Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

ОТЧЁТ

«Лабораторная работа № 11. Информационные динамические структуры.

Двунаправленный список»

Выполнил

Студент группы РИС-22-2Б

Хохряков А.С.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Пермь 2023

# Содержание отчета

1. Постановка задачи.
2. Анализ задачи.
3. Блок-схема
4. Текст программы

Постановка задачи

Записи в линейном списке содержат ключевое поле типа int. Сформировать двунаправленный список. Удалить из него К элементов, начиная с заданного номера, добавить элемент перед элементом с заданным ключом;

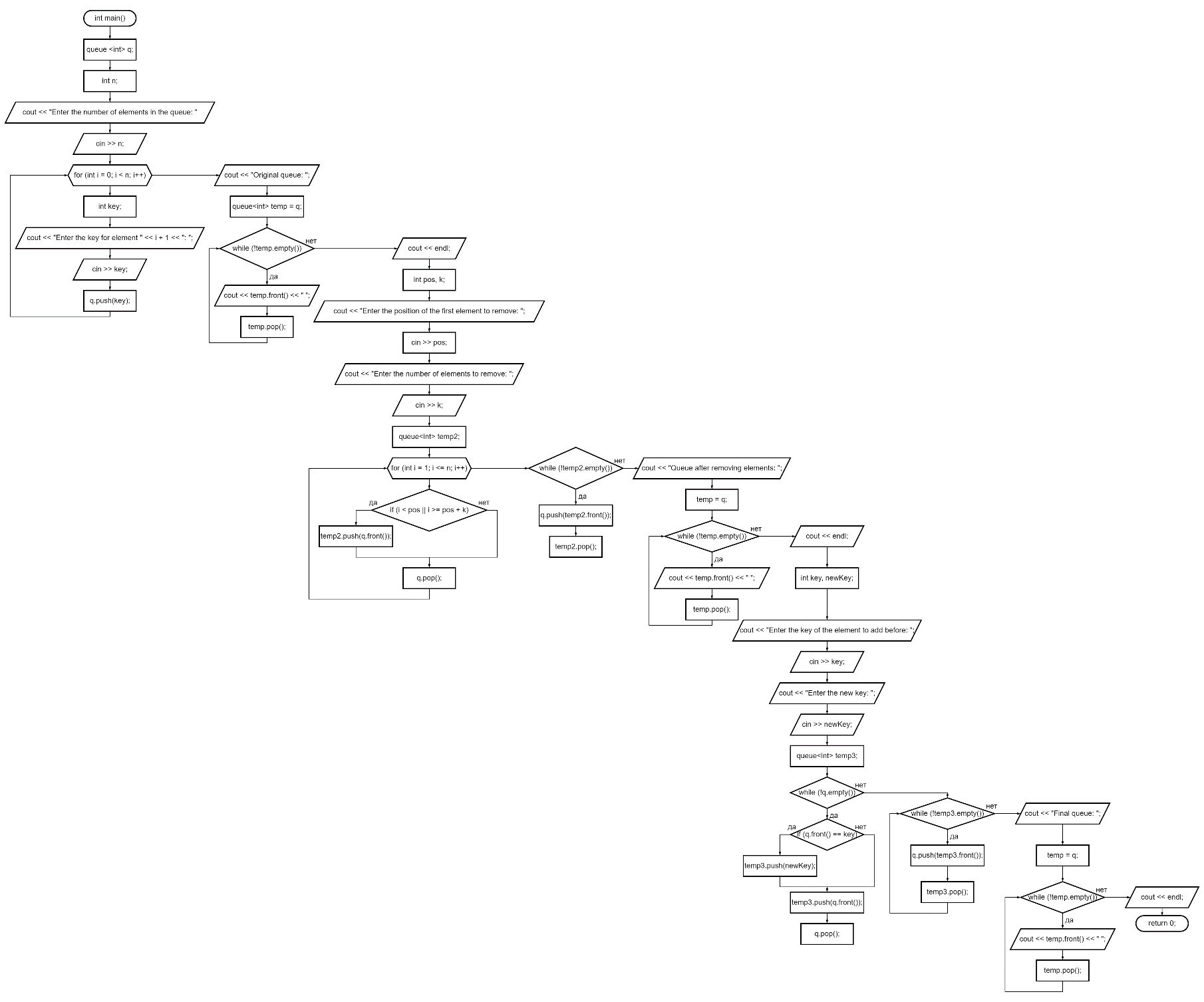
Анализ задачи

Задача решается с помощью двунаправленного линейного списка - структуры данных, которая состоит из элементов, каждый из которых содержит как минимум два поля: значение и указатель на следующий элемент списка.

Сначала мы создаем и заполняем исходный список данными, вводимыми пользователем. Затем мы выводим исходный список и приступаем к удалению элементов из заданного диапазона. Для этого мы проходимся по всем элементам списка, удаляя элементы в заданном диапазоне. Если элемент, который нужно удалить, находится в начале списка, мы изменяем указатель на начало списка. Если элемент находится в середине списка, мы изменяем указатель на следующий элемент у предыдущего элемента списка. Если элемент находится в конце списка, мы просто удаляем его.

Затем мы добавляем новый элемент перед элементом с заданным ключом. Для этого мы снова проходимся по всем элементам списка, ищем элемент с нужным ключом и добавляем новый элемент перед ним, изменяя указатели на элементы списка. В конце мы выводим окончательный результат - итоговый список.

Блок-схема



Текст программы

#include <iostream>

using namespace std;

// Структура элемента списка

struct Node {

int key;

Node\* next;

Node\* prev;

};

// Функция создания нового элемента списка

Node\* createNode(int key) {

Node\* node = new Node;

node->key = key;

node->next = NULL;

node->prev = NULL;

return node;

}

// Функция добавления нового элемента в конец списка

void addNode(Node\*& head, Node\*& tail, int key) {

Node\* node = createNode(key);

if (head == NULL) {

head = node;

tail = node;

}

else {

tail->next = node;

node->prev = tail;

tail = node;

}

}

// Функция удаления K элементов из списка, начиная с заданного индекса

void deleteNode(Node\*& head, Node\*& tail, int index, int k) {

Node\* node = head;

for (int i = 0; i < index && node != NULL; i++) {

node = node->next;

}

if (node == NULL) {

cout << "Invalid index" << endl;

return;

}

for (int i = 0; i < k && node != NULL; i++) {

Node\* nextNode = node->next;

if (node == head) {

head = nextNode;

}

if (node == tail) {

tail = node->prev;

}

if (node->prev != NULL) {

node->prev->next = node->next;

}

if (node->next != NULL) {

node->next->prev = node->prev;

}

delete node;

node = nextNode;

}

}

// Функция добавления нового элемента перед элементом с заданным ключом

void addNodeBeforeKey(Node\*& head, Node\*& tail, int searchKey, int newKey) {

Node\* node = head;

while (node != NULL && node->key != searchKey) {

node = node->next;

}

if (node == NULL) {

cout << "Key not found" << endl;

return;

}

Node\* newNode = createNode(newKey);

if (node->prev != NULL) {

node->prev->next = newNode;

newNode->prev = node->prev;

}

else {

head = newNode;

}

node->prev = newNode;

newNode->next = node;

}

// Функция вывода списка на экран

void printList(Node\* node) {

while (node != NULL) {

cout << node->key << " ";

node = node->next;

}

cout << endl;

}

int main() {

Node\* head = NULL;

Node\* tail = NULL;

int n, key;

cout << "Enter the number of elements: ";

cin >> n;

cout << "Enter " << n << " elements:" << endl;

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

cin >> key;

addNode(head, tail, key);

}

int index, k;

cout << "Enter the index and number of nodes to delete: ";

cin >> index >> k;

deleteNode(head, tail, index, k);

int searchKey, newKey;

cout << "Enter the search key and new key: ";

cin >> searchKey >> newKey;

addNodeBeforeKey(head, tail, searchKey, newKey);

cout << "Updated list: ";

printList(head);

return 0;

}