Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Информатика»

Тема: Очереди

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Тарасов C.В.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**Постановка задачи**

Написать программу, в которой реализуется работа с очередью.

А именно:

1. Добавление элемента

2. Удаление элемента

3. Показ первого элемента очереди

4. Показ последнего элемента очереди

**Анализ решения**

Очередью (англ. – queue) называется структура данных, из которой удаляется первым тот элемент, который был первым в очередь добавлен. То есть очередь в программировании соответствует «бытовому» понятию очереди. Очередь также называют структурой типа FIFO (first in, first out — первым пришел, первым ушел).

Для реализации поставленной задачи нужно 2 структуры: Node и Queue.

struct Node {

int data;

Node\* next = nullptr;

};

struct Queue {

int size = 0;

Node\* head = nullptr;

Node\* tail = nullptr;

};

Функция удаления элемента всегда будет удалять первый элемент в очереди.

Функция добавления элемента будет добавлять элемент в конец очереди.

Функция показа очереди может осуществляться с двух сторон – с начала и с конца.

**Код программы**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next = nullptr;

};

struct Queue {

int size = 0;

Node\* head = nullptr;

Node\* tail = nullptr;

};

void push(Queue\* queue, int value);

void pop(Queue\* queue);

void showl(Queue\* queue);

void showf(Queue\* queue);

int main()

{

setlocale(LC\_ALL, "Russian");

Queue\* queue = new Queue;

int n, value;

do {

system("cls");

cout << "Введите 0 для завершения работы\n";

cout << "Введите 1 для добавления элемента \n";

cout << "Введите 2 для удаление элемента\n";

cout << "Введите 3 для показа первого элемента очереди\n";

cout << "Введите 4 для показа последнего элемента очереди\n";

cout << "============================================================================\n";

cin >> n;

switch (n)

{

case 1:

{

cout << " Введите значение элемента "; cin >> value;

push(queue, value);

break;

}

case 2:

{

pop(queue);

break;

}

case 3:

{

showf(queue);

cout << " Введите чтобы продолжить ";cin >> value;

break;

}

case 4:

{

showl(queue);

cout << " Введите чтобы продолжить ";cin >> value;

break;

}

case 0:

cout << " Завершение работы ";

break;

}

} while (n != 0);

delete queue;

}

void push(Queue\* queue, int value)

{

Node\* new\_node = new Node;

new\_node->data = value;

new\_node->next = nullptr;

if (queue->head == nullptr)

{

queue->head = new\_node;

queue->tail = new\_node;

}

else

{

queue->tail->next = new\_node;

queue->tail = new\_node;

}

queue->size++;

}

void pop(Queue\* queue)

{

if (queue->head != nullptr)

{

Node\* temp = queue->head;

queue->head = temp->next;

if (queue->head == nullptr)

{

queue->tail = nullptr;

}

queue->size--;

delete temp;

}

}

void showf(Queue\* queue)

{

if (queue->head == nullptr)

{

cout << "Очередь пуста\n";

}

else

{

cout << "Первый элемент очереди - " << queue->head->data << endl;

}

}

void showl(Queue\* queue)

{

if (queue->head == nullptr)

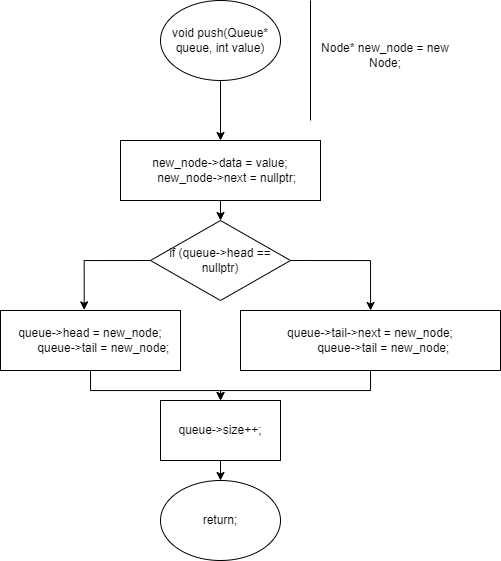
cout << "Очередь пуста\n";

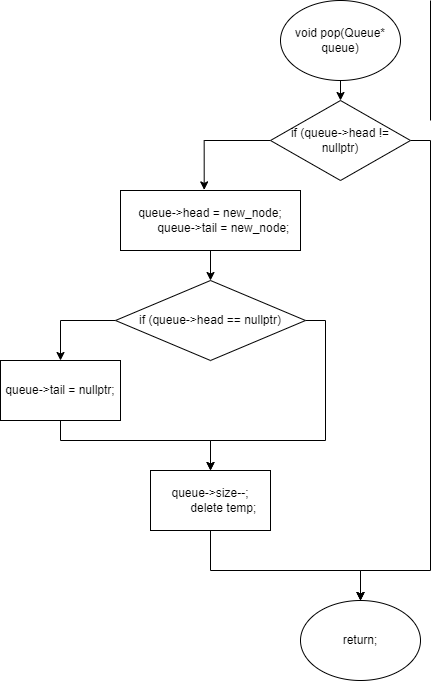
else

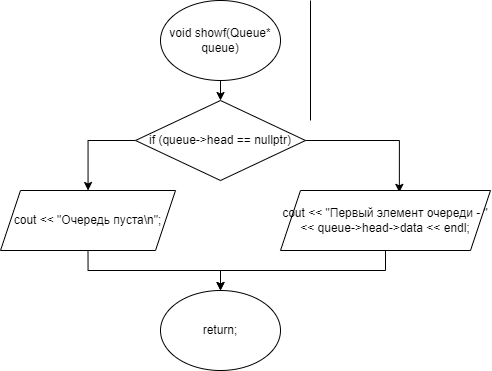
cout << "Последний элемент очереди - " << queue->tail->data << endl;

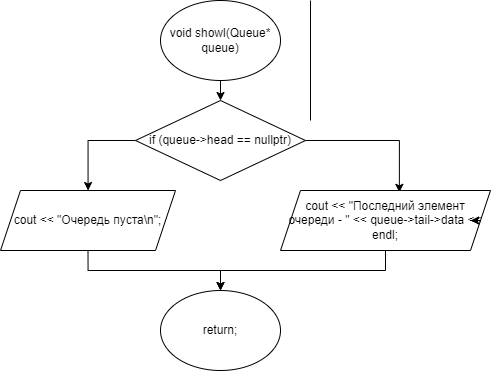
}

**Блок-схемы**

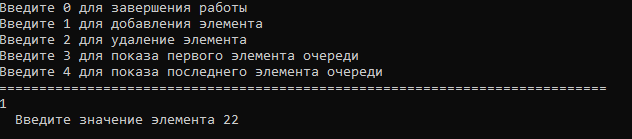
****

****

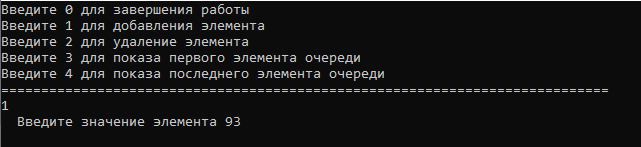
****

****

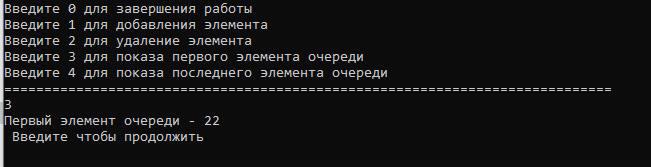
**Результат работы программы**



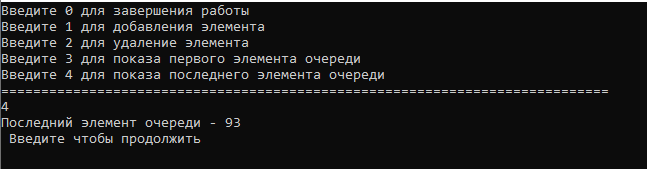
Добавление элемента



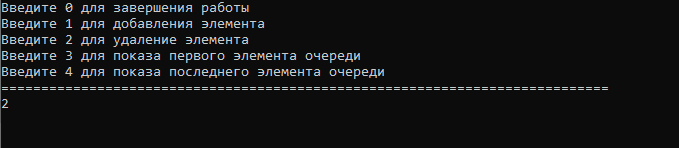
Добавление элемента



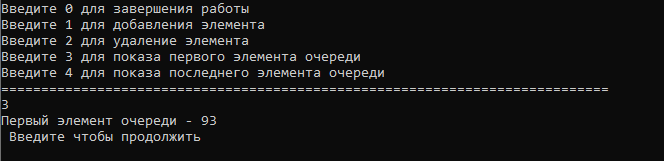
Показ первого элемента очереди



Показ последнего элемента очереди



Удаление элемента



Показ первого элемента в очереди