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 ИВТ-21-1б  
Мифтахов И.Г

Проверил  
ст. п. кафедры ИТАС  
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.

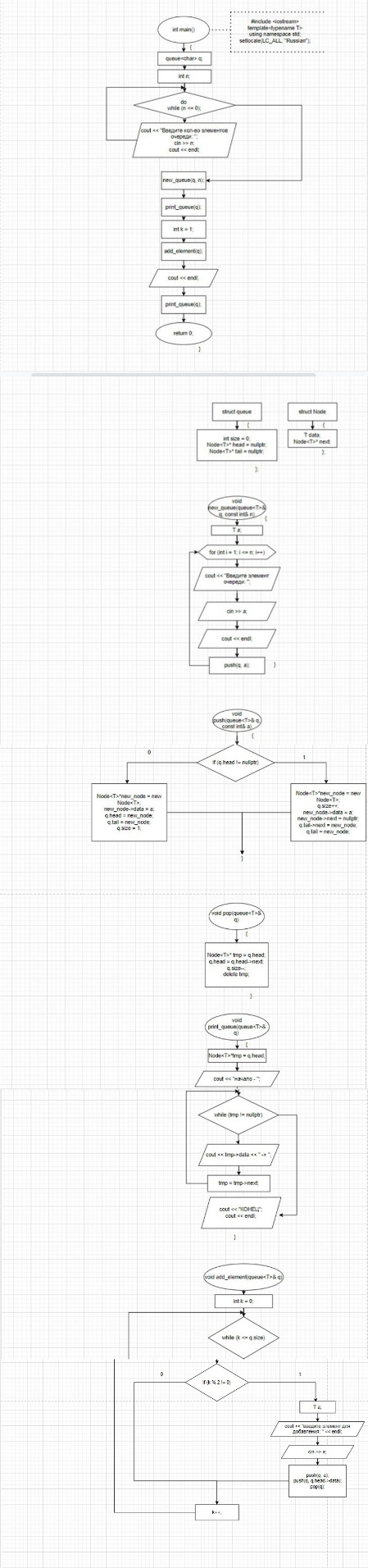
5. Удалить соответствующую структуру из памяти.

Выполнить двумя способами.

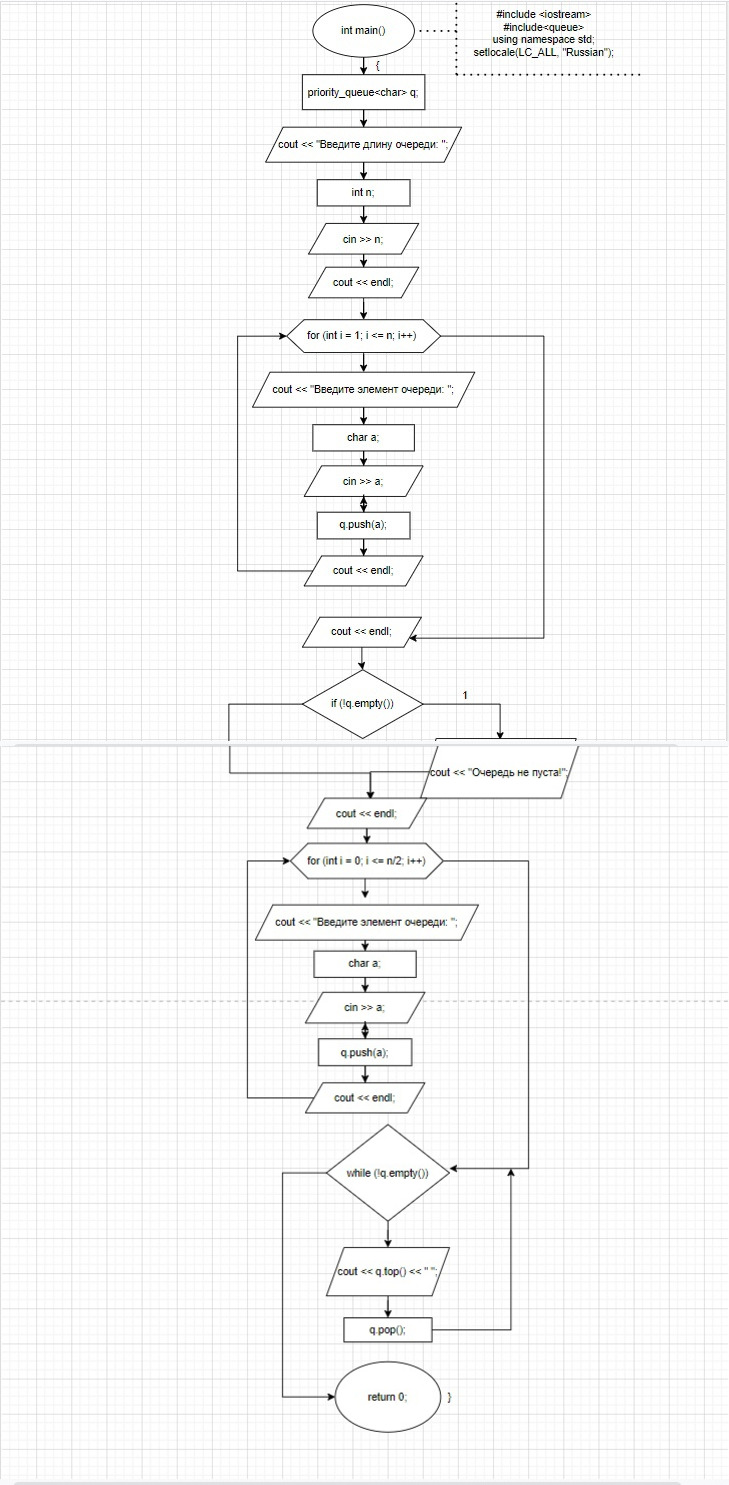
Анализ задачи:

1. Везде использован шаблон функции для свободного назначения типа поля структуры. Две структуры – первая(аналогична однонаправленному списку) и вторая(аналогична двунаправленному списку). Каждое действие – новая функция. В функции new\_queue создаем элементы и переходим к функции push, которая как раз и создает поле и работает с указателями. Вывод очереди аналогична выводу однонаправленного списка. Добавление элемента в очередь аналогично созданию очереди.
2. Для работы с готовыми командами я реализовал задачу через приоритетную очередь. Подключил библиотеку «queue», создал очередь в цикле for, перед этим задав количество членов очереди. Проверил, не пуста ли очередь. Добавлял элементы аналогично созданию очереди, в таком же цикле, только элементов было на n/2-1 меньше. Приоритетная очередь упорядочила их в порядке убывания. Вывод осуществил через доступ к верхнему(первому) элементу и его удаление в цикле while, пока очередь не пуста.

Блок-схема 1 варианта:



Блок-схема 2 варианта:



Код программы 1 варианта:

#include<iostream>

using namespace std;

template<typename T>

struct Node

{

T data;

Node<T>\* next;

};

template<typename T>

struct queue

{

int size = 0;

Node<T>\* head = nullptr;

Node<T>\* tail = nullptr;

};

template<typename T>

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

{

T a;

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

{

cout << "Введите элемент очереди: ";

cin >> a;

cout << endl;

push(q, a);

}

}

template<typename T>

void push(queue<T>& q, const int& a)

{

if (q.head != nullptr)

{

Node<T>\*new\_node = new Node<T>;

q.size++;

new\_node->data = a;

new\_node->next = nullptr;

q.tail->next = new\_node;

q.tail = new\_node;

}

else

{

Node<T>\*new\_node = new Node<T>;

new\_node->data = a;

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(queue<T>& q)

{

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

cout << "начало - ";

while (tmp != nullptr)

{

cout << tmp->data << " -> ";

tmp = tmp->next;

}

cout << "КОНЕЦ";

cout << endl;

}

template<typename T>

void add\_element(queue<T>& q)

{

int k = 0;

while (k <= q.size)

{

if (k % 2 != 0)

{

T a;

cout << "введите элемент для добавления: " << endl;

cin >> a;

push(q, a);

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

pop(q);

}

k++;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

queue<char> q;

int n;

do

{

cout << "Введите кол-во элементов очереди: ";

cin >> n;

cout << endl;

} while (n <= 0);

new\_queue(q, n);

print\_queue(q);

int k = 1;

add\_element(q);

cout << endl;

print\_queue(q);

return 0;

}

Код программы 2 варианта:

#include<iostream>

#include<queue>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

priority\_queue<char> q;

cout << "Введите длину очереди: ";

int n;

cin >> n;

cout << endl;

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

{

cout << "Введите элемент очереди: ";

char a;

cin >> a;

q.push(a); // добавляем в очередь элементы

cout << endl;

}

cout << endl;

if (!q.empty()) cout << "Очередь не пуста!"; // проверяем пуста ли очередь (нет)

cout << endl;

for (int i = 0; i <= n/2; i++)

{

cout << "Введите элемент очереди: ";

char a;

cin >> a;

q.push(a); // добавляем в очередь элементы

cout << endl;

}

while (!q.empty()) {

cout << q.top() << " ";

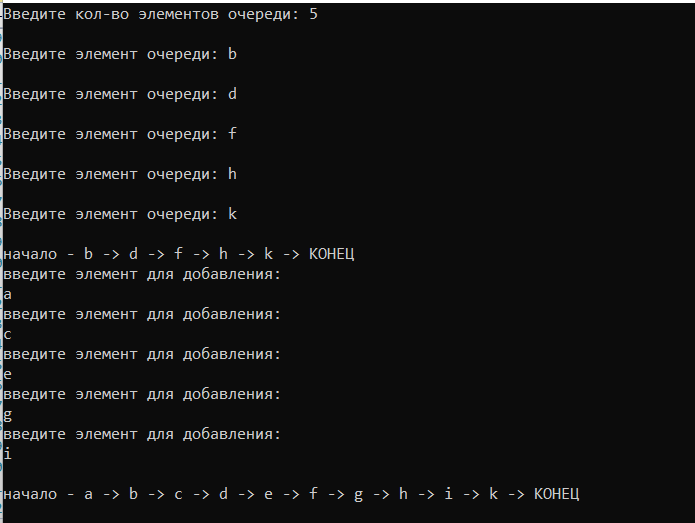
q.pop();

}

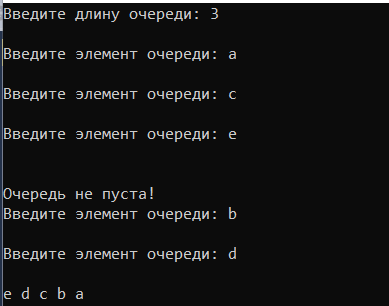
return 0;

}

Скриншоты результатов 1 варианта:



Скриншоты результатов 2 варианта:



Анализ результатов:

Программы работают корректно.