Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное учреждение высшего образования  
«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т  
по лабораторной работе**

Дисциплина: «Основы алгоритмизации и программирования»

Выполнил  
студент группы КЗИ-21-1б  
Полковников Т.Ю.

Проверил  
доцент кафедры ИТАС  
ст. преп. Яруллин Д.В.

Пермь, 2022

## Постановка задачи:

1. Сформировать однонаправленный и двунаправленный списки или стек и очередь. Тип информационного поля указан в варианте.

2. Распечатать полученную структуру.

3. Выполнить обработку структуры в соответствии с заданием.

4. Распечатать полученный результат.

5. Удалить соответствующую структуру из памяти.

## Анализ решения задачи:

Тип информационного поля char\*. Добавить в список элемент после элемента с заданным информационным полем.

## Описание переменных:

queue\_1 – начальная очередь

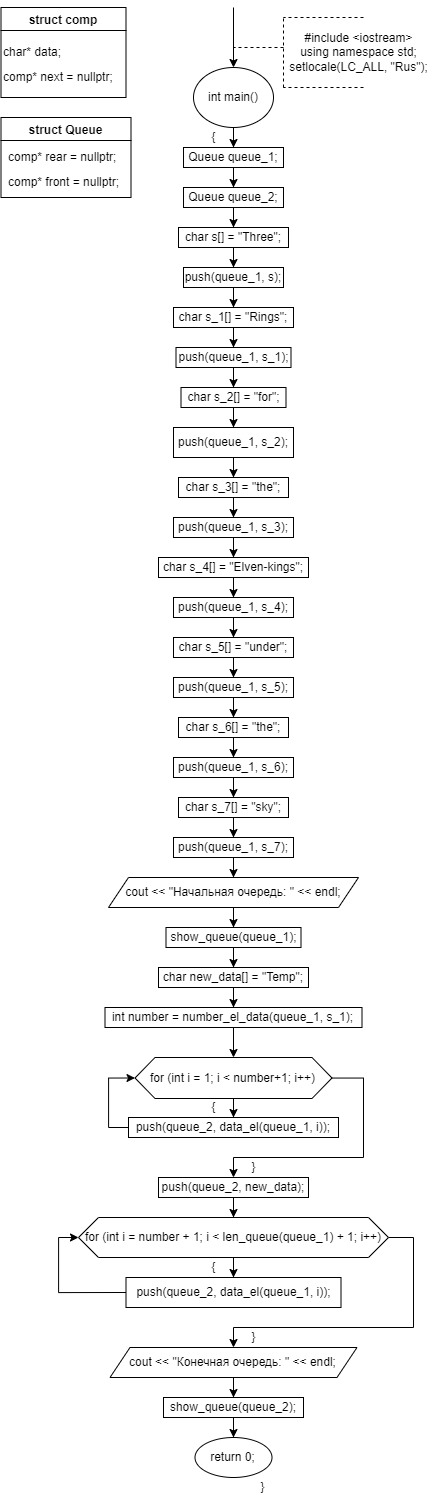
queue\_2 – конечная очередь

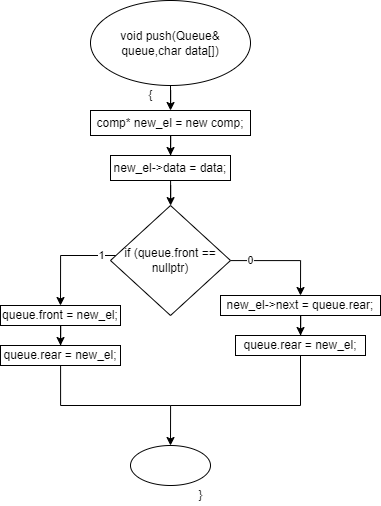
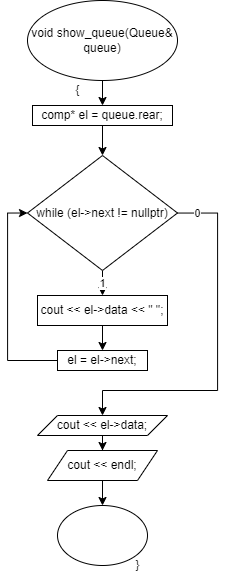
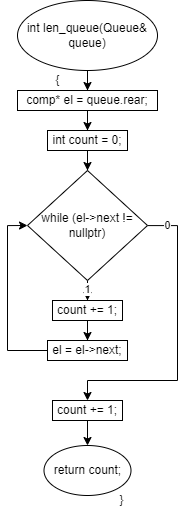
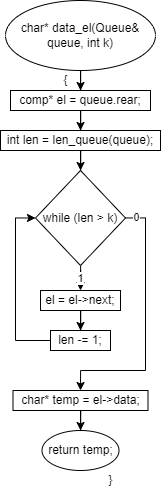
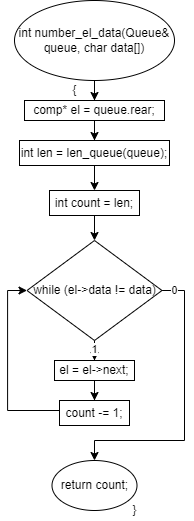
s, s\_1, s\_2, … s\_7 – строки для заполнения списка

new\_data – строка для информационного поля нового элемента очереди

number – номер элемента стека с заданным информационным полем

**Блок схема**



## Код программы

#include <iostream>

using namespace std;

// компонент очереди

struct comp {

char\* data;

comp\* next = nullptr;

};

// очередь

struct Queue {

comp\* rear = nullptr;

comp\* front = nullptr;

};

// добавление элемента

void push(Queue& queue,char data[]) {

comp\* new\_el = new comp;

new\_el->data = data;

if (queue.front == nullptr) {

queue.front = new\_el;

queue.rear = new\_el;

}

else {

new\_el->next = queue.rear;

queue.rear = new\_el;

}

}

// вывод очереди

void show\_queue(Queue& queue) {

comp\* el = queue.rear;

while (el->next != nullptr) {

cout << el->data << " ";

el = el->next;

}

cout << el->data;

cout << endl;

}

// длинна очереди

int len\_queue(Queue& queue) {

comp\* el = queue.rear;

int count = 0;

while (el->next != nullptr) {

count += 1;

el = el->next;

}

count += 1;

return count;

}

//поле элемента по его номеру

char\* data\_el(Queue& queue, int k) {

comp\* el = queue.rear;

int len = len\_queue(queue);

while (len > k) {

el = el->next;

len -= 1;

}

char\* temp = el->data;

return temp;

}

// номер элемента по его полю

int number\_el\_data(Queue& queue, char data[]) {

comp\* el = queue.rear;

int len = len\_queue(queue);

int count = len;

while (el->data != data) {

el = el->next;

count -= 1;

}

return count;

}

int main()

{

setlocale(LC\_ALL, "Rus");

Queue queue\_1;

Queue queue\_2;

// заполнение очереди

char s[] = "Three";

push(queue\_1, s);

char s\_1[] = "Rings";

push(queue\_1, s\_1);

char s\_2[] = "for";

push(queue\_1, s\_2);

char s\_3[] = "the";

push(queue\_1, s\_3);

char s\_4[] = "Elven-kings";

push(queue\_1, s\_4);

char s\_5[] = "under";

push(queue\_1, s\_5);

char s\_6[] = "the";

push(queue\_1, s\_6);

char s\_7[] = "sky";

push(queue\_1, s\_7);

// вывод начальной очереди

cout << "Начальная очередь: " << endl;

show\_queue(queue\_1);

// информционное поле для нового элемента

char new\_data[] = "Temp";

// добавление нового элемента

int number = number\_el\_data(queue\_1, s\_1);

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

push(queue\_2, data\_el(queue\_1, i));

}

push(queue\_2, new\_data);

for (int i = number + 1; i < len\_queue(queue\_1) + 1; i++) {

push(queue\_2, data\_el(queue\_1, i));

}

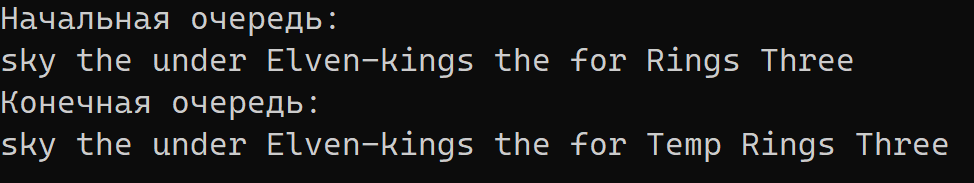
//вывод конечной очереди

cout << "Конечная очередь: " << endl;

show\_queue(queue\_2);

return 0;

Скриншоты результатов:



Анализ результатов:

Программа работает правильно. Выводиться начальная очередь, и очередь с новым элементом.