттям черги та черги з пріоритетом, їхнім функціоналом та реалізацією в програмному середовищі. Перевірити роботу функцій за допомогою написаних тестів.

Черга - динамічна структура даних для тимчасового їх зберігання, яка має дві точку доступу (голову і хвіст) і функціонує за принципом FIFO (першим прийшов - першим пішов).

Інтерфейс черги:

- push (вставляє елемент в хвіст);
- pop (видає елемент з голови);
- isEmpty (перевіряє чергу на наявність в ній елементів);
- getSize (повертає довжину черги - кількість елементів у ній);
- front (повертає значення первого елемента черги);
- back (повертає значення останнього елемента черги).

Черга з пріоритетами — це структура даних, що призначена для обслуговування множини елементів, кожний з яких додатково має "пріоритет", пов'язаний з ним.

Інтерфейс черги з пріоритетом (відмінності з чергою):

- push (вставляє елемент (пару "значення-пріоритет") у потрібне місце за пріоритетом);
- pop (видає елемент з найбільшим пріоритетом).

Виконання:

1. Реалізація черги:

- Спочатку була реалізована структура даних "черга" з використанням класу Queue. Це дозволило створити структуру, яка працює за принципом "перший прийшов - перший вийшов".
- Я додала методи для додавання (push), видалення (pop) елементів, перевірки на порожність (isEmpty) та отримання розміру черги (getSize), повернення значення первого елемента (front) та останнього (back).

2. Реалізація черги з пріоритетом:

- Після цього я розширила свою реалізацію, створивши чергу з пріоритетом з використанням класу PriorityQueue. Ця структура дозволяє зберігати елементи у визначеному порядку в залежності

від їхнього пріоритету.

- Для цього я змінила реалізацію функції push, дозволяючи їй працювати з пріоритетами елементів. Метод push тепер додає новий елемент у чергу, враховуючи його пріоритет та правильно розміщає його у черзі.

3. Тести:

- Для перевірки коректності роботи функцій було створено Google Tests.
- Тести перевіряли різні аспекти роботи черги та черги з пріоритетом: додавання елементів, видалення елементів, отримання значень елементів, перевірка порожності та розміру черги.

Результати роботи програмної реалізації черги та тестування:

```
Queue: 1 2 3 4
The queue is not empty
Back element: 4
Front element:1
Size:4
Queue after pop:2 3 4
The queue is not empty
Back element: 4
Front element:2
Size:3
[=====] Running 6 tests from 1 test case.
[-----] Global test environment set-up.
[-----] 6 tests from QueueTest
[RUN    ] QueueTest.PushTest
[OK     ] QueueTest.PushTest (0 ms)
[RUN    ] QueueTest.PopTest
[OK     ] QueueTest.PopTest (10 ms)
[RUN    ] QueueTest.FrontTest
[OK     ] QueueTest.FrontTest (0 ms)
[RUN    ] QueueTest.BackTest
[OK     ] QueueTest.BackTest (1 ms)
[RUN    ] QueueTest.SizeType
[OK     ] QueueTest.SizeType (0 ms)
[RUN    ] QueueTest.EmptyTest
[OK     ] QueueTest.EmptyTest (0 ms)
[-----] 6 tests from QueueTest (38 ms total)

[-----] Global test environment tear-down
[=====] 6 tests from 1 test case ran. (51 ms total)
[ PASSED ] 6 tests.
```

Результати роботи програмної реалізації черг з пріоритетом та тестування:

```
Queue elements without priority: 1 2 3 4 5 6
Priority Queue: 2 4 5 3 1
The queue is not empty
Back element: 1
Front element:2
Size:5
Priority Queue after pop:4 5 3 1
The queue is not empty
Back element: 1
Front element:4
Size:4
[=====] Running 6 tests from 1 test case.
[-----] Global test environment set-up.
[-----] 6 tests from PriorityQueueTest
[RUN    ] PriorityQueueTest.Push
[OK     ] PriorityQueueTest.Push (2 ms)
[RUN    ] PriorityQueueTest.Pop
[OK     ] PriorityQueueTest.Pop (0 ms)
[RUN    ] PriorityQueueTest.IsEmptyQueue
[OK     ] PriorityQueueTest.IsEmptyQueue (1 ms)
[RUN    ] PriorityQueueTest.GetSizeQueue
[OK     ] PriorityQueueTest.GetSizeQueue (0 ms)
[RUN    ] PriorityQueueTest.Front
[OK     ] PriorityQueueTest.Front (0 ms)
[RUN    ] PriorityQueueTest.Back
[OK     ] PriorityQueueTest.Back (0 ms)
[-----] 6 tests from PriorityQueueTest (531 ms total)

[-----] Global test environment tear-down
[=====] 6 tests from 1 test case ran. (839 ms total)
[PASSED ] 6 tests.
```

Висновок: завдяки виконанню лабораторної роботи я отримала краще розуміння роботи з чергами та чергами з пріоритетом. Реалізація обох структур дозволила мені навчитися робити різні операції з даними, такі як додавання, видалення та роботу з пріоритетами. Тести, які були написані, допомогли перевірити коректність роботи моїх класів.