Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра экономической информатики

Дисциплина: Объектно-ориентированное программирование

Индивидуальная практическая работа №11

Студент: гр. 724402 Чернявский Я.А.

Руководитель: Сторожев Д.А.

Минск 2019

**Задание:** Напишите шаблон класса для работы с очередью. Определите несколько очередей разных типов и поработайте с их данными.

Листинг кода:

Main.cpp

#include <iostream>

using namespace std;

#include "queue.h" // подключаем шаблон класса

int main()

{

setlocale(LC\_ALL, "RUSSIAN");

Queue<char> myQueue(14); // объект класса очередь

myQueue.printQueue(); // вывод очереди

int ct = 1;

char ch;

// добавление элементов в очередь

while (ct++ < 14)

{

cin >> ch;

myQueue.enqueue(ch);

}

myQueue.printQueue(); // вывод очереди

// удаление элемента из очереди

myQueue.dequeue();

myQueue.dequeue();

myQueue.dequeue();

myQueue.printQueue(); // вывод очереди

cout << "\n\nСработал конструктор копирования:\n";

Queue<char> newQueue(myQueue);

newQueue.printQueue(); // вывод очереди

return 0;

}

Queue.h

#include <cassert>

template<typename T>

class Queue

{

private:

T\* queuePtr; // указатель на очередь

const int size; // максимальное количество элементов в очереди

int begin, // начало очереди

end; // конец очереди

int elemCT; // счетчик элементов

public:

Queue(int = 10); // конструктор по умолчанию

Queue(const Queue<T>&); // конструктор копирования

~Queue(); // деструктор

void enqueue(const T&); // добавить элемент в очередь

T dequeue(); // удалить элемент из очереди

void printQueue();

};

// реализация методов шаблона класса Queue

// конструктор по умолчанию

template<typename T>

Queue<T>::Queue(int sizeQueue) :

size(sizeQueue), // инициализация константы

begin(0), end(0), elemCT(0)

{

// дополнительная позиция поможет нам различать конец и начало очереди

queuePtr = new T[size + 1];

}

// конструктор копии

template<typename T>

Queue<T>::Queue(const Queue& otherQueue) :

size(otherQueue.size), begin(otherQueue.begin),

end(otherQueue.end), elemCT(otherQueue.elemCT),

queuePtr(new T[size + 1])

{

for (int ix = 0; ix < size; ix++)

queuePtr[ix] = otherQueue.queuePtr[ix]; // копируем очередь

}

// деструктор класса Queue

template<typename T>

Queue<T>::~Queue()

{

delete[] queuePtr;

}

// функция добавления элемента в очередь

template<typename T>

void Queue<T>::enqueue(const T& newElem)

{

// проверяем, ести ли свободное место в очереди

assert(elemCT < size);

// очередь начинает заполняться с 0 индекса

queuePtr[end++] = newElem;

elemCT++;

// проверка кругового заполнения очереди

if (end > size)

end -= size + 1; // возвращаем end на начало очереди

}

// функция удаления элемента из очереди

template<typename T>

T Queue<T>::dequeue()

{

// проверяем, есть ли в очереди элементы

assert(elemCT > 0);

T returnValue = queuePtr[begin++];

elemCT--;

// проверка кругового заполнения очереди

if (begin > size)

begin -= size + 1; // возвращаем begin на начало очереди

return returnValue;

}

template<typename T>

void Queue<T>::printQueue()

{

cout << "Очередь: ";

if (end == 0 && begin == 0)

cout << " пустая\n";

else

{

for (int ix = end; ix >= begin; ix--)

cout << queuePtr[ix] << " ";

cout << endl;

}

}

Пример выполнения работы:

