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.

Списки»

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 однонаправленный список. Удалить из него К элементов, начиная с заданного номера, добавить элемент перед элементом с заданным ключом;

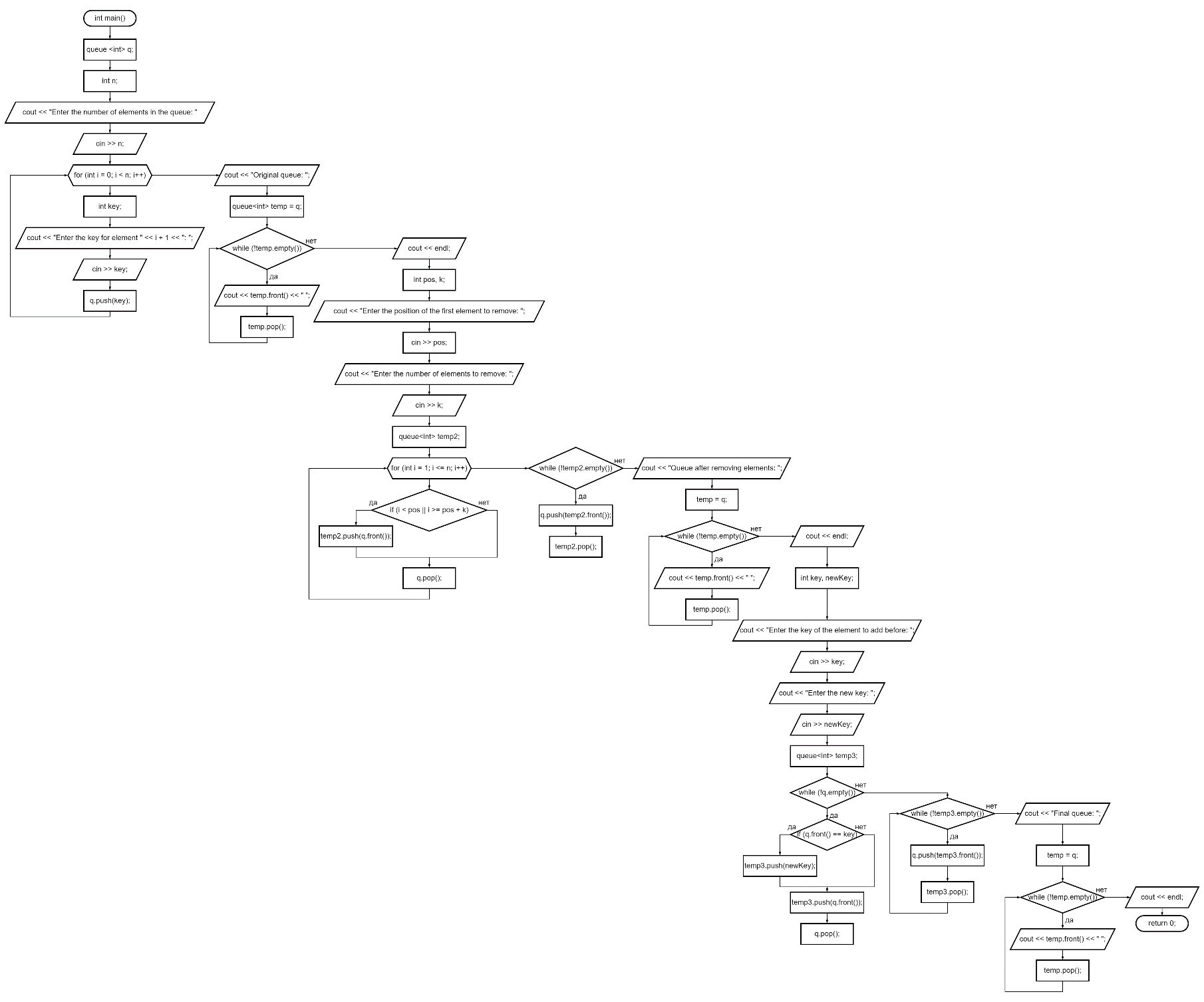
Анализ задачи

Задача решается с помощью односвязного 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;

struct Node {

int key;

Node\* next;

};

void deleteElements(Node\*& head, int start, int k) {

if (head == nullptr) return;

Node\* prev = nullptr;

Node\* curr = head;

int count = 1;

while (curr != nullptr && count < start) {

prev = curr;

curr = curr->next;

count++;

}

int deleteCount = 0;

while (curr != nullptr && deleteCount < k) {

Node\* temp = curr;

curr = curr->next;

delete temp;

deleteCount++;

}

if (prev == nullptr) {

head = curr;

}

else {

prev->next = curr;

}

}

void addBefore(Node\*& head, int key, int newKey) {

if (head == nullptr) return;

Node\* prev = nullptr;

Node\* curr = head;

while (curr != nullptr && curr->key != key) {

prev = curr;

curr = curr->next;

}

Node\* newNode = new Node;

newNode->key = newKey;

newNode->next = curr;

if (prev == nullptr) {

head = newNode;

}

else {

prev->next = newNode;

}

}

int main() {

Node\* head = nullptr;

int n;

cout << "Enter the number of items in the list: ";

cin >> n;

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

int key;

cout << "Enter key for element: " << i + 1 << ": ";

cin >> key;

Node\* newNode = new Node;

newNode->key = key;

newNode->next = head;

head = newNode;

}

cout << "Source List: ";

Node\* curr = head;

while (curr != nullptr) {

cout << curr->key << " ";

curr = curr->next;

}

cout << endl;

int start, k;

cout << "Enter the element number from which you want to delete elements: ";

cin >> start;

cout << "Enter the number of items to be removed: ";

cin >> k;

deleteElements(head, start, k);

cout << "List after elements removed: ";

curr = head;

while (curr != nullptr) {

cout << curr->key << " ";

curr = curr->next;

}

cout << endl;

int key, newKey;

cout << "Enter the element key before which you want to add the new element: ";

cin >> key;

cout << "Enter a new key: ";

cin >> newKey;

addBefore(head, key, newKey);

cout << "Final list: ";

curr = head;

while (curr != nullptr) {

cout << curr->key << " ";

curr = curr->next;

}

cout << endl;

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

return 0;

}