

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ 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 “ХЕШ-ТАБЛИЦЫ” ПО ДИСЦИПЛИНЕ  
СТРУКТУРЫ И АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ**

Направление 02.03.03 «Математическое обеспечение и администрирование  
информационных систем»

Выполнил студент гр. Б8119-02.03.03техпро  
Хмелевский Е.Д.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Проверил: к.т.н., доцент кафедры ПММУ и ПО Остроухова С.Н \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Владивосток  
2021

## Описание

Цель лабораторной работы: разработать собственный класс **Queue** для работы с не кольцевой очередью с хвостом(индексом) на первую свободную ячейку со следующим необходимым пакетом методов:

1. Конструктор
2. Конструктор копирования
3. Перегруженное присваивание очередей
4. Деструктор
5. Добавление
6. Удаление (с вытаскиванием элемента и без)
7. Длина очереди
8. Очистка очереди (удаление всех элементов)
9. Печать очереди

## Спецификация

КЛАСС ОЧЕРЕДЬ – **Queue**

ПОЛЯ

Класс **Queue** состоит из следующих полей:

* \***vector** – вектор;
* **max\_size** – размер вектора;
* **head** – индекс первого элемента;
* **tail** – индекс последнего элемента;

МЕТОДЫ

1. **explicit Queue (unsigned int size)** – инициализирует **this->max\_size** с помощью **max\_size**, head и **tail** нулём и создаёт массив **vector** размера **max\_size**.
2. **Queue (const Queue& obj)** – конструктор копирования, копирующий объект класса **Queue**, поступающий на вход.
3. **~Queue()** – очищает память выделенную для **vector** память.
4. **Queue& operator= (const Queue& obj)** – перегруженное присваивание очередей. Присваивает объекту класса **this** (текущему) объект класса **obj**, путём присваивания отдельных ячеек.
5. **bool add (int a)** – добавление элемента **a** в вектор **vector**. При успешном добавлении увеличивает значение **tail** на 1 и возвращает значение **true**, иначе выводит в консоль *"Очередь заполнена"* и **false**.
6. **int del (bool check)** – если на вход поступил аргумент **check** со значением **true**, значение **tail** уменьшается на 1, если изначально **tail** не равен 0, и функция возвращает 1. Если же на вход поступил аргумент со значением **false**, функция возвращает индекс элемента и тоже уменьшает **tail** на единицу. Если же удалить элемент с извлечением или без не получилось, функция возвращает значение -1.
7. **[[nodiscard]] unsigned int length() const** – возвращает длину очереди.
8. **bool clear()** –удаляет все элементы очереди и присваивает полю **tail** значение 0. Если же функция будет пытаться очистить пустую очередь, в консоль выведется сообщение *"Очередь уже пуста"* и вернётся значение **false**.
9. **friend ostream &operator<<(ostream &out, Queue q)** – перегрузка вывода класса **Queue**. Выводит размер очереди **max\_size** с последующим переносом строки. Далее выводятся все элементы очереди. Если элементов очереди не было добавлено выводится сообщение *"Очередь пуста!"*.

## Тесты

**explicit Queue (unsigned int size) – обычный конструктор**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Размер очереди не равен нулю. | **Queue q; Размер: 10** | **return: q** {  *this->max\_size = size; // 10*  *head = 0;*  *tail = 0;*  *vector = new int[size]; // вектор //размера 10*  } |
| Количество записей равно нулю. | **Queue q; Размер: 0** | **return: q** {  *this->max\_size = size; // 0*  *head = 0;*  *tail = 0;*  *vector = new int[size]; // вектор //размера 0*  } |

**Queue (const Queue& obj) – конструктор копирования**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Копирование не пустого объекта класса Queue  (**Queue q1 = Queue(q)**) | **Queue q; q: Размер: 10; \*vector = {1, 2, 3, 4, 4}** | **Queue q1; q1: Размер: 10; \*vector = {1, 2, 3, 4, 4}** |
| Копирование пустого объекта класса Queue  (**Queue q1 = Queue(q)**) | **Queue q; q: Размер: 10; \*vector = {}** | **Queue q1; q1: Размер: 10; \*vector = {}** |

**Queue &operator=(const Queue &obj) – перегруженное присваивание очередей**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Присваивание двух очередей одного размера | **Queue q; q: Размер: 10; \*vector = {1, 2, 3, 4, 4} Queue q1; q1: Размер: 10; \*vector = {4, 4, 3, 2, 1} Присваивание: q = q1;** | **q: Размер: 10; \*vector = {4, 4, 3, 2, 1}** |
| Присваивание двух очередей разного размера и разной заполненности (первая заполнена больше, элементы не обрезаются) | **Queue q; q: Размер: 10; \*vector = {1, 2, 3, 4, 4, 5, 5} Queue q1; q1: Размер: 8; \*vector = {4, 4, 3, 2, 1} Присваивание: q = q1;** | **q: Размер: 8; \*vector = {4, 4, 3, 2, 1, 5, 5}** |
| Присваивание двух очередей разного размера и разной заполненности (первая заполнена меньше, элементы не обрезаются) | **Queue q; q: Размер: 10; \*vector = {1, 2, 3, 4, 4} Queue q1; q1: Размер: 8; \*vector = {4, 4, 3, 2, 1, 0, 0} Присваивание: q = q1;** | **q: Размер: 8; \*vector = {4, 4, 3, 2, 1, 0, 0}** |
| Присваивание двух очередей разного размера и разной заполненности  (элементы обрезаются) | **Queue q; q: Размер: 10; \*vector = {1, 2, 3, 4, 4, 5, 5, 6, 6, 6} Queue q1; q1: Размер: 8; \*vector = {4, 4, 3, 2, 1, 0, 0} Присваивание: q = q1;** | **q: Размер: 8; \*vector = {4, 4, 3, 2, 1, 0, 0, 6}** |
| Присваивание трёх очередей разного размера и разной заполненности | **Queue q; q: Размер: 5; \*vector = {1, 2, 3} Queue q1; q1: Размер: 8; \*vector = {0, 0, 1, 3, 2, 0, 0} Queue q2; q2: Размер: 3; \*vector = {4, 4, 3} Присваивание: q1 = q = q2;** | **q: Размер: 3; \*vector = {4, 4, 3} q1: Размер: 3; \*vector = {4, 4, 3} q2: Размер: 3; \*vector = {4, 4, 3}** |
| Присваивание трёх очередей разного размера и разной заполненности. Таких, что некоторые очереди дополняются своими элементами | **Queue q; q: Размер: 5; \*vector = {1, 2, 3} Queue q1; q1: Размер: 8; \*vector = {0, 0, 3, 2} Queue q1; q2: Размер: 6; \*vector = {4, 4, 3, -1, -2, -3} Присваивание: q1 = q = q2;** | **q: Размер: 5; \*vector = {1, 2, 3} q1: Размер: 5; \*vector = {1, 2, 3, 2} q2: Размер: 5; \*vector = {1, 2, 3, 2, -2}** |

**bool add (int a) – добавление**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Добавление элемента в пустую очередь | **Queue q; q: Размер: 10; \*vector = {} a = 1** | **Return:** 1 **q:** **\*vector = {1}** |
| Добавление элемента в не пустую и не полностью заполненную очередь | **Queue q; q: Размер: 10; \*vector = {1, 2, 3, 4, 4} a = 10** | **q: Размер: 10; \*vector = {1, 2, 3, 4, 4, 10}** |
| Добавление элемента в заполненную очередь | **Queue q; q: Размер: 10; \*vector = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} a = 10** | **Return:** 0 **q: Размер: 10; \*vector = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}** |

**int del(bool check) – удаление**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Удаление элемента из пустой очереди с вытаскиванием | **Queue q; q: Размер: 10; \*vector = {}** | **return**: -1 **q:** **\*vector = {}** |
| Удаление элемента из не пустой очереди с вытаскиванием | **Queue q; q: Размер: 10; \*vector = {1, 2, 3, 4, 4}** | **return**: 1 **q:** **\*vector = {1, 2, 3, 4}** |
| Удаление элемента из пустой очереди без вытаскивания | **Queue q; q: Размер: 10; \*vector = {}** | **return**: -1 **q:** **\*vector = {}** |
| Удаление элемента из не пустой очереди без вытаскивания | **Queue q; q: Размер: 10; \*vector = {1, 2, 3, 4, 4}** | **return**: 4 **q:** **\*vector = {1, 2, 3, 4, 4}** |

**[[nodiscard]] unsigned int length() const – длина**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Очередь нулевого размера | **Queue q; q: Размер: 0; \*vector = {}** | **return**: 0 |
| Очередь не нулевого размера | **Queue q; q: Размер: 7; \*vector = {1, 2, 3, 4, 4}** | **return**: 7 |

**bool clear() – удаление всех элементов**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Очистка в пустой очереди | **Queue q; q: Размер: 7; \*vector = {}** | **return**: false **Console:** Очередь уже пуста |
| Очистка в заполненной очереди | **Queue q; q: Размер: 7; \*vector = {1, 2, 3, 4, 4}** | **return**: true **q: Размер: 7; \*vector = {}** |

**friend ostream &operator<<(ostream &out, Queue q) – печать**

|  |  |  |
| --- | --- | --- |
| **Тестовая ситуация** | **Входные данные** | **Выходные данные** |
| Пустая очередь | **Queue q; q: Размер: 10; \*vector = {}** | **Console:** размер очереди: 10 Очередь пуста! --------------------- |
| Не пустая очередь, количество элементов меньше размера | **Queue q; q: Размер: 10; \*vector = {5, 5, 5, 5, 5, 5, 5}** | **Console:** размер очереди: 8 5 5 5 5 5 5 5 --------------------- |
| Не пустая очередь, количество элементов равно размеру | **Queue q; q: Размер: 10; \*vector = {5, 5, 5, 5, 5, 5, 5}** | **Console:** размер очереди: 8 5 5 5 5 5 5 5 5 5 5 --------------------- |