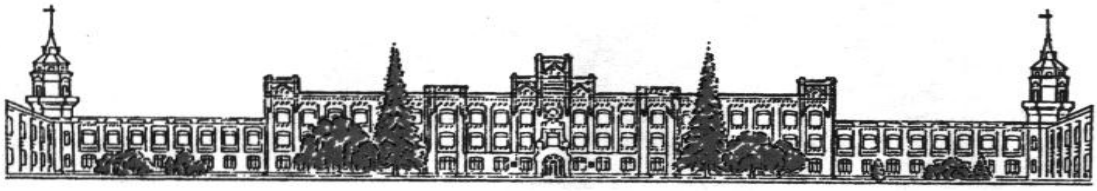

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра системного проектування

Алгоритми та структури даних

Лабораторна робота №3

“Дослідження структур даних пріоритетна черга та купа”

Київ – 2021

1 Мета роботи

Ознайомитись і дослідити структуру даних бінарна купа. Набути навичок реалізації абстрактної структури даних черга з пріоритетом на основі бінарної купи мовою програмування C++. Ознайомитись з механізмом перевантаження операторів та дослідити особливості сортування купою (пірамідального сортування). Порівняти власну реалізацію пріоритетної черги з готовим бібліотечним рішенням STL.

2 Короткі теоретичні відомості

- Перевантаження операторів в C++: http://www.amse.ru/courses/cpp2/2010_11_22.html
- Бінарне дерево: https://uk.wikipedia.org/wiki/Двійкове_дерево
- Бінарна купа: https://neerc.ifmo.ru/wiki/index.php?title=Двоичная_куча https://habr.com/ru/post/112222/
- Пріоритетна черга: https://uk.wikipedia.org/wiki/Бінарна_купа https://acm.khpnets.info/wiki/Очередь_с_приоритетами
- Сортування купою (пірамідальне сортування): https://neerc.ifmo.ru/wiki/index.php?title=Сортировка_кучей

3 Завдання

Обрати одну із запропонованих задач та реалізувати пріоритетну чергу для ефективнного вирішення цієї задачі.

1. **Створити структуру для зберігання об’єктів з характеристиками відповідно до обраної задачі:**
   - Обрати певний власний критерій для визначення більш пріоритетного об’єкту на основі його характеристик
   - Перевантажити оператор “менше” (operator<) для порівняння двох об’єктів за критерієм

- Перевантажити оператор “дорівнює” (operator==) для визначення рівності двох об’єктів за цим самим критерієм

2. Реалізувати пріоритетну чергу на основі незростаючої бінарної купи:

2.1 Створити структуру PriorityQueue, яка буде містити в собі бінарне дерево для роботи бінарної купи. Для зберігання бінарного дерева можна використати звичайний динамічний масив, що вже був реалізований у минулих роботах.

2.2 Реалізувати основні методи переміщення по бінарному дереву:

- getParent(index) знайти індекс батьківського вузла
- getLeftChild(index) знайти індекс лівого дочірнього вузла
- getRightChild(index) знайти індекс правого дочірнього вузла

2.3 Реалізувати внутрішні методи для підтримки властивості незростаючої (max-heap) купи:

- siftUp(index) просіяти елемент вверх по дереву
- siftDown(index) просіяти елемент вниз по дереву

2.4 Реалізувати основні методи для роботи пріоритетної черги:

- push(object) додати новий елемент в чергу
- top() отримати верхній елемент з черги
- pop() видалити верхній елемент з черги
- size() знайти кількість елементів в черзі
- empty() перевірити чергу на пустоту

3. Провести тестування, використавши вказану нижче функцію testPriorityQueue(). Перевірити правильність та швидкість роботи, порівнявши з готовим бібліотечним рішенням STL priority_queue.

4. Продумати реалізацію сортування купою (пірамідального сортування) на базі бінарної купи, вміти обгрунтовано пояснити та показувати в коді.

Додаткові завдання:

- Реалізувати сортування купою, порівняти його з готовим бібліотечним рішенням STL std::sort() або з іншими алгоритмами сортувань, реалізованими в минулих роботах, – провести тестування з замірами часу на різних вхідних даних, зробити аналіз та висновки.

Код для тестування пріоритетної черги:

```
#include <queue>
#include <cstdlib>
#include <ctime>
#include <iostream>

using namespace std;

template <typename T>
float testPriorityQueueSpeed(T&& priorityQueue)
{
    const int iters = 100000;

    clock_t timeStart = clock();
    for (int i = 0; i < iters; i++)
    {
        int insertDataAmount = rand() % 6 + 5;
        for (int j = 0; j < insertDataAmount; j++)
        {
            priorityQueue.push(Data());
        }

        priorityQueue.top();
        priorityQueue.pop();
    }
    clock_t timeEnd = clock();
    float time = (float(timeEnd - timeStart)) / CLOCKS_PER_SEC;

    return time;
}

bool testPriorityQueue()
{
    srand(time(NULL));

    const int iters = 20000;

    PriorityQueue myPriorQueue;
    priority_queue<Data> stlPriorQueue;

    bool isDataEqual = true;
    for (int i = 0; i < iters; i++)
    {
        int insertDataAmount = rand() % 6 + 5;
        for (int j = 0; j < insertDataAmount; j++)
        {
            Data randData = Data();
            myPriorQueue.push(randData);
            stlPriorQueue.push(randData);
        }
```

```
        if (!(myPriorQueue.top() == stlPriorQueue.top()))
        {
            isDataEqual = false;
            cerr << "Comparing failed on iteration " << i << endl << endl;
            break;
        }

        int removeDataAmount = rand() % insertDataAmount;
        for (int j = 0; j < removeDataAmount; j++)
        {
            myPriorQueue.pop();
            stlPriorQueue.pop();
        }
    }

    int myQueueSize = myPriorQueue.size();
    int stlQueueSize = stlPriorQueue.size();

    float stlTime =
testPriorityQueueSpeed<priority_queue<Data>>(priority_queue<Data>());
    float myTime = testPriorityQueueSpeed<PriorityQueue>(PriorityQueue());

    cout << "My PriorityQueue:" << endl;
    cout << "Time: " << myTime << ", size: " << myQueueSize << endl;
    cout << "STL priority_queue:" << endl;
    cout << "Time: " << stlTime << ", size: " << stlQueueSize << endl << endl;

    if (isDataEqual && myQueueSize == stlQueueSize)
    {
        cout << "The lab is completed" << endl << endl;
        return true;
    }

    cerr << ":(" << endl << endl;
    return false;
}
```

4 Зміст звіту

Звіт має містити:

1) Титульний аркуш
2) Мету роботи
3) Варіант завдання
4) Хід виконання роботи:
    a) Умова задачі
    b) Скріншоти результатів виконання
    c) Лістинг програми (код)
5) Висновки

5 Контрольні питання

1) Навіщо потрібна структура даних пріоритетна черга? Чим вона відрізняється від звичайної черги?
2) Які структури даних із вже відомих можна було б застосувати для вирішення поставлених задач, у чому їх недоліки?
3) Що таке бінарне дерево? Намалюйте його, опишіть на ньому особливості бінарної купи.
4) Поясніть процес побудови купи та просіювання елементів у незростаючій (max-heap) / неспадній (min-heap) бінарній купі.
5) Опишіть логіку роботи пірамідального сортування, його асимтотичну складність, переваги та недоліки.

6 Варіанти завдань

Задача 1

Впродовж тижня студент отримує багато домашніх завдань. Кожне завдання характеризується цікавістю, рівнем корисності предмета, важкістю виконання, дедлайном тощо. Студент за певним критерієм повинен обрати найбільш оптимальне завдання та почати його виконувати, після цього перейти до наступного завдання. Потрібно врахувати, що впродовж тижня у студента можуть з’являтися нові завдання з більш високим пріоритетом.

Задача 2

Перед ботом з’являються вороги в зоні видимості. Кожен ворог характеризується рівнем здоров’я, можливим уроном, скілом гравця тощо. Бот за певним критерієм повинен обрати найбільш оптимальну для себе ціль та знищити її, після цього перейти до наступної цілі. Потрібно врахувати, що впродовж гри можуть з’являтися нові вороги з більш високим пріоритетом.

Задача 3

У лікарню надходять пацієнти у важкому стані. Кожен пацієнт характеризується ступенем важкості, шансом на можливе врятування життя, рівнем дотримання карантину до погіршення свого стану тощо. Лікар повинен обрати одного пацієнта та почати його реанімувати, після цього перейти до наступного пацієнта. Потрібно врахувати, що під час лікування в клініку можуть надходити нові пацієнти з більш високим пріоритетом.

Власна задача

Самостійно придумати задачу в якій буде оптимально використати пріоритетну чергу.