МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра прикладной математики, механики, управления и программного обеспечения**

**О Т Ч Е Т**

по лабораторной работе №5 по дисциплине “Структуры и алгоритмы компьютерной обработки данных”

|  |
| --- |
| Выполнил студент гр. Б8204  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. А. Федоренко |
| Проверил к.т.н. доцент  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Н. Остроухова |

г. Владивосток

2018

**Неформальная постановка задачи**

Разработать шаблонный класс кольцевой очереди.

**Спецификация класса**

template <class T>

class Queue

{

public:

Queue(int n = 10); // Конструктор

Queue(const Queue&); // Конструктор копирования

int pushElement (T); // Функция добавления элемента в очередь

T popElement (); // Функция удаления чтения элемента из очереди

int length (); // Функция, определения длины очереди

void printAll(); // Функция печати массива array

int print(); // Функция печати очереди

Queue& operator = ( Queue& q); // Перегрузка оператора присваивания

~Queue(); // Деструктор

private:

T\* array = nullptr;

int SIZE; // Переменная, хранящая длину массива

H = 0; // Переменная, указывающая на начало очереди

T = -1; // Переменная, указывающая на последний добавленный элемент

};

**Спецификация функций**

int **pushElement** (T **newElement**) - Функция добавления элемента в очередь.

На вход принимает элемент **newElement** и возвращает значение, которое находится в ячейке массива с индексом **H**:

* 1, если очередь полная
* 0, если добавление элемента **newElement** прошло успешно

T **popElement** () - Функция чтения элемента из очереди.

На вход ничего не принимает и возвращает значение, которое находится в ячейке массива **array** с индексом **H**

int **length** () – Функция определения длины очереди.

На вход ничего не принимает и возвращает целочисленное значение равное длине очереди

void **printAll**() - Функция печати массива **array** на экран.

На вход ничего не принимает и ничего не возвращает.

void **print**() - Функция печати очереди на экран.

На вход ничего не принимает и ничего не возвращает.

**Тесты**

int **pushElement** (T **newElement**)**:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ситуация** | **Входные данные** | **Очередь до** | **Очередь после** | **Значение функции** |
| Добавление элемента **newElement** в пустую очередь | 5 | 0 0 0 0 0 0 0 0 0 0  H = 0, T = -1 | 5 0 0 0 0 0 0 0 0 0 | 0 |
| Добавление элемента **newElement** в очередь, когда в конце массива есть свободное место | 5 | 10 4 3 6 8 2 0 0 0 0  H = 0, T = 5 | 10 4 3 6 8 2 5 0 0 0  H = 0, T = 6 | 0 |
| Добавление элемента **newElement** в очередь, когда в конце массива нет свободное место | 5 | 0 0 0 0 10 4 3 6 8 2  H = 4, T = 9 | 5 0 0 0 10 4 3 6 8 2  H = 4, T = 0 | 0 |
| Добавление элемента **newElement** в полную очередь | 5 | 10 4 3 6 8 2 5 9 7 1  H = 5, T = 4 | 10 4 3 6 8 2 5 9 7 1  H = 5, T = 4 | 1 |

T **popElement** ()**:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ситуация** | **Очередь до** | **Очередь после** | **Значение функции** |
| Чтение элемента из пустой очереди | 0 0 0 0 0 0 0 0 0 0  H = 0, T = -1 | 0 0 0 0 0 0 0 0 0 0 | 0 |
| Чтение элемента из последней ячейки массива | 10 4 3 6 8 2 0 0 0 5  H = 9, T = 5 | 10 4 3 6 8 2 0 0 0 5  H = 0, T = 5 | 5 |
| Чтение элемента, когда очередь не заполнена полностью | 0 0 0 0 10 4 3 6 8 2  H = 4, T = 9 | 0 0 0 0 10 4 3 6 8 2  H = 5, T = 9 | 10 |
| Чтение элемента из полной очереди | 10 4 3 6 8 2 5 9 7 1  H = 5, T = 4 | 10 4 3 6 8 2 5 9 7 1  H = 6, T = 4 | 2 |
| Чтение последнего элемента очереди | 0 0 0 0 0 4 0 0 0 0  H = 5, T = 5 | 0 0 0 0 0 4 0 0 0 0  H = 0, T = -1 | 4 |
| Чтение предпоследнего элемента очереди | 0 0 0 0 5 4 0 0 0 0  H = 4, T = 5 | 0 0 0 0 5 4 0 0 0 0  H = 5, T = 5 | 5 |

int **length** ()**:**

|  |  |  |
| --- | --- | --- |
| **Ситуация** | **Очередь** | **Значение функции** |
| Пустая очередь | 0 0 0 0 0 0 0 0 0 0  H = 0, T = -1 | 0 |
| Полная очередь | 10 4 3 6 8 2 5 9 7 1  H = 0, T = 9 | 10 |
| Очередь заполнена не полностью, T находится справа от H | 0 0 0 0 10 4 3 6 8 2  H = 4, T = 9 | 6 |
| Очередь заполнена не полностью, T находится слева от H | 5 0 0 0 10 4 3 6 8 2  H = 4, T = 0 | 7 |

**Queue& operator = (Queue& q):**

|  |  |  |
| --- | --- | --- |
| **Ситуация** | **Очереди до** | **Очереди после** |
| Присваивание меньшей очереди **q1** к большей **q**, когда в **q** достаточно места и конец очереди справа от головы | **q**: 1 2 3 0 0 0 0 0 0 0  **q1**: 1 2 3 4 5 | **q:** 1 2 3 1 2 3 4 5 0 0  **q1**: 1 2 3 4 5 |
| Присваивание меньшей очереди **q1** к большей **q**, когда в **q** недостаточно места и конец очереди справа от головы | **q**: 1 2 3 4 5 6 7 0 0 0  **q1**: 1 2 3 4 5 | **q**: 1 2 3 4 5 6 7 1 2 3  **q1**: 1 2 3 4 5 |
| Присваивание меньшей очереди **q1** к большей **q**, когда в **q** нет места | **q**: 1 2 3 4 5 6 7 8 9 10  **q1**: 1 2 3 4 5 | **q**: 1 2 3 4 5 6 7 8 9 10  **q1**: 1 2 3 4 5 |
| Присваивание меньшей очереди **q1** к большей **q**, когда в **q** достаточно места и конец очереди слева от головы | **q**: 1 2 3 0 0 0 0 0 0 10  **q1**: 1 2 3 4 5 | **q**: 1 2 3 1 2 3 4 5 0 10  **q1**: 1 2 3 4 5 |
| Присваивание меньшей очереди **q1** к большей **q**, когда в **q** недостаточно места и конец очереди справа от головы | **q**: 1 2 3 0 0 0 7 8 9 10  **q1**: 1 2 3 4 5 | **q**: 1 2 3 1 2 3 7 8 9 10  **q1**: 1 2 3 4 5 |
| Множественное присваивание очередей  **q, q1, q2** (**q = q1 = q2**) | **q**: 1 2 3 0 0 0 0 0 0 0  **q1**: 1 2 3 0 0 0 0 0 0 0  **q2**: 1 2 3 4 5 | **q**: 1 2 3 1 2 3 1 2 3 4  **q1**: 1 2 3 1 2 3 4 5 0 0  **q2**: 1 2 3 4 5 |