МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА «ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АВТОМАТИЗИРОВАННЫХ СИСТЕМ»

**ОТЧЁТ**

**«ЛАБОРАТОРНАЯ №11.1: ОЧЕРЕДЬ»**

Дисциплина: «Программирование»

Выполнил:

Студент группы ИВТ-21-2б

Безух Владимир Сергеевич

Проверил:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь, 2022

Содержание

[1. Постановка задачи 3](#_Toc98237010)

[2. Анализ задачи 4](#_Toc98237011)

[3. Исходный код 5](#_Toc98237012)

[4. Анализ результатов 8](#_Toc98237013)

# Постановка задачи

Продемонстрировать работу очереди как структуры данных.

# 

# Анализ задачи

Очередь можно реализовать на основе двусвязного списка с ограниченным функционалом. Необходимо реализовать методы добавления узла в конец очереди и удаления узла из начала очереди для организации принципа FIFO (first in, first out).

# 

# Исходный код

#include<string>

#include<iostream>

template <typename T>

class Queue {

public:

Queue();

~Queue();

size\_t size() const;

T back() const;

void push(const T& data);

T front() const;

void pop();

void clear();

private:

struct Node {

Node(T data = T(), Node\* pointer\_to\_prev\_node = nullptr, Node\* pointer\_to\_next\_node = nullptr)

: data(data), pointer\_to\_prev\_node(pointer\_to\_prev\_node), pointer\_to\_next\_node(pointer\_to\_next\_node) {}

Node(const Node& copy)

: data(copy.data), pointer\_to\_prev\_node(copy.pointer\_to\_prev\_node), pointer\_to\_next\_node(copy.pointer\_to\_next\_node) {}

Node& operator=(const Node& right) {

if (this != &right) {

data = right.data;

pointer\_to\_prev\_node = right.pointer\_to\_prev\_node;

pointer\_to\_next\_node = right.pointer\_to\_next\_node;

}

return \*this;

}

T data;

Node\* pointer\_to\_prev\_node;

Node\* pointer\_to\_next\_node;

};

void pushFirstNode(Node\* node);

void pushBackNode(Node\* node);

void popFirstNode();

void popFrontNode();

size\_t queue\_size;

Node\* head\_node;

Node\* tail\_node;

};

template<typename T>

Queue<T>::Queue()

: queue\_size(size\_t{0}), head\_node(nullptr), tail\_node(nullptr) {}

template<typename T>

Queue<T>::~Queue()

{

clear();

}

template<typename T>

size\_t Queue<T>::size() const

{

return queue\_size;

}

template<typename T>

T Queue<T>::back() const

{

return tail\_node->data;

}

template<typename T>

void Queue<T>::push(const T& data)

{

Node\* new\_node = new Node(data);

queue\_size ? pushBackNode(new\_node) : pushFirstNode(new\_node);

++queue\_size;

}

template<typename T>

T Queue<T>::front() const

{

return head\_node->data;

}

template<typename T>

void Queue<T>::pop()

{

if (queue\_size == size\_t{0}) return;

Node\* remove\_node = head\_node;

(queue\_size == size\_t{1}) ? popFirstNode() : popFrontNode();

delete remove\_node;

--queue\_size;

}

template<typename T>

void Queue<T>::clear()

{

if (queue\_size == size\_t{0}) return;

Node\* remove;

Node\* next\_node = head\_node;

while (queue\_size) {

remove = next\_node;

next\_node = next\_node->pointer\_to\_next\_node;

delete remove;

--queue\_size;

}

head\_node = nullptr;

tail\_node = nullptr;

}

template<typename T>

void Queue<T>::pushFirstNode(Node\* node)

{

head\_node = node;

tail\_node = node;

}

template<typename T>

void Queue<T>::pushBackNode(Node\* node)

{

tail\_node->pointer\_to\_next\_node = node;

node->pointer\_to\_prev\_node = tail\_node;

tail\_node = node;

}

template<typename T>

void Queue<T>::popFirstNode()

{

head\_node = nullptr;

tail\_node = nullptr;

}

template<typename T>

void Queue<T>::popFrontNode()

{

head\_node->pointer\_to\_next\_node->pointer\_to\_prev\_node = nullptr;

head\_node = head\_node->pointer\_to\_next\_node;

}

template <typename T>

void print(const Queue<T>& queue)

{

std::cout << queue.front() << " <- начальный элемент | конечный элемент -> " << queue.back() << '\n';

}

int main()

{

Queue<std::string> queue;

std::setlocale(LC\_ALL, "Russian");

queue.push("str1"); print(queue);

queue.push("str3"); print(queue);

queue.push("str5"); print(queue);

queue.push("str7"); print(queue);

queue.pop(); print(queue);

queue.pop(); print(queue);

queue.pop(); print(queue);

queue.pop();

}

# Анализ результатов

Результаты работы программы (рис. 1).

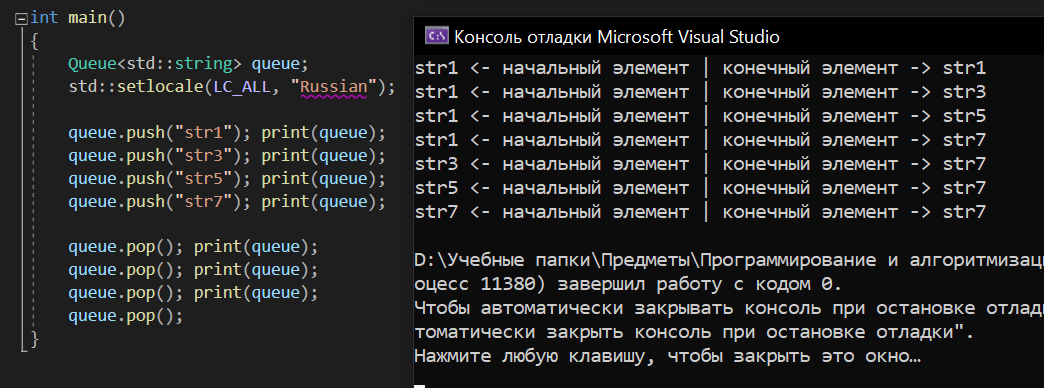


Рисунок 1 — Результаты