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: «Основы алгоритмизации и программирования»

Тема : "Очереди"

Выполнила работу:

Студентка группы РИС-22-1Б

Верхоланцева Е.С.

Проверила:

Доцент кафедры ИТАС

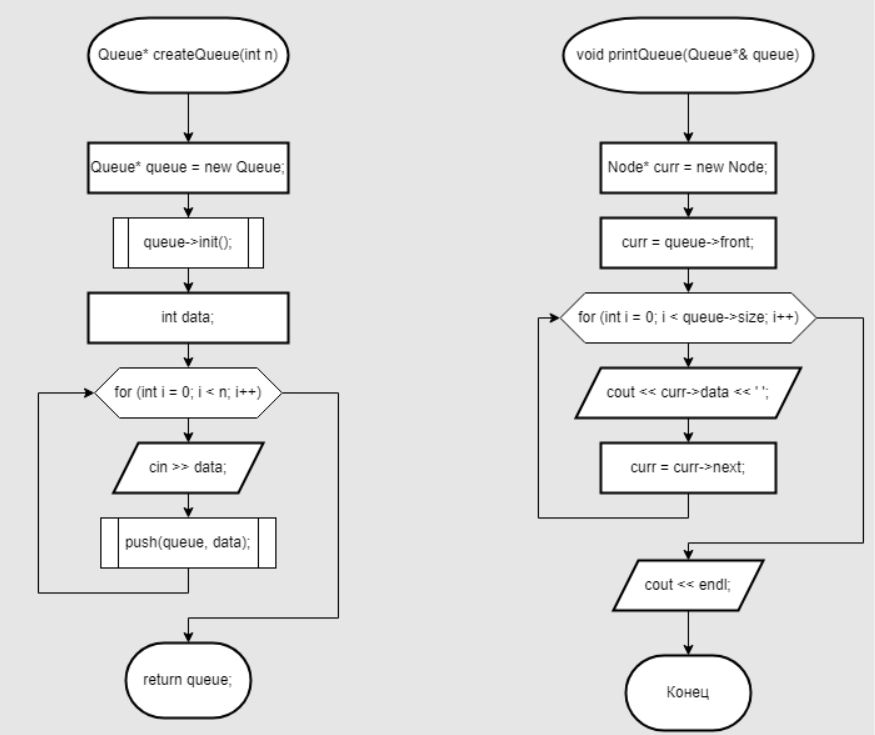
Полякова О.А.

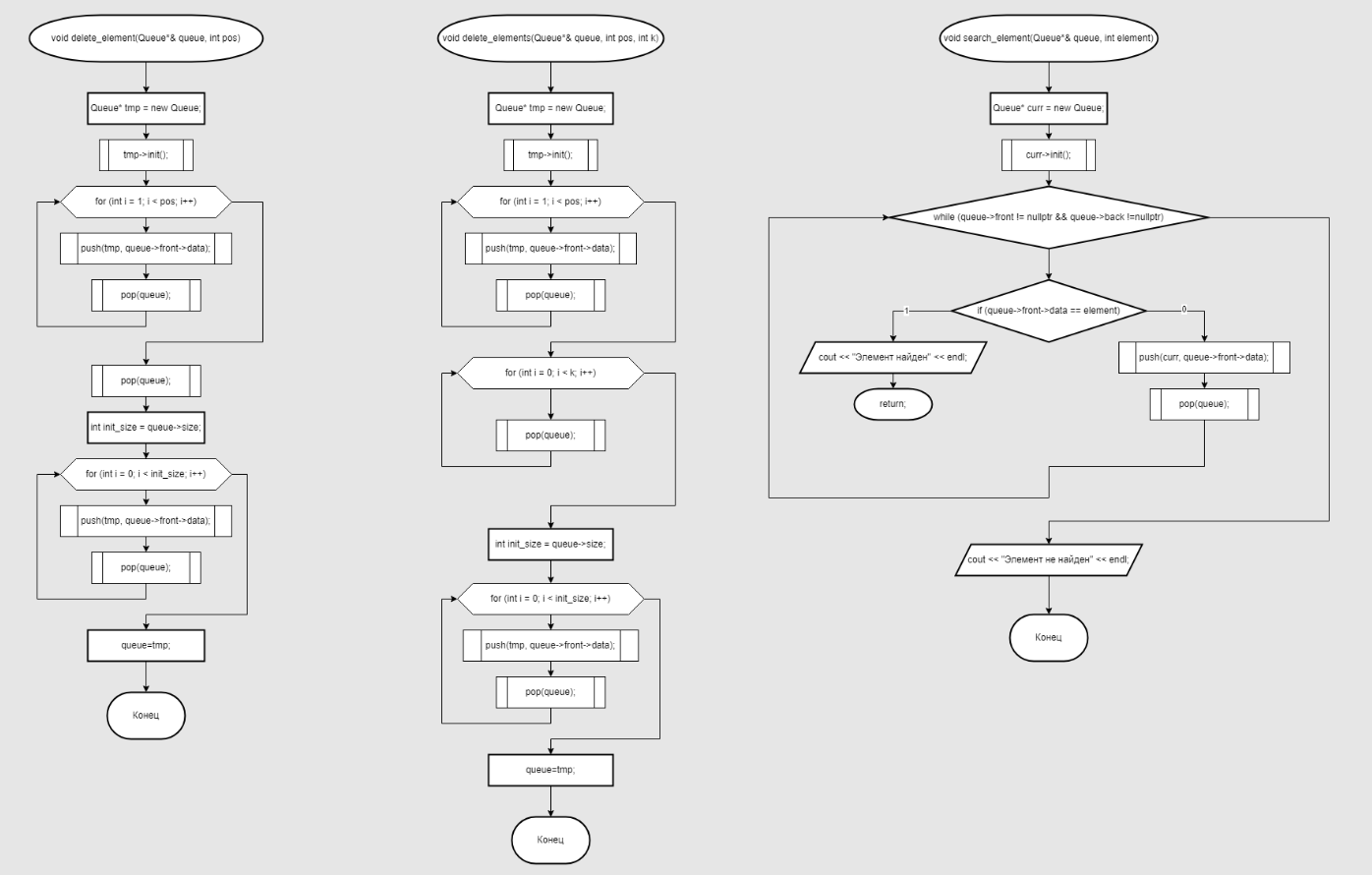
Г. Пермь-2023

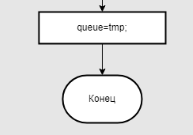
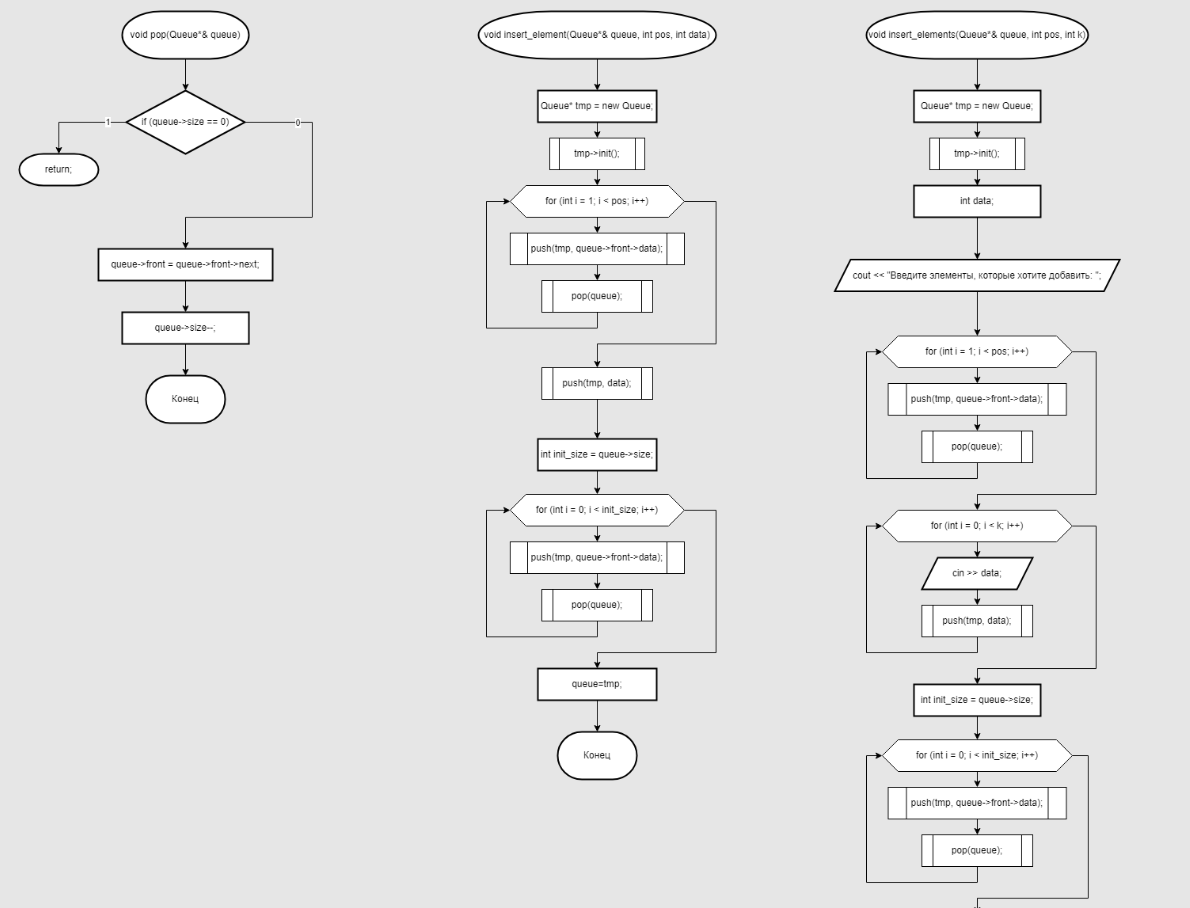
**1. Постановка задачи.**

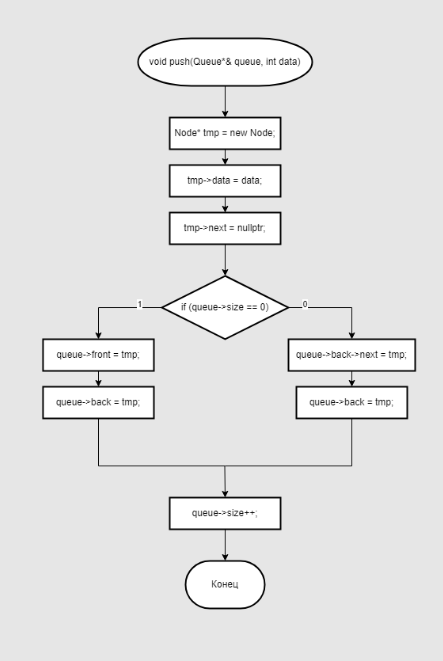
Сформировать очередь, добавить в неё k элементов, удалить из неё k элементов, реализовать поиск элемента по его значению.

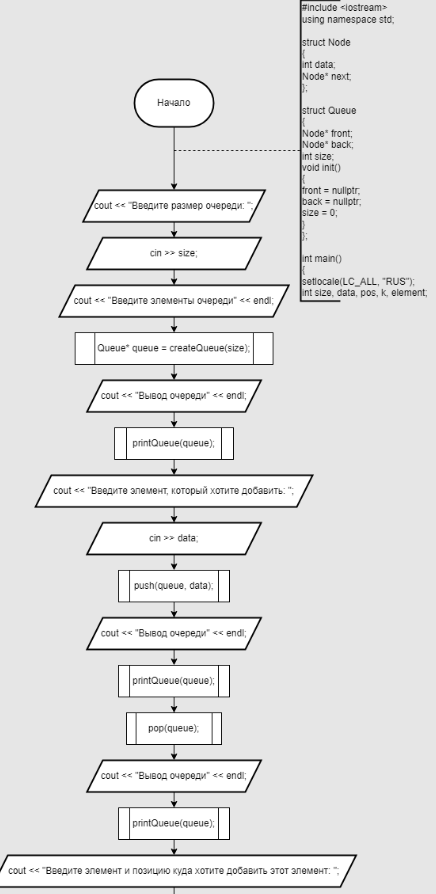
**2. Блок-схема.**

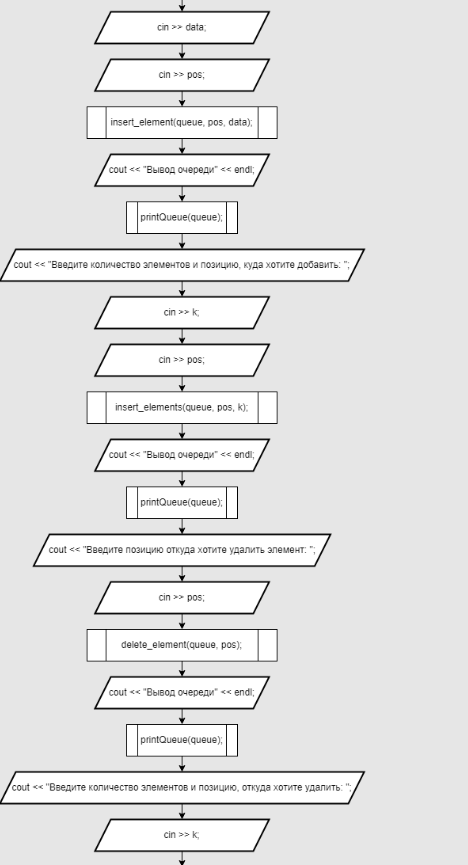


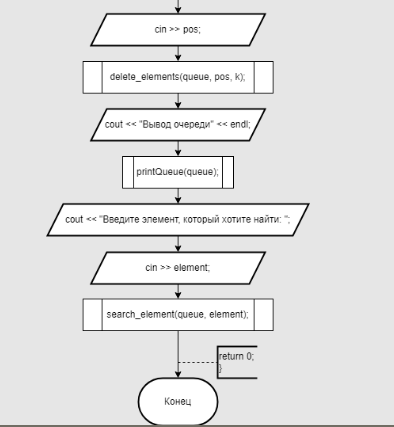












**3. Код.**

#include <iostream>

using namespace std;

struct Node // узел

{

int data;

Node\* next;

};

struct Queue // очередь

{

Node\* front;

Node\* back;

int size;

void init()

{

front = nullptr;

back = nullptr;

size = 0;

}

};

void push(Queue\*& queue, int data) // добавление элемента в очередь

{

Node\* tmp = new Node;

tmp->data = data;

tmp->next = nullptr;

if (queue->size == 0)

{

queue->front = tmp;

queue->back = tmp;

}

else

{

queue->back->next = tmp;

queue->back = tmp;

}

queue->size++;

}

void pop(Queue\*& queue) // удаление элемента очереди

{

if (queue->size == 0)

{

return;

}

queue->front = queue->front->next;

queue->size--;

}

void insert\_element(Queue\*& queue, int pos, int data) // вставка одного элемента в очередь на указанную позицию

{

Queue\* tmp = new Queue;

tmp->init();

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

{

push(tmp, queue->front->data);

pop(queue);

}

push(tmp, data);

int init\_size = queue->size;

for (int i = 0; i < init\_size; i++)

{

push(tmp, queue->front->data);

pop(queue);

}

queue = tmp;

}

void insert\_elements(Queue\*& queue, int pos, int k) // вставка k элементов в очередь на указанную позицию

{

Queue\* tmp = new Queue;

tmp->init();

int data;

cout << "Введите элементы, которые хотите добавить: ";

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

{

push(tmp, queue->front->data);

pop(queue);

}

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

{

cin >> data;

push(tmp, data);

}

int init\_size = queue->size;

for (int i = 0; i < init\_size; i++)

{

push(tmp, queue->front->data);

pop(queue);

}

queue = tmp;

}

void delete\_element(Queue\*& queue, int pos) // удаление одного элемента с заданной позиции

{

Queue\* tmp = new Queue;

tmp->init();

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

{

push(tmp, queue->front->data);

pop(queue);

}

pop(queue);

int init\_size = queue->size;

for (int i = 0; i < init\_size; i++)

{

push(tmp, queue->front->data);

pop(queue);

}

queue = tmp;

}

void delete\_elements(Queue\*& queue, int pos, int k) // удаление k элементов с заданной позиции

{

Queue\* tmp = new Queue;

tmp->init();

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

{

push(tmp, queue->front->data);

pop(queue);

}

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

{

pop(queue);

}

int init\_size = queue->size;

for (int i = 0; i < init\_size; i++)

{

push(tmp, queue->front->data);

pop(queue);

}

queue = tmp;

}

void search\_element(Queue\*& queue, int element) // поиск заданного элемента в очереди

{

Queue\* curr = new Queue;

curr->init();

while (queue->front != nullptr && queue->back != nullptr)

{

if (queue->front->data == element)

{

cout << "Элемент найден." << endl;

return;

}

else

{

push(curr, queue->front->data);

pop(queue);

}

}

cout << "Элемент не найден." << endl;

}

Queue\* createQueue(int n) // создание очереди

{

Queue\* queue = new Queue;

queue->init();

int data;

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

{

cin >> data;

push(queue, data);

}

return queue;

}

void printQueue(Queue\*& queue) // вывод очереди

{

Node\* curr = new Node;

curr = queue->front;

for (int i = 0; i < queue->size; i++)

{

cout << curr->data << ' ';

curr = curr->next;

}

cout << endl;

}

int main()

{

setlocale(LC\_ALL, "RUS");

int size, data, pos, k, element;

cout << "Введите размер очереди: ";

cin >> size;

cout << "Введите элементы очереди" << endl;

Queue\* queue = createQueue(size);

cout << "Вывод очереди:" << endl;

printQueue(queue);

cout << "Введите элемент, который хотите добавить: ";

cin >> data;

push(queue, data);

cout << "Вывод очереди:" << endl;

printQueue(queue);

pop(queue);

cout << "Вывод очереди" << endl;

printQueue(queue);

cout << "Введите элемент и позицию куда хотите добавить этот элемент: ";

cin >> data;

cin >> pos;

insert\_element(queue, pos, data);

cout << "Вывод очереди:" << endl;

printQueue(queue);

cout << "Введите количество элементов и позицию, куда хотите добавить: ";

cin >> k;

cin >> pos;

insert\_elements(queue, pos, k);

cout << "Вывод очереди:" << endl;

printQueue(queue);

cout << "Введите позицию откуда хотите удалить элемент: ";

cin >> pos;

delete\_element(queue, pos);

cout << "Вывод очереди:" << endl;

printQueue(queue);

cout << "Введите количество элементов и позицию, откуда хотите удалить: ";

cin >> k;

cin >> pos;

delete\_elements(queue, pos, k);

cout << "Вывод очереди:" << endl;

printQueue(queue);

cout << "Введите элемент, который хотите найти: ";

cin >> element;

search\_element(queue, element);

return 0;

}

**4. Скриншот вывода.**

