Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: «Очередь»

Выполнил студент ИВТ-22-2б:

Афонин Артем Александрович

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

Записи в линейном списке содержат ключевое поле типа string (строка символов). Сформировать очередь. Удалить элемент с заданным ключом. Добавить К элементов перед элементом с заданным номером.

**Алгоритм решения**

1. Определить узел очереди через структуру struct Node. Узел содержит ключевое слово типа string и указатель на следующий узел.

2. Определить структуру стека struct Line. Структура содержит указатели на начало и конец очереди, конструктор для указателей.

3. Реализовать функцию void push() для добавления элемента в конец очереди. В функции создаётся и заполняется узел очереди, а также обновляется конец очереди.

4. Реализовать функцию для удаления элемента из очереди void pop(). В функции проводится проверка на пустоту очереди, удаляется узел очереди и обновляется её конец.

5. Реализовать функцию void printLine(), которая выводит элементы стека.

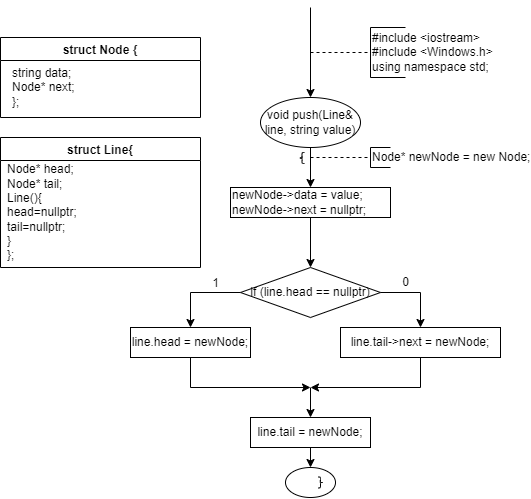
6. Реализовать функцию void delete\_key(), с помощью которой происходит удаление заданного ключа и сохраняются элементы изначальной очереди с помощью промежуточных очередей.

7. Реализовать функцию void delete\_elem() для удаления элемента перед элементом с заданным номером.

8. Реализовать функцию void copy\_line1(), с помощью которой происходит копирование элемента из изначальной очереди в промежуточную очередь для выполнения первой части задания.

9. Реализовать функцию void copy\_line2(), с помощью которой происходит копирование элемента из изначальной очереди в промежуточную очередь для выполнения второй части задания.

10. В главной функции int main() реализовать вызов всех внешних функций согласно постановке задачи.

**Блок-схемы**

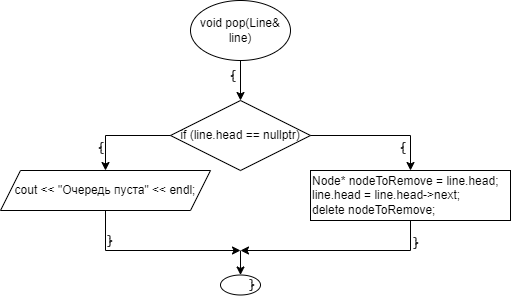
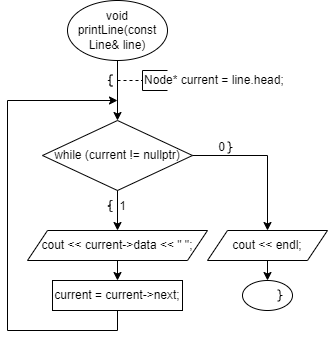
Рисунок 1 – Структуры struct Node и struct Line, функция void push()

Рисунок 2 – Функция void pop()

Рисунок 3 – Функция void printLine()

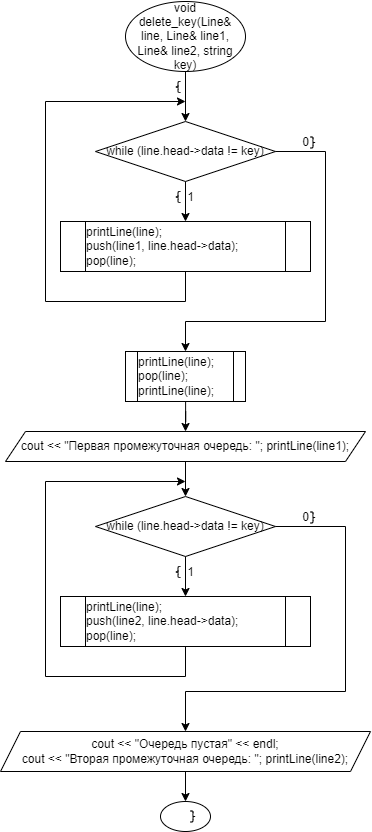


Рисунок 4 – Функция void delete\_key()

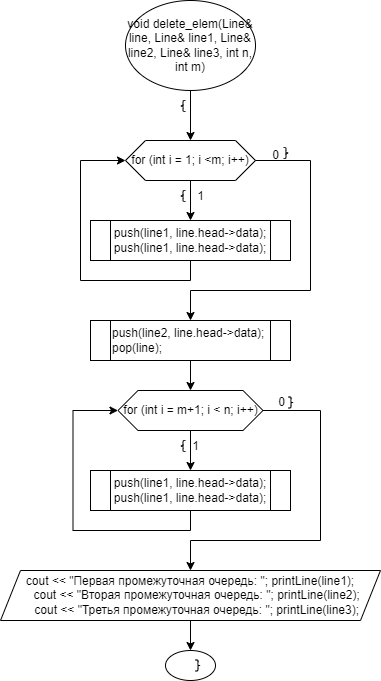


Рисунок 5 – Функция void delete\_elem()

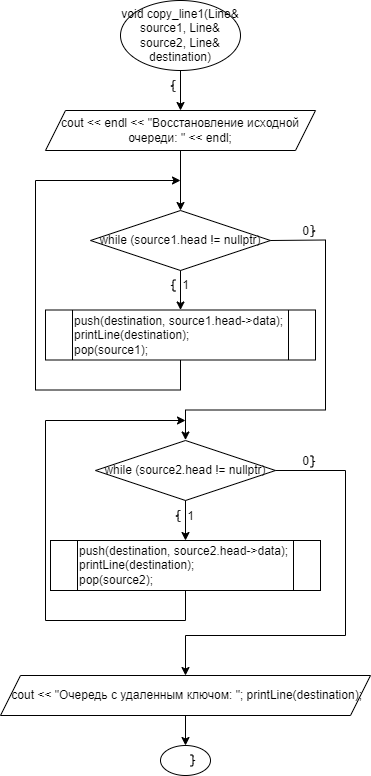


Рисунок 6 – Функция void copy\_line1()

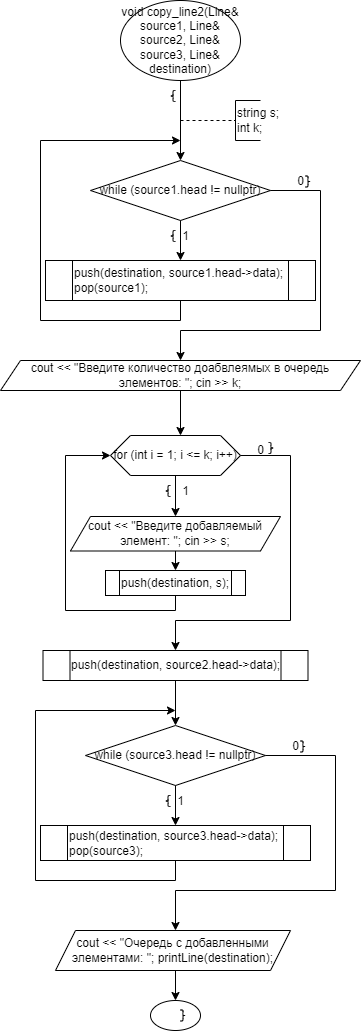


Рисунок 7 – Функция copy\_line2()

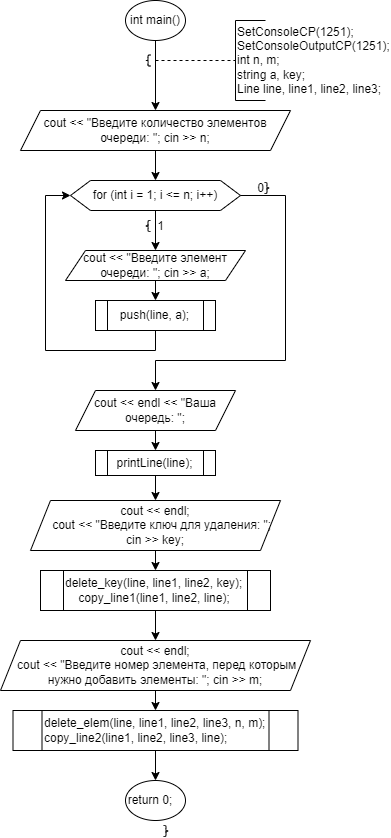


Рисунок 8 – Функция int main()

**Код программы**

#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

//Определение узла очереди

struct Node {

string data; // Данные одного узла

Node\* next; // Указатель на следующий узел

};

//Определение структуры очереди

struct Line {

Node\* head; // Указатель на начало очереди

Node\* tail; // Указатель на конец очереди

Line() {

head = nullptr;

tail = nullptr;

}

};

// Функция добавления элемента в конец очереди

void push(Line& line, string value) {

Node\* newNode = new Node; // Создание нового узла очереди

newNode->data = value;

newNode->next = nullptr;

if (line.head == nullptr)

line.head = newNode; // Если очередь была пуста, обновление указателя на начало

else

line.tail->next = newNode; // Иначе обновление указатель на следующий элемент у последнего элемента

line.tail = newNode; // Обновление указателя на конец очереди

}

// Функция удаления элемента из начала очереди

void pop(Line& line) {

if (line.head == nullptr) {

cout << "Очередь пуста" << endl; // Очередь пуста, невозможно удалить элемент

}

else {

Node\* nodeToRemove = line.head; // Сохранение указателя на удаляемый узел

line.head = line.head->next; // Обновление указателя на начало очереди

delete nodeToRemove; // Удаление узла списка

}

}

// Функция вывода содержимого очереди

void printLine(const Line& line) {

Node\* current = line.head; // Создание узла для вывода

while (current != nullptr) {

cout << current->data << " "; // Вывод значения узла

current = current->next; // Переход к следующему узлу

}

cout << endl;

}

// Функция для удаления ключа

void delete\_key(Line& line, Line& line1, Line& line2, string key) {

while (line.head->data != key) { // Пока не будет достигнут ключ

printLine(line);

push(line1, line.head->data); // Добавление в промежуточную очередь элементов до ключа

pop(line); // Удаление из изначальной очереди элементов до ключа

}

printLine(line);

pop(line); // Удаление ключа из изначальной очереди

printLine(line);

cout << "Первая промежуточная очередь: "; printLine(line1);

while (line.head != nullptr) {

printLine(line);

push(line2, line.head->data); // Добавление в промежуточную очередь элементов после ключа

pop(line); // Удаление из изначальной очереди элементов после ключа

}

cout << "Очередь пустая" << endl;

cout << "Вторая промежуточная очередь: "; printLine(line2);

}

// Функция для разбиение на промежуточные очереди

void delete\_elem(Line& line, Line& line1, Line& line2, Line& line3, int n, int m) {

for (int i = 1; i <m; i++) { // Добавление элементов до заданного номера в промежуточную очередь

push(line1, line.head->data);

pop(line);

}

push(line2, line.head->data); // Добавление эелемента на заднном номере в промежуточную очередь

pop(line);

for (int i = m+1; i < n; i++) {

push(line3, line.head->data); // Добавление элементов после заданного номера в промежуточную очередь

pop(line);

}

cout << "Первая промежуточная очередь: "; printLine(line1);

cout << "Вторая промежуточная очередь: "; printLine(line2);

cout << "Третья промежуточная очередь: "; printLine(line3);

}

// Функция для восстановления очереди по 1-му заданию

void copy\_line1(Line& source1, Line& source2, Line& destination) {

cout << endl << "Восстановление исходной очереди: " << endl;

while (source1.head != nullptr) {

push(destination, source1.head->data); // Копирование значения элемента из промежуточной очереди в изначальную

printLine(destination);

pop(source1); // Переход к следующему элементу в промежуточной очереди

}

while (source2.head != nullptr) {

push(destination, source2.head->data); // Копирование значения элемента из промежуточной очереди в изначальную

printLine(destination);

pop(source2); // Переход к следующему элементу в промежуточной очереди

}

cout << "Очередь с удаленным ключом: "; printLine(destination);

}

// Функция для восстановления очереди по 2-му заданию

void copy\_line2(Line& source1, Line& source2, Line& source3, Line& destination) {

string s;

int k;

while (source1.head != nullptr) {

push(destination, source1.head->data); // Копирование значения элемента из промежуточной очереди в изначальную

pop(source1); // Переход к следующему элементу в промежуточной очереди

}

cout << "Введите количество доабвлеямых в очередь элементов: "; cin >> k;

for (int i = 1; i <= k; i++) {

cout << "Введите добавляемый элемент: "; cin >> s;

push(destination, s); // Добавление элементов перед заданным номером

}

push(destination, source2.head->data);

while (source3.head != nullptr) {

push(destination, source3.head->data); // Копирование значения элемента из промежуточной очереди в изначальную

pop(source3); // Переход к следующему элементу в промежуточной очереди

}

cout << "Очередь с добавленными элементами: "; printLine(destination);

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n, m;

string a, key;

Line, line1, line2, line3;

cout << "Введите количество элементов очереди: "; cin >> n;

for (int i = 1; i <= n; i++) {

cout << "Введите элемент очереди: "; cin >> a;

push(line, a);

}

cout << endl << "Ваша очередь: ";

printLine(line);

cout << endl;

cout << "Введите ключ для удаления: "; cin >> key;

delete\_key(line, line1, line2, key);

copy\_line1(line1, line2, line);

cout << endl;

cout << "Введите номер элемента, перед которым нужно добавить элементы: "; cin >> m;

delete\_elem(line, line1, line2, line3, n, m);

copy\_line2(line1, line2, line3, line);

return 0;

}

**Вывод программы**

