# Постановка задачи

Создать родительский класс «Очередь» с функциями инициализации очереди, добавления элемента в очередь и извлечения элемента из очереди. Создать метод создания копии очереди. Результатом должен стать новый экземпляр класса

«Очередь», состоящий из элементов (копий элементов) исходно очереди. Порядок следования элементов должен быть сохранен. Создать функцию слияния двух очередей. Результатом должна быть очередь, состоящая из элементов первой и второй очереди. Порядок следования элементов должен быть сохранен. На основе родительского класса «Очередь» создать дочерний класс «Очередь1» с функциями нахождения и отображения на экране требуемого в соответствии с вариантом задания значения.

Вариант 9 – Нахождение последнего элемента, меньшего среднего гармонического

# Формализация задачи

Данная программа разбита на 9 файлов: 4 головных файла (header.h, header1.h, header2.h, header3.h), 4 реализационных файла (source.cpp, private.cpp, protected.cpp, public.cpp) и один главный (main.cpp) – это файлы с объявлением, определением методов класса соответственно, а также файл с управляющей функцией.

Элемент очереди реализованный с помощью структуры под именем Node, состоит из данных целочисленного типа, а также указателя на предыдущий такой же элемент previous.

Элементы очереди вводится вручную с клавиатуры (их количество заранее не известно). Вследствие этого реализовано динамическое выделение памяти для очереди. Наряду с этим реализовано динамическое удаление; в базовом классе прописаны конструкторы и деструкторы.

В программе реализованы два пользовательских меню – выбор модификатора доступа производного класса (очереди), с которым хочет работать пользователь и выбор действий для работы с очередью: 1 – Добавить элемент в очередь; 2 – Извлечение элемента очереди; 3 – Вывод очереди на экран; 4 – Подсчет среднего арифметического; 5 – Создание копии очереди; 6 – Слияние двух очередей; 7 – Выход.

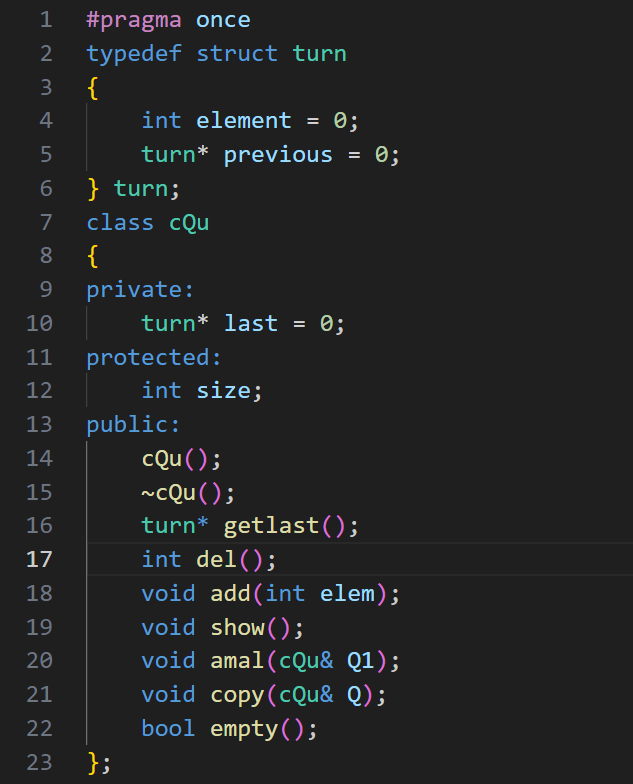
В методе для реализации задачи выделяются 4 переменных: указатель на тип данных last (равный последнему элементу очереди), целочисленный count, result, h, hn. Сначала проверяем на нуль переменную size. Если она равна нулю, то очередь пуста, иначе работаем. Пока указатель не равен нулевой строке, count увеличивается на один, а к hn прибавляется 1/значение элемента. После выхода из цикла мы находим среднее гармоническое: h = count/hn.

При работе с одним из трёх классов (публичный, защищенный, приватный), то объявляем и определяем функцию, принимающую класс Qu ничего не возвращающая.

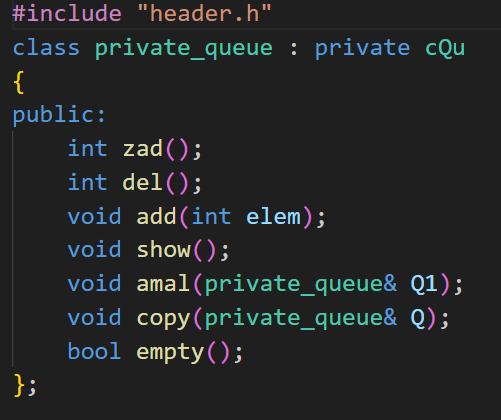
Когда используем различные модификаторы доступа в наследных классах, меняется видимость данных и методов базового класса для производных классов. Ради этой демонстрации в базовом классе прописаны все три спецификатора доступа: в частном доступе находится переменная элемента очереди, в защищенном доступе – размер очереди, в публичном – геттер элемента, конструктор, деструктор, прочие методы. Когда мы используем публичный класс-наследник, то все публичные методы и данные базового класса также становятся публичными и для наследника; все защищенные данные и методы – защищенными в наследнике; приватные данные и методы напрямую недоступны из базового класса. Когда мы используем защищенный класс-наследник, то все публичные и защищенные данные и методы базового класса становятся здесь защищенными; приватные данные напрямую недоступны. Когда мы используем приватный класс-наследник, то все данные и методы базового касса становятся приватными; приватные данные базового класса напрямую недоступны.

# Исходный код

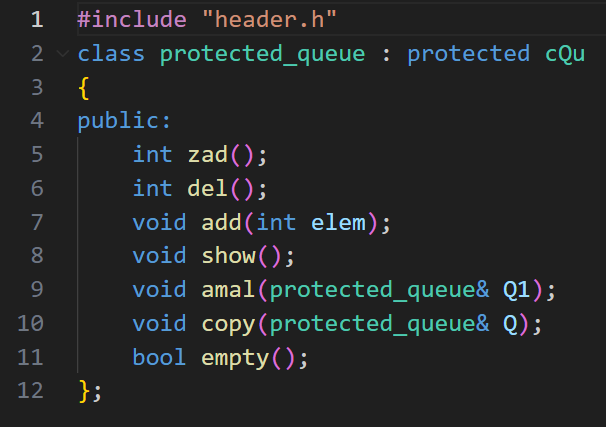
Файл header.h



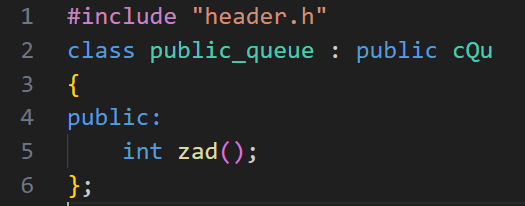
Файл header1.h



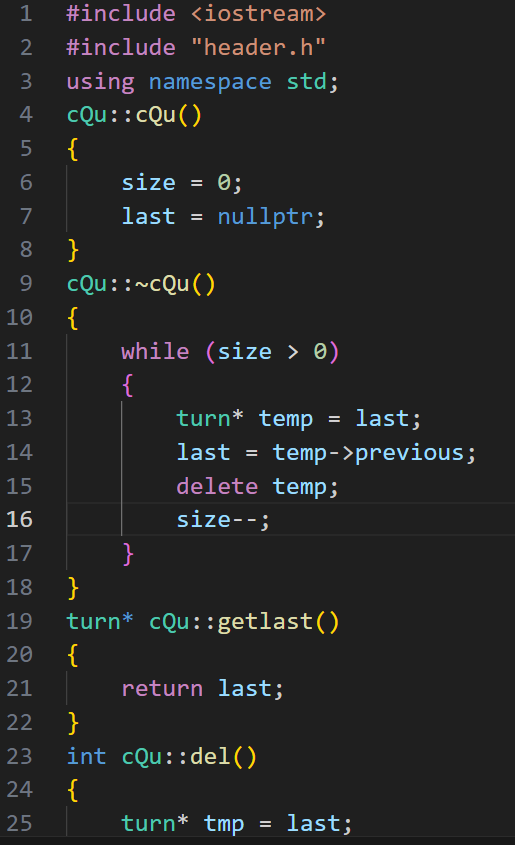
Файл header2.h

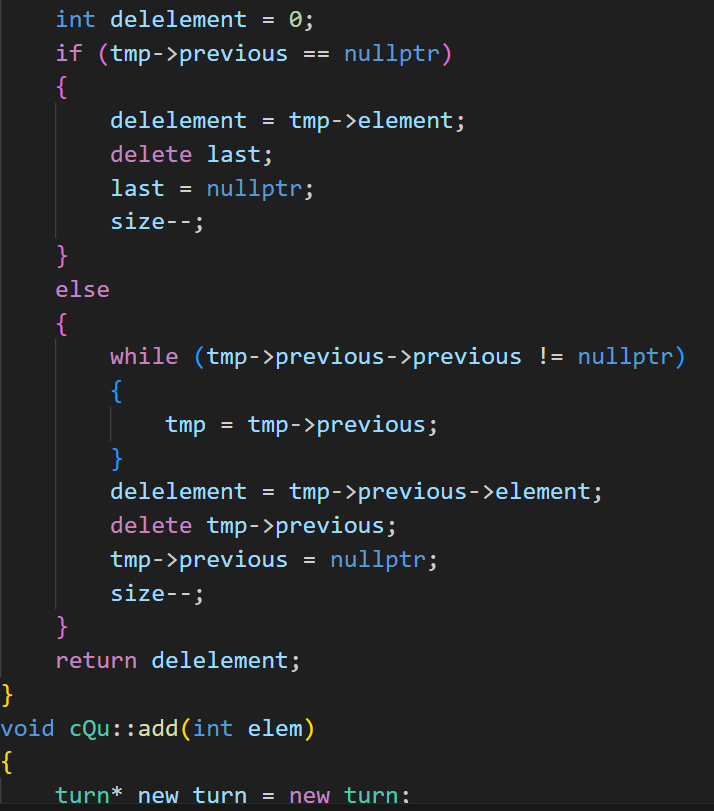


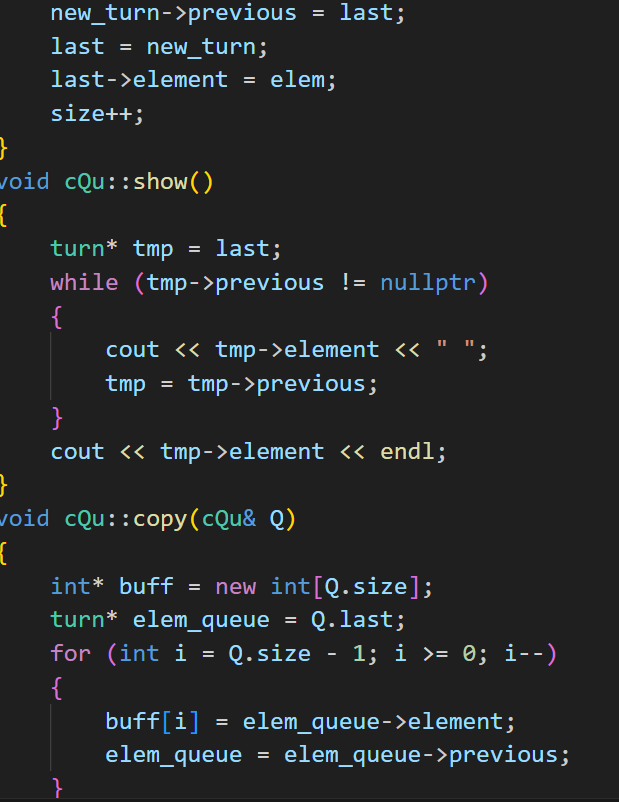
Файл header3.h

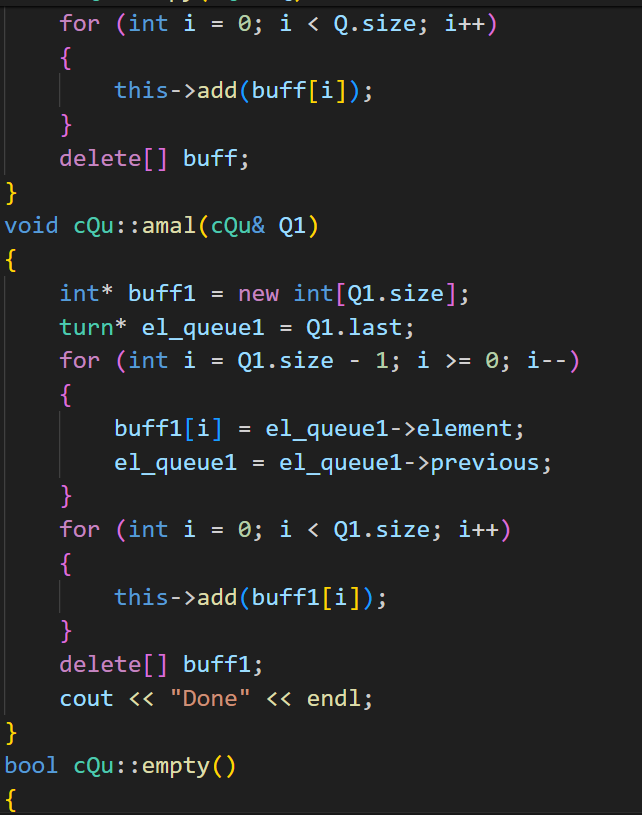


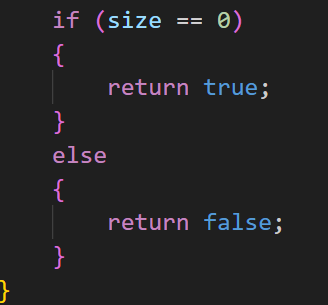
Файл source.cpp



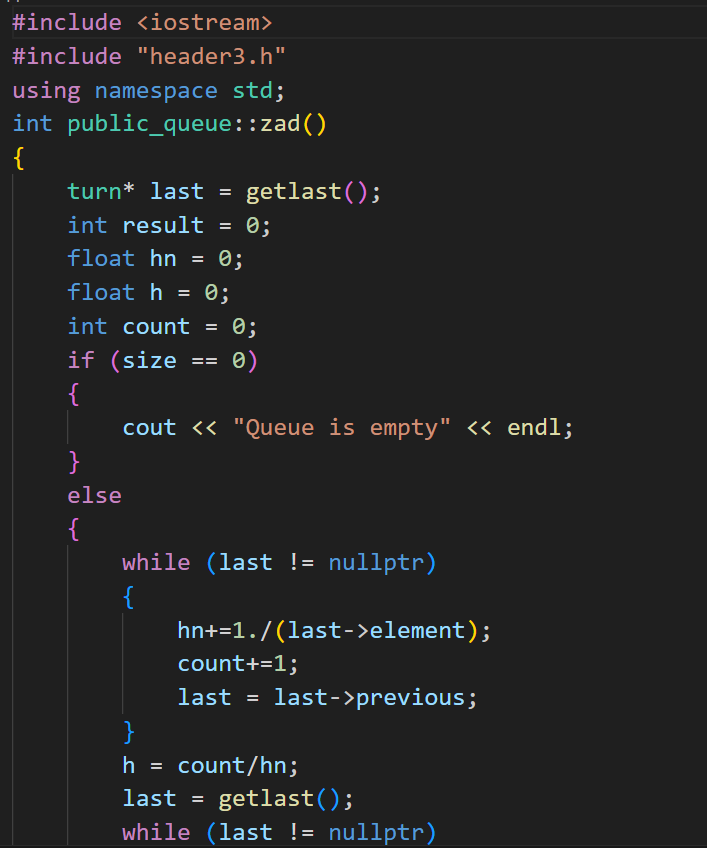


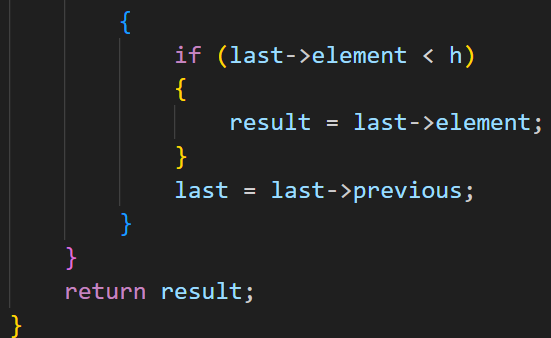




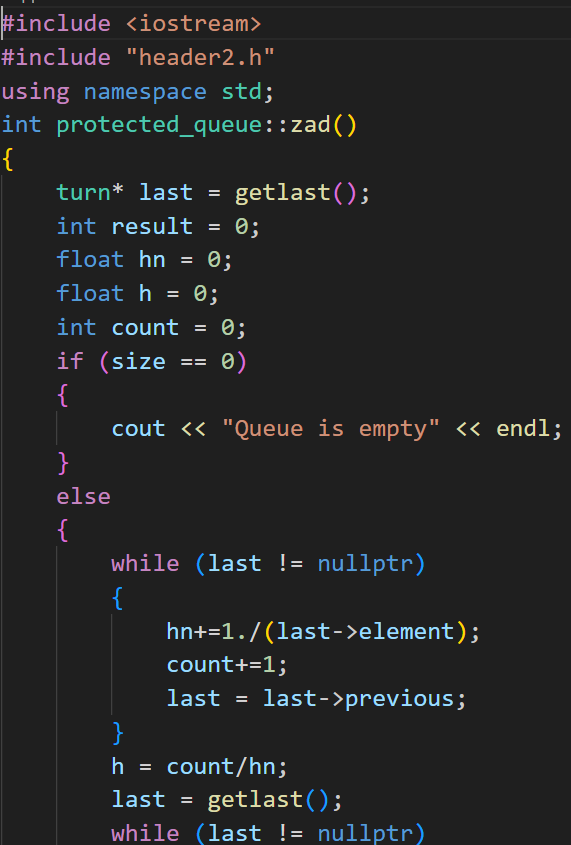


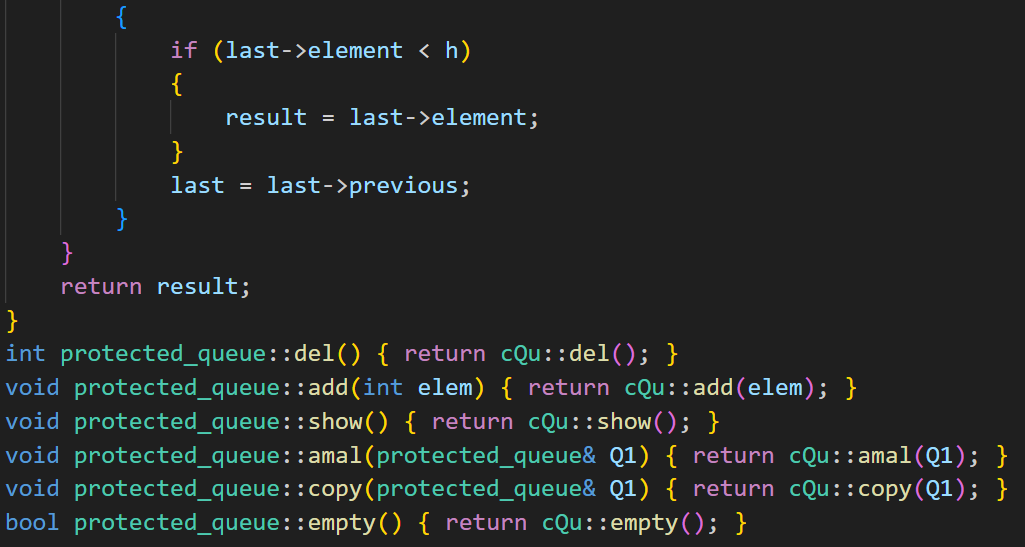
Файл public.cpp



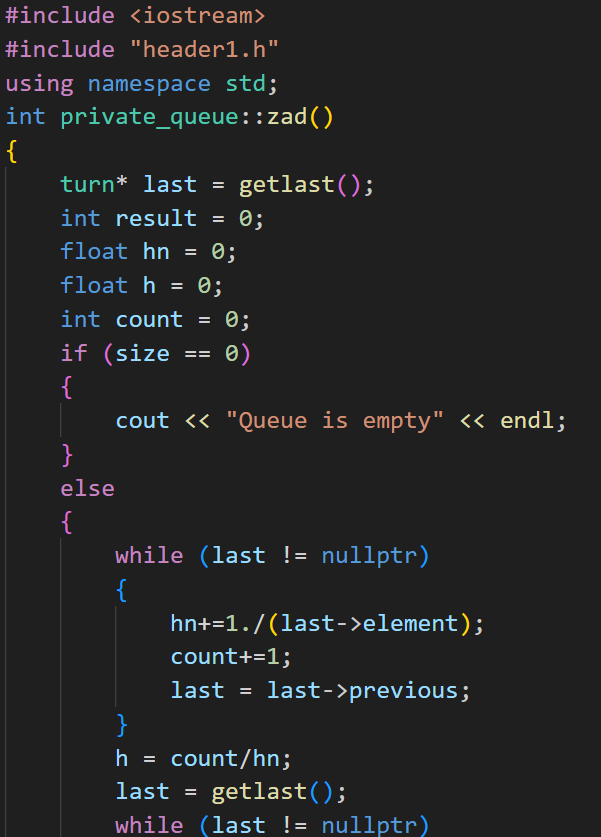


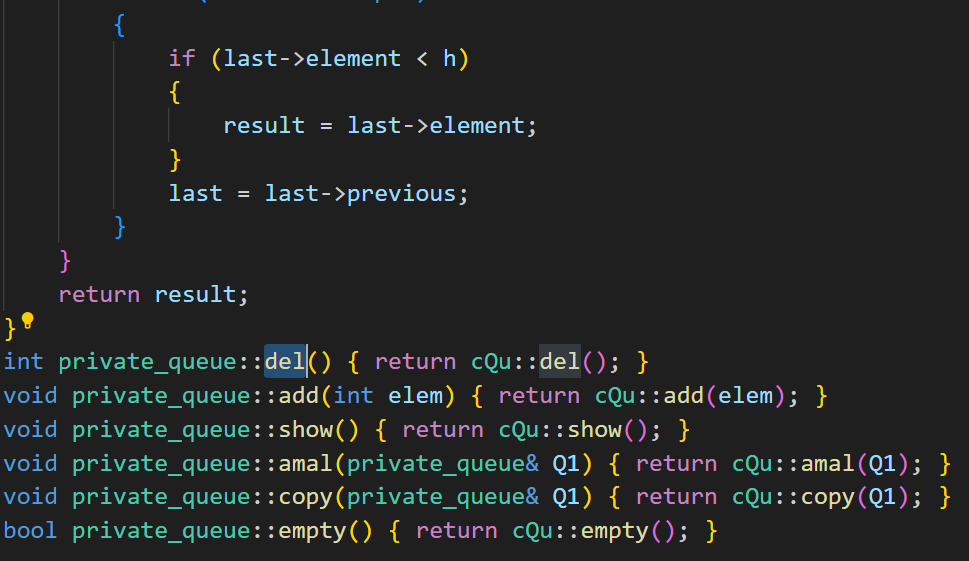
Файл protected.cpp





Файл private.cpp





Файл main.cpp

#include <iostream>

#include "header.h"

#include "header1.h"

#include "header2.h"

#include "header3.h"

using namespace std;

template <class T>

void Qu(T\* q, int n\_q)

{

    int key;

    int flag = 1;

    int value;

    int nomer = 0;

    int total = 0;

    int number;

    int copiya = 0;

    while (flag == 1)

    {

        cout << "1 - Add element" << endl;

        cout << "2 - Delete element" << endl;

        cout << "3 - Show queue" << endl;

        cout << "4 - MinimumofH" << endl;

        cout << "5 - Copy queue" << endl;

        cout << "6 - Amalganation and show" << endl;

        cout << "7 - Exit" << endl;

        cout << "Choose: ";

        cin >> key;

        switch (key)

        {

            case 1:

                cout << "Enter value: ";

                cin >> value;

                q[nomer].add(value);

                cout << "Value added.\n " << endl;

                break;

            case 2:

                if (q[nomer].empty())

                {

                        cout << "Queue is empty.\n " << endl;

                        system("pause");

                        break;

                }

                else

                {

                        value = q[nomer].del();

                        cout << "Delete element: " << value << endl;

                        cout << "\n";

                }

                break;

            case 3:

                if (q[nomer].empty())

                {

                    cout << "Queue is empty.\n" << endl;

                    system("pause");

                    break;

                }

                else

                {

                    q[nomer].show();

                    cout << "\n";

                }

                break;

            case 4:

                if (q[nomer].empty())

                {

                    cout << "Queue is empty.\n" << endl;

                    system("pause");

                    break;

                }

                else

                {

                    total = q[nomer].zad();

                    cout << "Minimal element: " << total << endl;

                    cout << "\n";

                }

                break;

            case 5:

                if (q[nomer].empty())

                {

                    cout << "Queue is empty.\n" << endl;

                    system("pause");

                    break;

                }

                else

                {

                    cout << "The number of queue is: ";

                    cin >> copiya;

                    if (1 == n\_q || copiya == nomer)

                    {

                        cout << "Amount of queues is only 1.\n" << endl;

                        system("pause");

                        break;

                    }

                    q[copiya].copy(q[nomer]);

                    cout << "Copied.\n" << endl;

                }

                break;

            case 6:

                if (n\_q == 1)

                {

                    cout << "Amount of queues is only 1.\n" << endl;

                    system("pause");

                    break;

                }

                else

                {

                    cout << "Choose queue: ";

                    cin >> number;

                    if ((number < 0) || (number >= n\_q) || (number == nomer))

                    {

                        cout << "Incorrect value.\n" << endl;

                        system("pause");

                        break;

                    }

                    if (q[number].empty())

                    {

                        cout << "The second queue is empty.\n" << endl;

                        system("pause");

                        break;

                    }

                    q[nomer].amal(q[number]);

                    cout << "\n";

                }

                break;

            case 7:

                exit(1);

                break;

            default:

                cout << "Error" << endl;

                break;

        }

    }

}

int main()

{

    int amount\_queues = 0; //количество очередей

    int key;

    int flag = 1;

    private\_queue\* q1 = NULL;

    protected\_queue\* q2 = NULL;

    public\_queue\* q3 = NULL;

    cout << "Enter amount of queues: ";

    do

    {

        cin >> amount\_queues;

    } while (amount\_queues <= 0 && cout << "Incorrect value\n" && cout << "Enter amount of queues: ");

    cout << "Choose access mode: " << endl;

    while (flag == 1)

    {

        cout << "1 - Private" << endl;

        cout << "2 - Protected" << endl;

        cout << "3 - Publiuc" << endl;

        cout << "0 - Exit" << endl;

        cout << "Choose: ";

        cin >> key;

        switch (key)

        {

        case 1:

            q1 = new private\_queue[amount\_queues];

            Qu(q1, amount\_queues);

            delete[] q1;

            break;

        case 2:

            q2 = new protected\_queue[amount\_queues];

            Qu(q2, amount\_queues);

            delete[] q2;

            break;

        case 3:

            q3 = new public\_queue[amount\_queues];

            Qu(q3, amount\_queues);

            delete[] q3;

            break;

        case 0:

            flag = 0;

            break;

        default:

            cout << "Error" << endl;

            break;

        }

    }

    return 0;

}

1. **Результаты работы программы**

При запуске программы, нам предлагают выбрать модификатор доступа, с которым мы будем работать. Выберем публичный (Рис. 1)

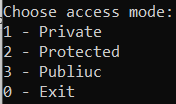


Рис 1 – Выбор режима

Для работоспособности программы введем последовательно очередь: 9 2 4 5. (Рис. 2).

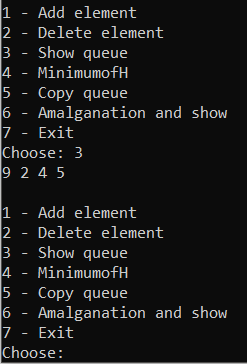


Рис 2 –Вывод очереди

Найдем среднее арифметическое, согласно поставленному условию в задании работы (Рис. 3)

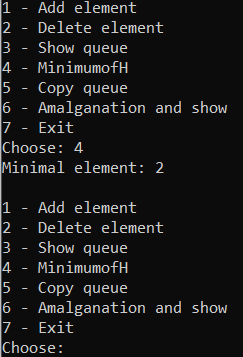


Рис 3 –Вывод последнего элемента, меньшего среднего гармонического

Произведём слияние нашей очереди и её копии (Рис. 4).

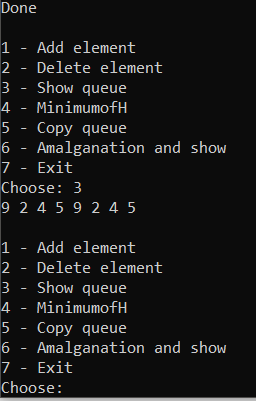


Рис 4 – Слияние с копией и вывод на экран

Удалим из полученной очереди элемент (Рис. 5)

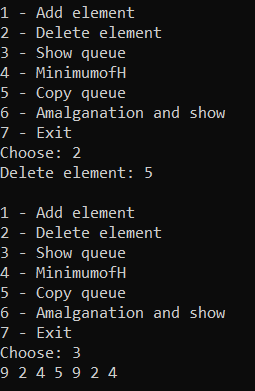


Рис 5 – Удаление элемента и вывод на экран.

# Выводы

В процессе выполнения лабораторной работы мы изучили различные модификаторы доступа при наследовании классов от одного базового, и то, как эти модификаторы влияют на работу с этими классами-наследниками. При выполнении тестов программа доказала свою правильную работоспособность.