Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа № 11.4**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**(Семестр** 2)

Выполнил студент гр. ИВТ-21-1б

Гребнев Алексей Дмитриевич

Проверил:

Яруллин Денис Владимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г.Пермь -2022

**Постановка задачи**

1. Сформировать 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.

Выполнить двумя способами (без библиотеки queue и вместе с ней).

**Исходные данные**

Без библиотеки

1. Создаем структуру Node с информационным полем типа char и указателем на следующий элемент;
2. В функции main просим пользователя ввести количество элементов, которое мы передадим в функцию push;
3. Пользователь вводит элемент вне цикла, и он становится первым;
4. Заходим в цикл и выделяем память под новый элемент, пользователь вводит значение с клавиатуры, меняем указатели;
5. В функции main вызываем функцию для вывода элементов на экран - print типа void, в которую передаем начало очереди.

#include <iostream>

using namespace std;

template <typename T>

struct Node //T - тип данных

{

T data; //данные

Node<T>\* next; //указатель на следующий элемент

};

template <typename T>

struct Queue

{

int size = 0; //размер

Node<T>\* head = nullptr; // указатель на голову

Node<T>\* tail = nullptr; //указатель на хвост

};

template <typename T> //T - тип данных

void new\_queue(Queue<T>& q, const int& n)

{

T a;

cout << "Задайте " << n << " элементов для очереди ";

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

{

cin >> a;

push(q, a); //Функция элемент и вставляем в конец очереди (q-куда, a-что )

}

}

template <typename T>

void push(Queue<T>& q, const T& value)

{

if (q.head != nullptr) //если очередь не пуста

{

Node<T>\* new\_node = new Node<T>; //Резервируем память под новый элемент

q.size++; //Увеличиваем размер очереди на 1

new\_node->data = value; //в поле ключа присваиваем передаваемое значение

new\_node->next = nullptr; //Последний элемент ни на что не указывает

q.tail->next = new\_node; //предпоследний элемент указывает на последний

q.tail = new\_node; //указатель на последний элемент теперь указывает на новый

}

else

{

Node<T>\* new\_node = new Node<T>; //резервируем память под первый элемент

new\_node->data = value; //в поле ключа присваиваем передаваемое значение

q.head = new\_node;

q.tail = new\_node;

q.size = 1;

}

}

template <typename T>

void pop(Queue<T>& q)

{

Node<T>\* tmp = q.head;

q.head = q.head->next;

q.size--;

delete tmp;

}

template <typename T>

void print(Queue<T>& q)

{

Node<T>\* tmp = q.head;

while (tmp != nullptr)

{

cout << tmp->data;

tmp = tmp->next;

}

}

template <typename T>

void insert(Queue<T>& q, const int& index)

{

const int count = 1;

int i = 1;

while (i < index + 1) //пока не дойдём до нужного элемента

{

push(q, q.head->data);

pop(q); //переставляем элементы с головы в хвост

i++;

}

T a;

cout << "Введите элемент для добавления: ";

for (int j = 0; j < count; j++)

{

cin >> a;

push(q, a); //добавляем count новых элементов

}

for (i; i <= q.size - count; i++)//чтобы первый элемент оказался на своём месте

{

push(q, q.head->data);

pop(q);//переставляем элементы с головы в хвост

}

}

template <typename T>

void deleteQ(Queue<T>& q)

{

while (q.head != nullptr)

{

pop(q);

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

int n;

int index = 3;

cout << "Введите размер очереди: ";

cin >> n;

cout << endl;

Queue<char> q;

new\_queue(q, n);

cout << endl;

cout << "Введите номер элемента, после которого мы добавим новый элемент: ";

cin >> index;

cout << endl;

insert(q, index);

cout << endl;

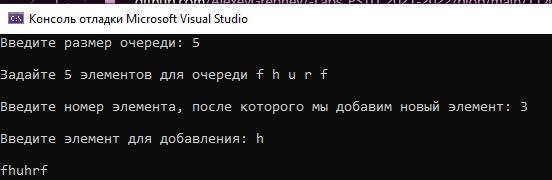
print(q);

deleteQ(q);

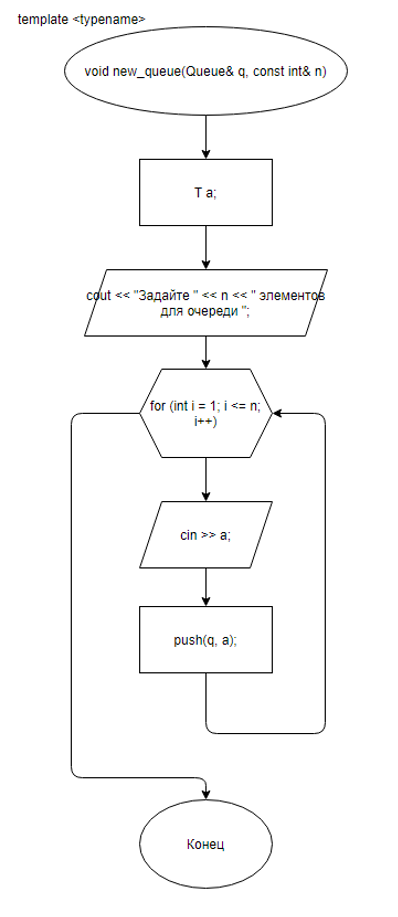
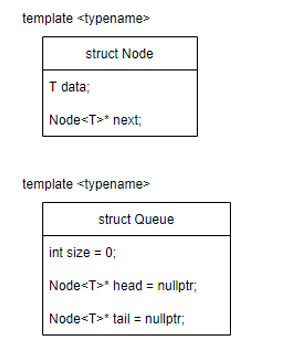
return 0;

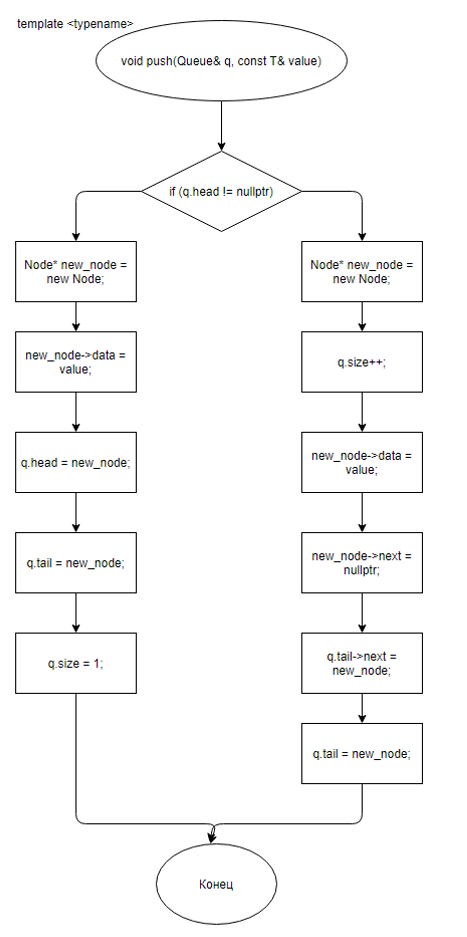
}

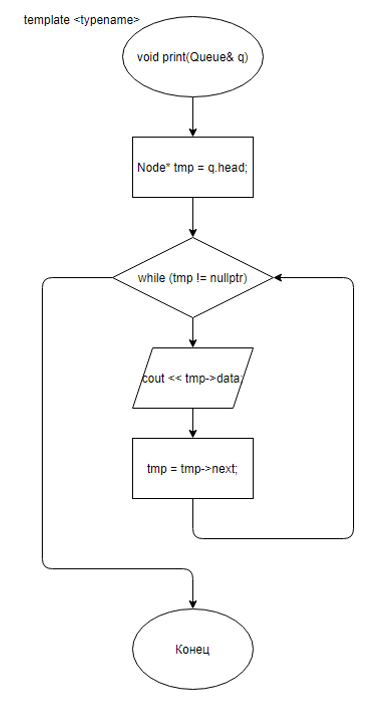
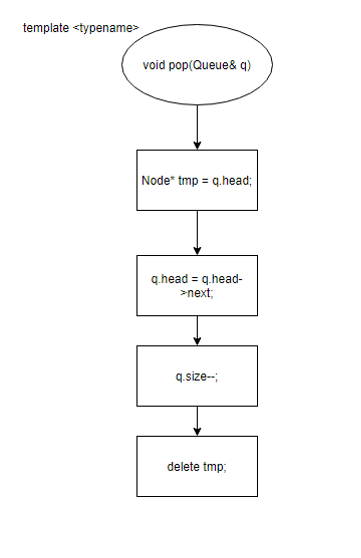
Результат

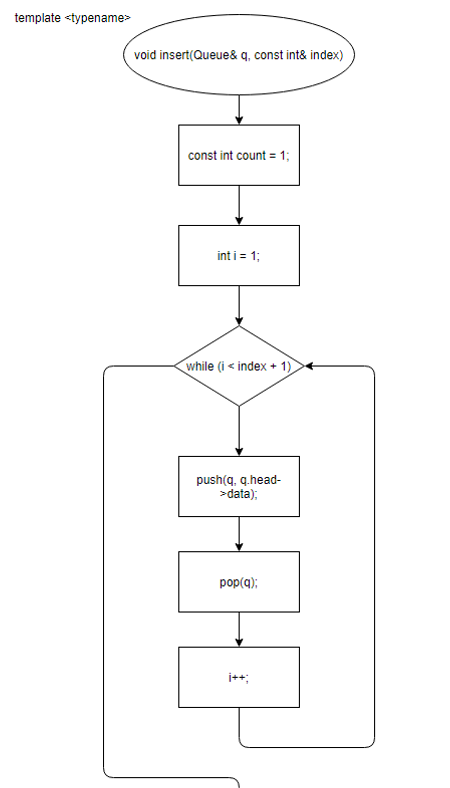


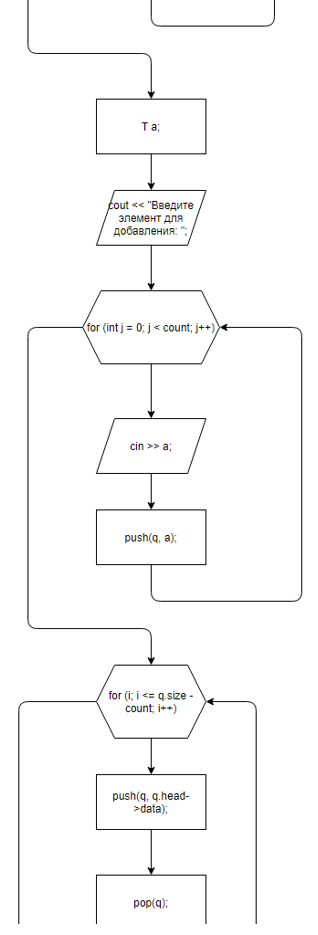
Блок-схема

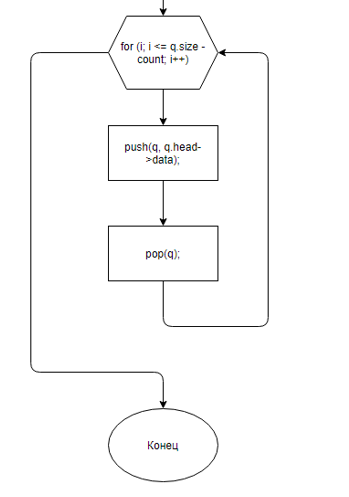


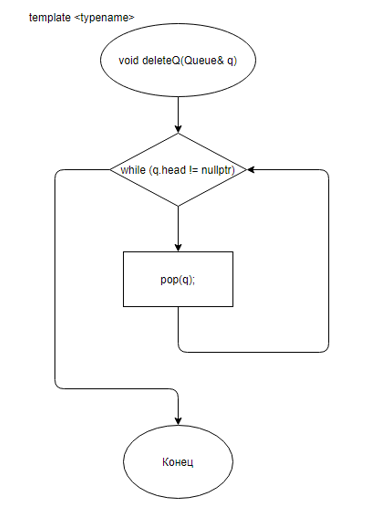


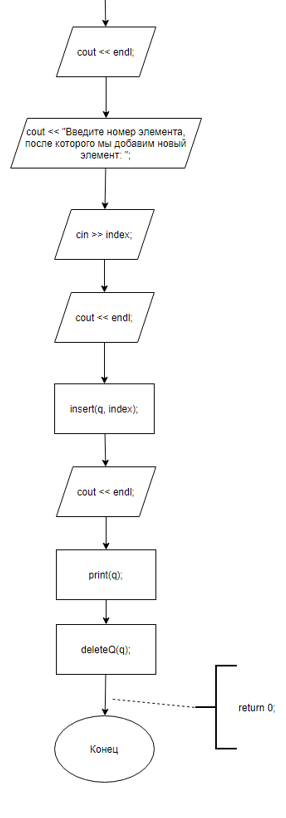
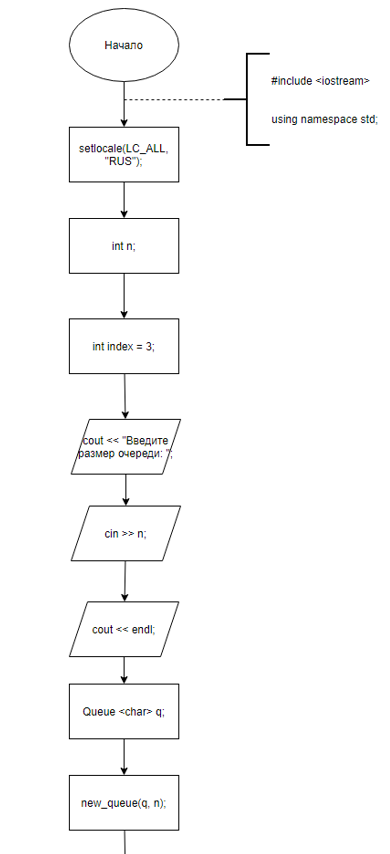












С помощью библиотеки

1. Подключаем библиотеку <queue>, создаем переменную int n(количество элементов),сhar a(переменная для ввода данных);
2. Создаем очередь с типом данных char с названием qu;
3. Вводим кол-во элементов
4. Выводим элементы
5. Проходим по функции change
6. Далее выводим результат

#include <iostream>

#include <queue>

using namespace std;

void print(queue<char> q)

{

int n = q.size();

for (int i = 0; i < n; i++)

{

cout << q.front();

q.pop();

}

}

template <typename T>

void change(Queue<T>& q, const int& index)

{

int count = 1;

int i = 1;

while (i < index + 1) //пока не дойдём до нужного элемента

{

q.push(q.front());

q.pop();

i++;

}

char a;

cout << "Введите элемент для добавления: ";

for (int j = 0; j < count; j++)

{

cin >> a;

q.push(a); //добавляем count новых элементов

}

for (i; i <= q.size - count; i++)//чтобы первый элемент оказался на своём месте

{

q.push(q.front());

q.pop();

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int n;

char a;

int index;

cout << "Введите размер очереди: ";

cin >> n;

queue<char> qu;

for (int i = 0; i < n; i++)

{

cin >> a;

qu.push(a);

}

print(qu);

cout << endl;

cout << "Введите номер элемента, после которого мы добавим новый элемент: ";

cin >> index;

cout << endl;

change(q, index);

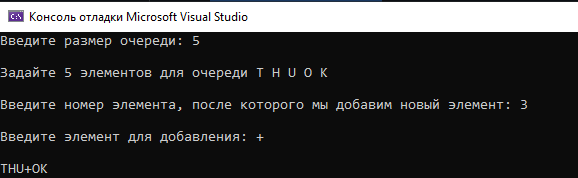
cout << endl;

print(q);

return 0;

}

Результат



Блок-схема

