

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования  
**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК  
  
Кафедра прикладной математики, механики, управления и программного обеспечения**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6 ПО ДИСЦИПЛИНЕ  
«СТРУКТУРЫ И АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ»**

Направление 02.03.03 «Математическое обеспечение и администрирование  
информационных систем»

* Выполнил студент гр. Б8204  
  Проскурин Денис Александрович  
  Проверил:  
  Доцент, к.т.н С.Н.Остроухова

Владивосток 2019

**Неформальная постановка задачи**

Реализовать класс циклическая очередь с приоритетным выходом. Хвост указывает на последний добавленный элемент. Класс должен содержать следующие методы:

1. Конструктор
2. Деструктор
3. Добавить элемент
4. Удалить элемент
5. Неразрушающее чтение
6. Длина(текущее кол-во)
7. Печать (от left до right)
8. Перегрузка присваивания
9. Конструктор копирования

**Спецификация методов класса**

class Circle\_queue\_exit {

private:

struct element \*vector;//вектор записей вида статус, данные

unsigned int max\_l;//максимальная длина

unsigned int right, left;//правая, левая границы

void move\_left(const unsigned int pos);//сдвинуть вектор влево

void move\_right(const unsigned int pos); //сдвинуть вектор вправо

public:

Circle\_queue\_exit(unsigned int size);//конструктор

Circle\_queue\_exit(const Circle\_queue\_exit &obj);//конструктор копирования

~Circle\_queue\_exit();//дестуктор

unsigned int add(element);//добавить элемент в очередь

unsigned int del();//удалить приоритетный элемент

element read(unsigned int deep);//неразрушающее чтение

unsigned int length();//текущая длина

unsigned int print();//печать всей очереди

void operator =(Circle\_queue\_exit&);//перегрузка оператора =

}

**Circle\_queue\_exit(unsigned int size)** — конструктор,создающий очередь максимальным размером size.

**unsigned int add(element)** — добавить элемент в конец очереди. Возвращает 0 если операция прошла успешно. Возвращает 1 если очередь переполнена.

**unsigned int del()** - удаление элемента с наивысшим приоритетом. Возвращает 0 , если операция прошла успешно. Возвращает 2, если очередь пуста.

**element read() —** возвращает элемент c самым большим приоритетом. Возвращает элемент empty, если очередь пуста

**unsigned int length()**-возвращает текущий размер очереди.

**unsigned int print()**-выводит на экран всю очередь от левой до правой границы.

**void operator =(Circle\_queue\_exit&)**-левому операнду присваивает данные правого операнда. Если левый операнд меньше правого, присваивает возможное количество элементов.

**Тесты**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **Тестирование конструктора класса** | |
| Circle\_queue\_exit example(5);  element rec;  rec.data = "A";  cout << "Length=" << example.length() << endl;  for (int i = 0; i < 5; i++) {  rec.priority = 5 - i;  cout << "code=" << example.add(rec) << endl;  rec.data[0]++;  } | 5|A  4|B  3|C  2|D  1|E |
| Circle\_queue\_exit example(0);  element rec;  rec.data = "A";  cout << "Length=" << example.length() << endl;  for (int i = 0; i < 10; i++) {  rec.priority = 10 - i;  cout << "code=" << example.add(rec) << endl;  rec.data[0]++;  } | 10|A  9|B  8|C  7|D  6|E  5|F  4|G  3|H  2|I  1|J |
| **Тестирование конструктора-копирования класса** | |
| Circle\_queue\_exit example(5);  4|A  2|B  3|C  1|D  4|E  Circle\_queue\_exit example2=example;  example2.print(); | 4|A  2|B  3|C  1|D  4|E |
| Circle\_queue\_exit example(5); //пустая  Circle\_queue\_exit example2=example; | пусто |
| **Тестирование метода add** | |
| Circle\_queue\_exit example(5);  element rec;  rec.data = "A";  cout << "Length=" << example.length() << endl;  for (int i = 0; i < 5; i++) {  rec.priority = i;  cout << "code=" << example.add(rec) << endl;  rec.data[0]++;  } | 0|A  1|B  2|C  3|D  4|E  коды возврата 0 0 0 0 0 |
| Circle\_queue\_exit example(5);  element rec;  rec.data = "A";  cout << "Length=" << example.length() << endl;  for (int i = 0; i < 10; i++) {  rec.priority = i;  cout << "code=" << example.add(rec) << endl;  rec.data[0]++;  } | 0|A  1|B  2|C  3|D  4|E  коды возврата 0 0 0 0 0 1 1 1 1 1 |
| **Тестирование метода del** | |
| 0|A  1|B  2|C  3|D  4|E  example.del(); | 0|A  1|B  2|C  3|D  4|E |
| 0|A  1|B  2|C  3|D  4|E  for (int i = 0; i < 5; i++) {  example.del();  } | Коды возврата:0 0 0 0 0 |
| 0|A  1|B  2|C  3|D  4|E  for (int i = 0; i < 10; i++) {  example.del();  } | Коды возврата: 0 0 0 0 0 2 2 2 2 2 |
| 4|A  2|B  3|C  1|D  4|E  example.del() | 2|B  3|C  1|D  4|E  код возврата 0 |
| 2|B  3|C  1|D  4|E  example.del() | 2|B  3|C  1|D  код возврата 0 |
| 2|B  3|C  1|D  example.del() | 2|B  1|D  код возврата 0 |
| 2|B  1|D  example.del() | 1|D  код возврата 0 |
| **Тестирование метода read** | |
| Пустая очередь | priority=0 data=empty |
| 4|A  7|B  8|C  6|D  4|E  6|F  7|G  3|H  10|I  2|J  cout << example.read().priority << "|" << example.read().data << endl; | 10|I |
| **Тестирование метода length** | |
| Circle\_queue\_exit example(5);  cout << "Length=" << example.length() << endl; | 0 |
| Circle\_queue\_exit example(5);  element rec;  rec.data = "A";  for (int i = 0; i < 10; i++) {  rec.priority = i;  cout << "code=" << example.add(rec) << endl;  rec.data[0]++;  }  cout << "Length=" << example.length() << endl; | 5 |
| **Тестирование метода print** | |
| Circle\_queue\_exit example(5); | Код возврата 2 |
| Circle\_queue\_exit example(5);  element rec;  rec.data = "A";  cout << "Length=" << example.length() << endl;  for (int i = 0; i < 5; i++) {  rec.priority = 5-i;  cout << "code=" << example.add(rec) << endl;  rec.data[0]++;  }  cout << "Length=" << example.length() << endl;  example.print(); | 5|A  4|B  3|C  2|D  1|E  код возврата 0 |
| **Тестирование метода =** | |
| Circle\_queue\_exit example(5);  4|A  2|B  3|C  1|D  4|E  Circle\_queue\_exit example2(10);  10|F  9|G  8|H  7|I  6|J  5|K  4|L  3|M  2|N  1|O  example=example2; | Example имеет вид:  10|F  9|G  8|H  7|I  6|J |
| Circle\_queue\_exit example(5);  4|A  2|B  3|C  1|D  4|E  Circle\_queue\_exit example2(10);  10|F  9|G  8|H  7|I  6|J  5|K  4|L  3|M  2|N  1|O  example2=example; | Example2 имеет вид:  4|A  2|B  3|C  1|D  4|E |
| Circle\_queue\_exit example(5);  4|A  2|B  3|C  1|D  4|E  Circle\_queue\_exit example2(10)- пусто  example=example2; | Код возврата 2 |