Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**Пермский национальный исследовательский политехнический университет**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Двунаправленный список»

Семестр 2

*Работу выполнил: Учащийся группы ИВТ - 22-2б: Цыбуцынин Фёдор Александрович*

*Работу проверил: доцент кафедры ИТАС:*

*Полякова Ольга Андреевна*

Г. Пермь – 2023

**Постановка задачи**

Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом.

1.Создание списка.

2.Добавление элемента в список (в соответствии со своим вариантом).

3.Удаление элемента из списка (в соответствии со своим вариантом).

4.Печать списка.

5.Запись списка в файл.

6.Уничтожение списка.

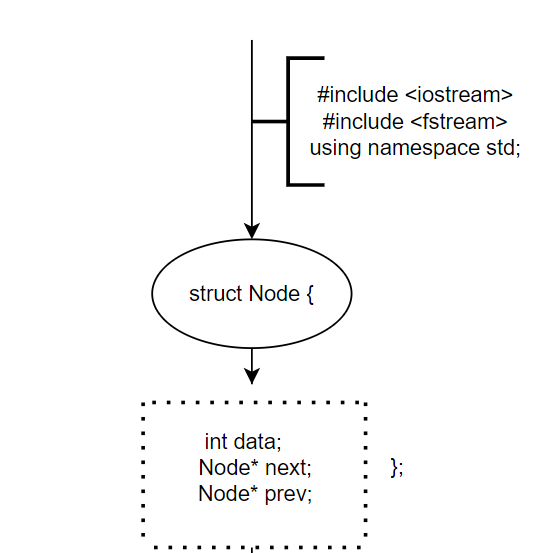
Восстановление списка из файла.

Записи в линейном списке содержат ключевое поле типа \*int (строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить К элементов перед элементом с заданным номером.

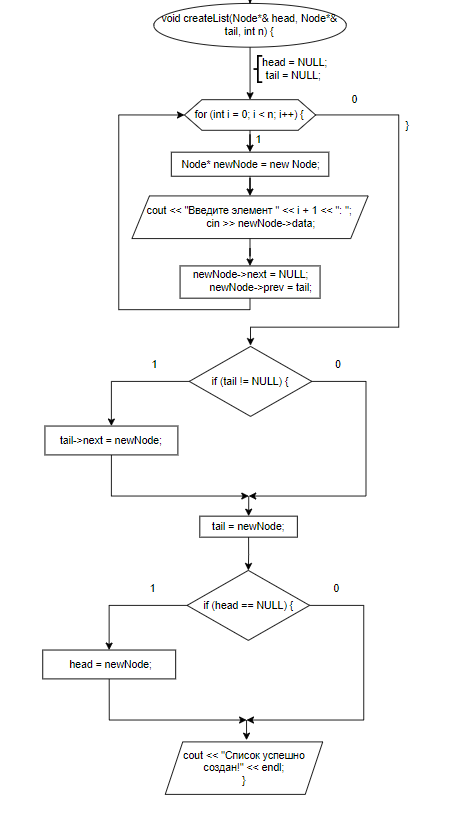
**Алгоритм решения задачи**

1. Создаётся структура struct Node с полями int data (информация, хранящаяся в узле) и Node\* next (указатель на следующий элемент списка). Данная структура является структурой узла.
2. Создаётся функция void createList(). Эта функция сначала просит ввести пользователя размер списка, а затем циклично просит ввести информацию, которая будет записываться в узел списка, а после создаёт указатель на следующий узел.
3. Создаётся функция void addNode() с параметрами. Через цикл находится элемент, предшествующий выбранному, после чего пользователя просят ввести новый элемент списка столько раз, сколько это попросит сделать пользователь. Затем указатель с предыдущего элемента переносится на новый, с нового на указанный, с указанного снова на новый и с нового на следующий.
4. Создаётся функция void deleteNode() с параметрами int k (номер удаляемого элемента). С помощью цикла находится элемент, предшествующий элементу k. Затем указатель предыдущего элемента переносится с удаляемого на следующий.
5. Создаётся функция void printlist(). Через цикл она выводит значения ключа, а затем переносит указатель на следующий элемент.
6. Создаётся функция void writeListToFile() с параметрами string path (ссылка на файл). Создаётся элемент point p (первый элемент списка), а так же указатель на nullptr. Далее создаётся цикл, в котором вся информация записывается в новые узлы, а так же создаются указатели. Цикл завершается, когда заканчивается информация в файле.
7. Создаётся функция void destroyList(), которая удаляет список.
8. Создаётся функция void readListFromFile(). С помощью библиотеки fstream открывается файл по указанной ссылке, и через цикл в него записывается информация из списка. Если файл не удаётся открыть, функция выводит ошибку.
9. Создаётся функция int main(). В ней создаётся список point\* first, который пользователя просят заполнить.

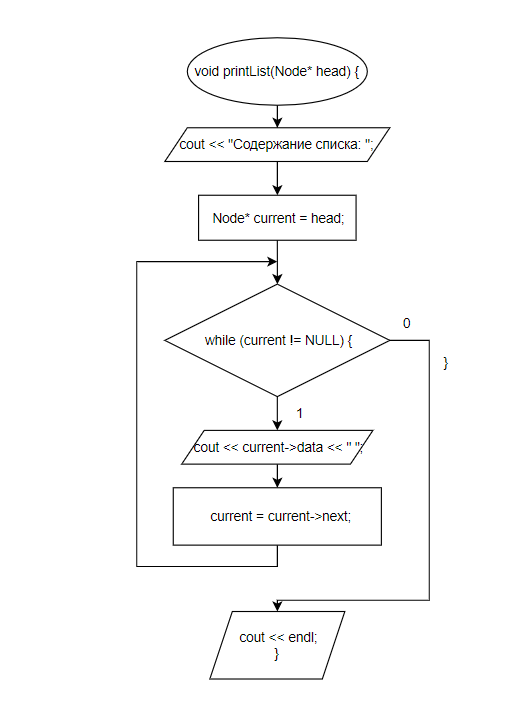
**Блок-схема**



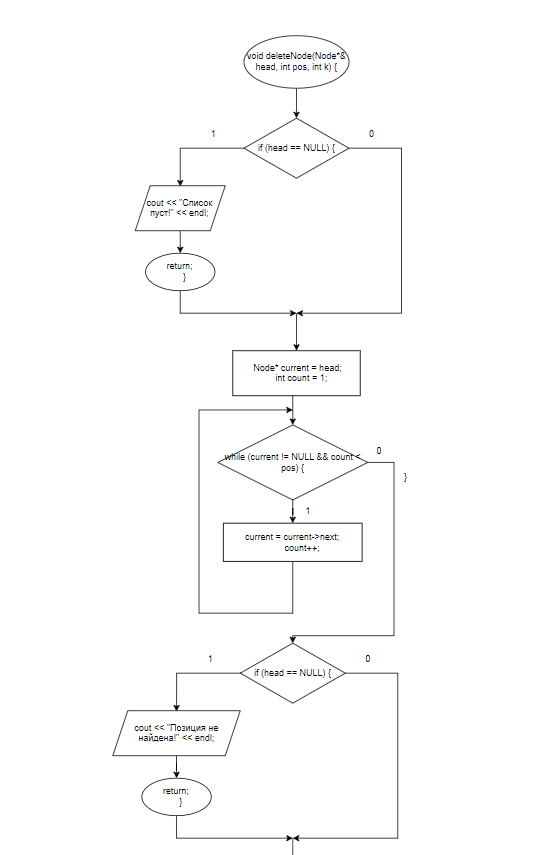
**Рис. 1 – Блок-схема структуры Node**

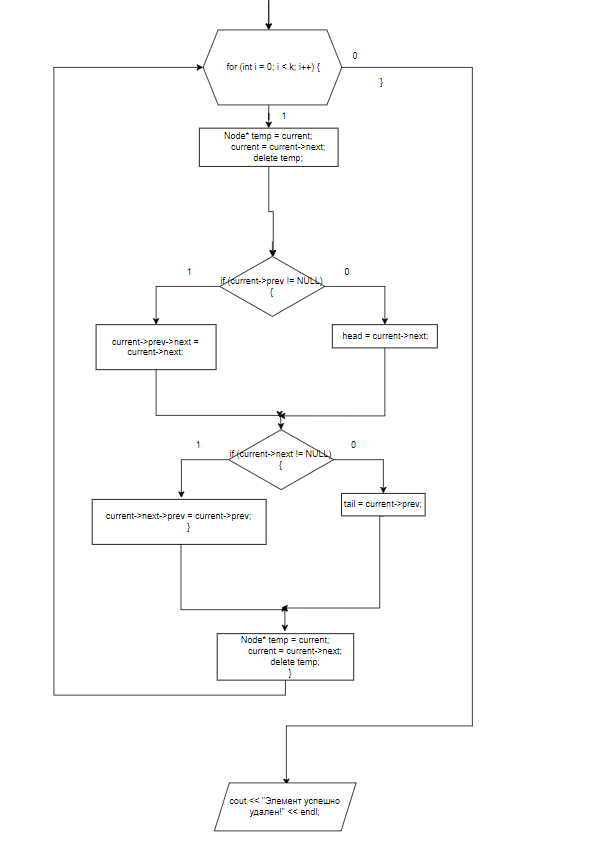
****

**Рис. 2 – Блок-схема функции void createList()**



**Рис. 3 – Блок-схема функции void print\_list()**

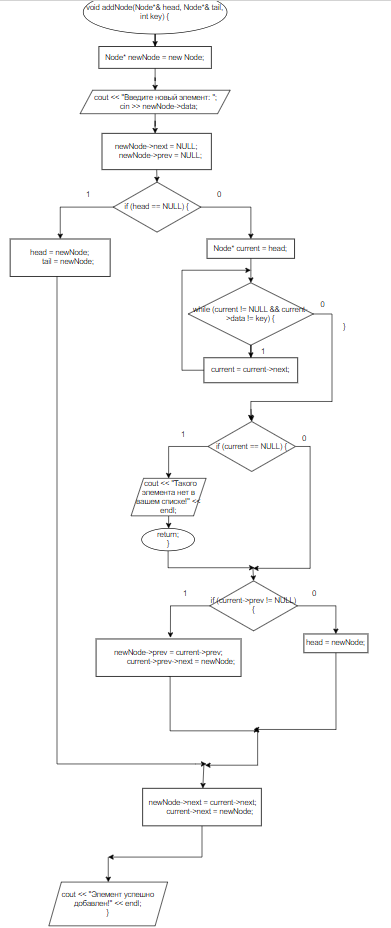




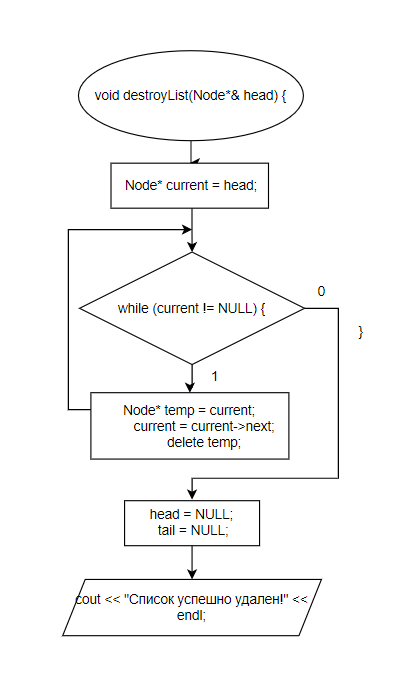
**Рис. 4 – Блок-схема**

**функции**

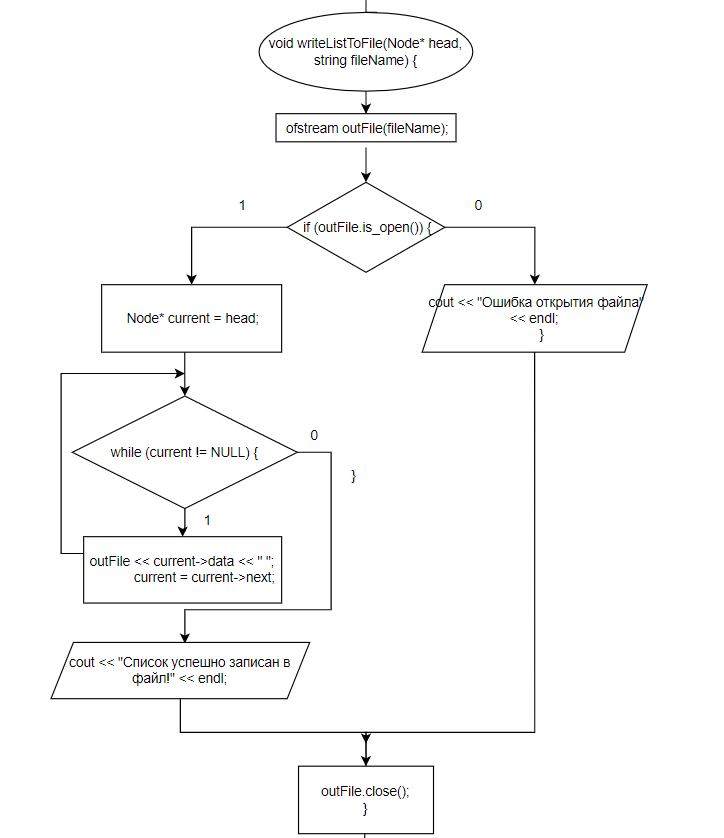
**void deleteNode**

****

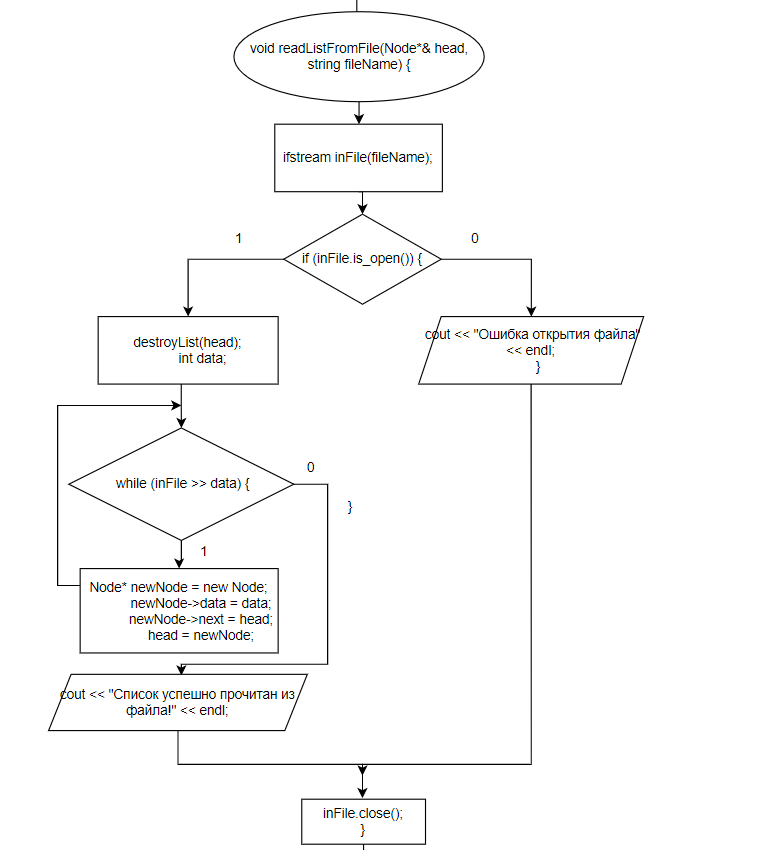
**Рис. 5 – Блок-схема функции void Addnode()**



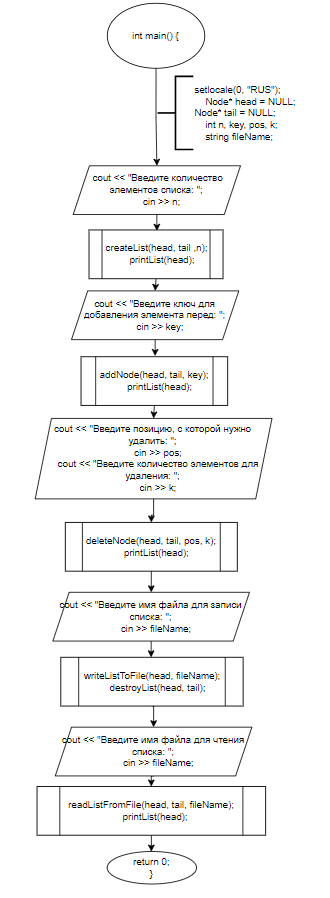
**Рис. 6 – Блок-схема функции void destroyList()**



**Рис. 7 – Блок-схема функции void writeListToFile()**



**Рис. 8 – Блок-схема функции void readListFromFile()**



**Рис. 9 – Блок-схема функции int main()**

**Код программы**

#include <iostream>

#include <fstream>

using namespace std;

struct Node {

int data;

Node\* next;

Node\* prev;

};

void createList(Node\*& head, Node\*& tail, int n) {

head = NULL;

tail = NULL;

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

Node\* newNode = new Node;

cout << "Введите элемент " << i + 1 << ": ";

cin >> newNode->data;

newNode->next = NULL;

newNode->prev = tail;

if (tail != NULL) {

tail->next = newNode;

}

tail = newNode;

if (head == NULL) {

head = newNode;

}

}

cout << "Список успешно создан!" << endl;

}

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

Node\* newNode = new Node;

cout << "Введите новый элемент: ";

cin >> newNode->data;

newNode->next = NULL;

newNode->prev = NULL;

if (head == NULL) {

head = newNode;

tail = newNode;

}

else {

Node\* current = head;

while (current != NULL && current->data != key) {

current = current->next;

}

if (current == NULL) {

cout << "Такого элемента нет в вашем списке!" << endl;

return;

}

if (current->prev != NULL) {

newNode->prev = current->prev;

current->prev->next = newNode;

}

else {

head = newNode;

}

newNode->next = current;

current->prev = newNode;

}

cout << "Элемент успешно добавлен!" << endl;

}

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

if (head == NULL) {

cout << "Список пуст!" << endl;

return;

}

Node\* current = head;

int count = 1;

while (current != NULL && count < pos) {

current = current->next;

count++;

}

if (current == NULL) {

cout << "Позиция не найдена!" << endl;

return;

}

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

if (current->prev != NULL) {

current->prev->next = current->next;

}

else {

head = current->next;

}

if (current->next != NULL) {

current->next->prev = current->prev;

}

else {

tail = current->prev;

}

Node\* temp = current;

current = current->next;

delete temp;

}

cout << "Элемент успешно удален!" << endl;

}

void printList(Node\* head) {

cout << "Содержание списка: ";

Node\* current = head;

while (current != NULL) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

void writeListToFile(Node\* head, string fileName) {

ofstream outFile(fileName);

if (outFile.is\_open()) {

Node\* current = head;

while (current != NULL) {

outFile << current->data << " ";

current = current->next;

}

cout << "Список успешно записан в файл!" << endl;

}

else {

cout << "Ошибка открытия файла" << endl;

}

outFile.close();

}

void destroyList(Node\*& head, Node\*& tail) {

Node\* current = head;

while (current != NULL) {

Node\* temp = current;

current = current->next;

delete temp;

}

head = NULL;

tail = NULL;

cout << "Список успешно удален!" << endl;

}

void readListFromFile(Node\*& head, Node\*& tail, string fileName) {

ifstream inFile(fileName);

if (inFile.is\_open()) {

destroyList(head, tail);

int data;

while (inFile >> data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = NULL;

newNode->prev = tail;

if (tail != NULL) {

tail->next = newNode;

}

tail = newNode;

if (head == NULL) {

head = newNode;

}

}

cout << "Список успешно прочитан из файла!" << endl;

}

else {

cout << "Ошибка открытия файла!" << endl;

}

}

int main()

{

setlocale(0, "RUS");

Node\* head = NULL;

Node\* tail = NULL;

int n, key, pos, k;

string fileName;

cout << "Введите количество элементов списка: ";

cin >> n;

createList(head, tail ,n);

printList(head);

cout << "Введите ключ для добавления элемента перед: ";

cin >> key;

addNode(head, tail, key);

printList(head);

cout << "Введите позицию, с которой нужно удалить: ";

cin >> pos;

cout << "Введите количество элементов для удаления: ";

cin >> k;

deleteNode(head, tail, pos, k);

printList(head);

cout << "Введите имя файла для записи списка: ";

cin >> fileName;

writeListToFile(head, fileName);

destroyList(head, tail);

cout << "Введите имя файла для чтения списка: ";

cin >> fileName;

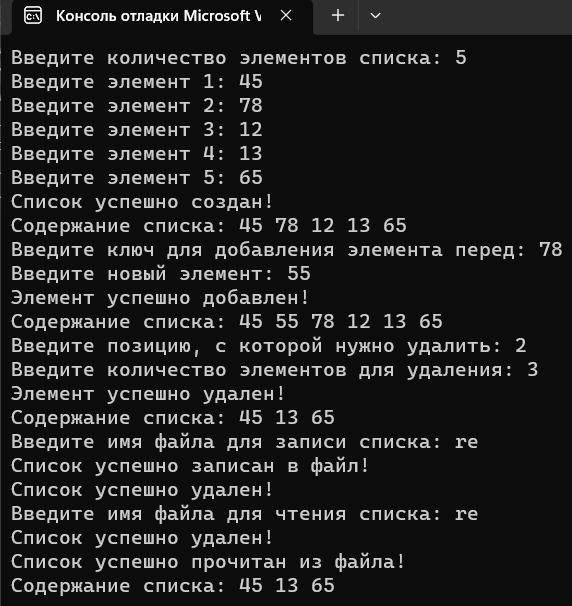
readListFromFile(head, tail, fileName);

printList(head);

return 0;

}

**Результаты работы программы**

****

**Рис. 10 – Вывод консоли решения программы**