Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: «Информационные динамические структуры.

Двунаправленные списки»

Выполнил студент ИВТ-22-2б:

Захаров Дмитрий Сергеевич

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**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

**Алгоритм решения**

1. Создать структуру записи списка
2. Создать функцию создания списка
3. Создать функцию добавления элемента в список
4. Создать функцию удаления элементов из конца списка
5. Создать функцию печати списка
6. Создать функцию записи списка в файл
7. Создать функцию восстановления списка из файла
8. Создать функцию уничтожения списка
9. Создать главную функцию main

**Блок-схема**

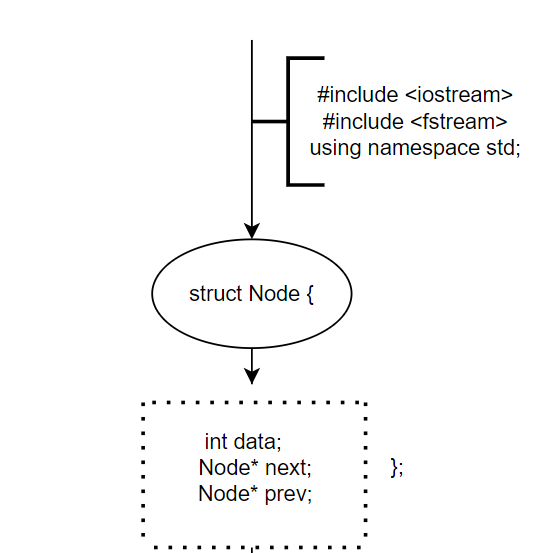


Рис. 1 – Блок-схема структуры Node

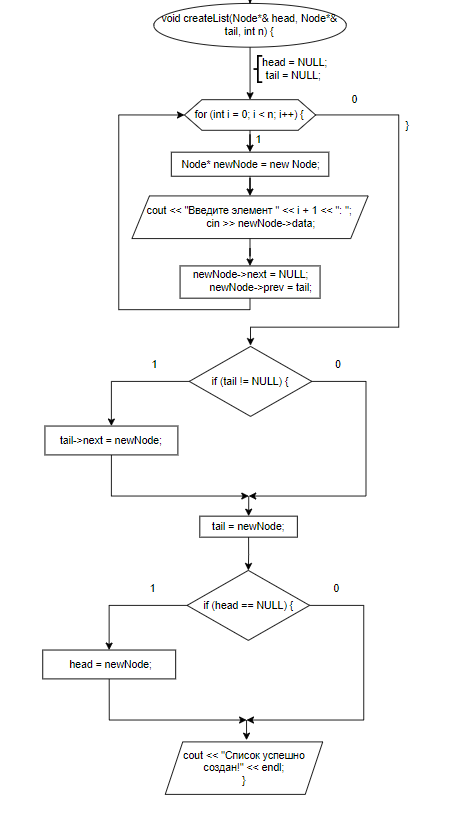


Рис. 2 – Блок-схема функции void createList

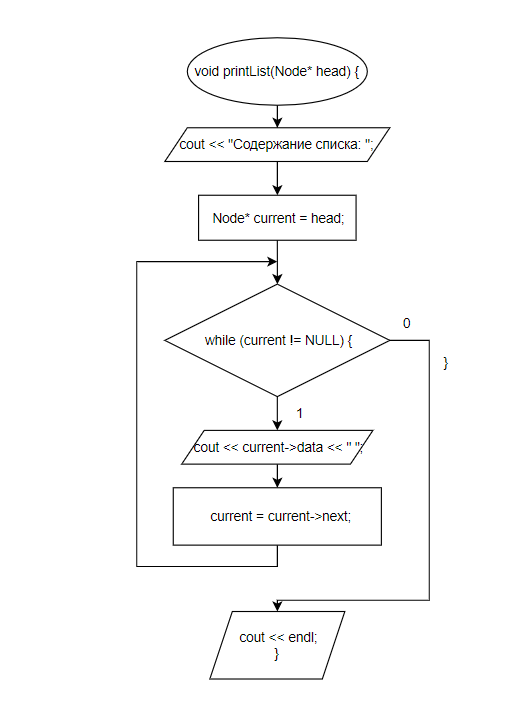
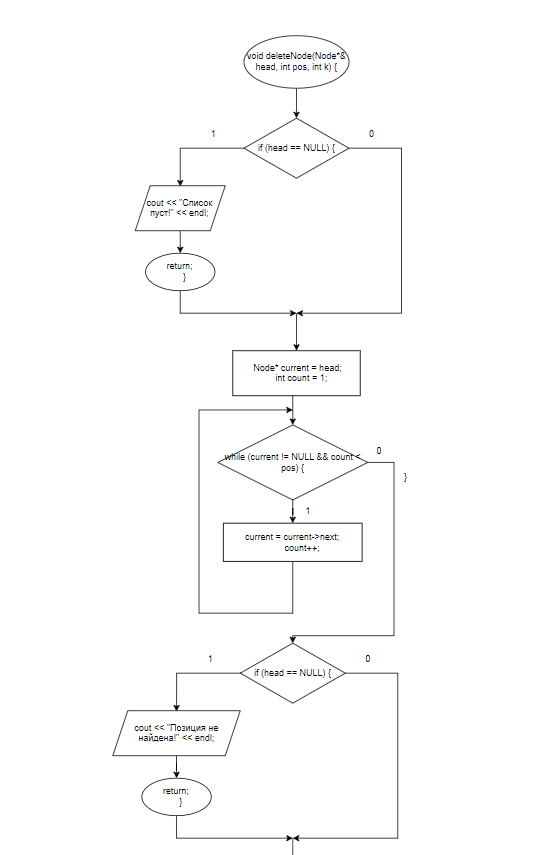


Рис. 3 – Блок-схема функции void print\_list



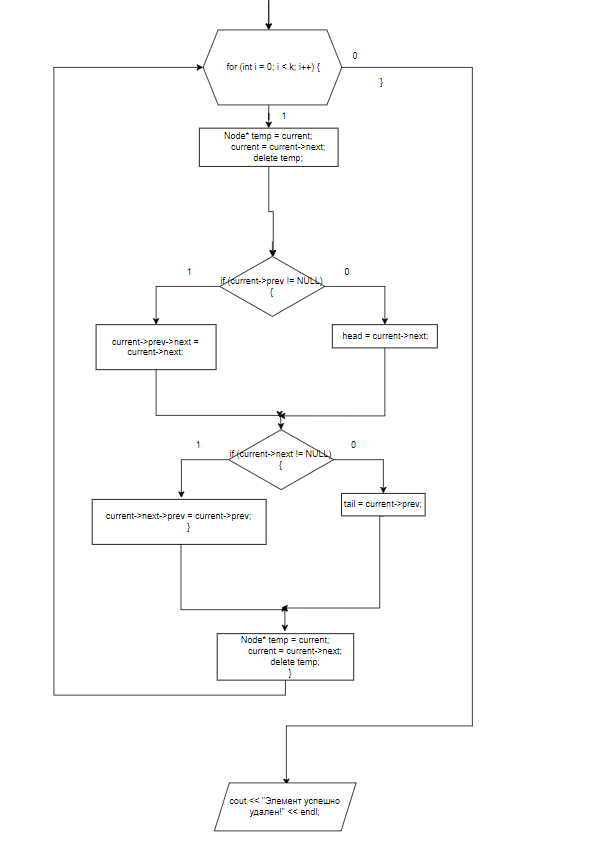


Рис. 4 – Блок-схема

функции

void deleteNode

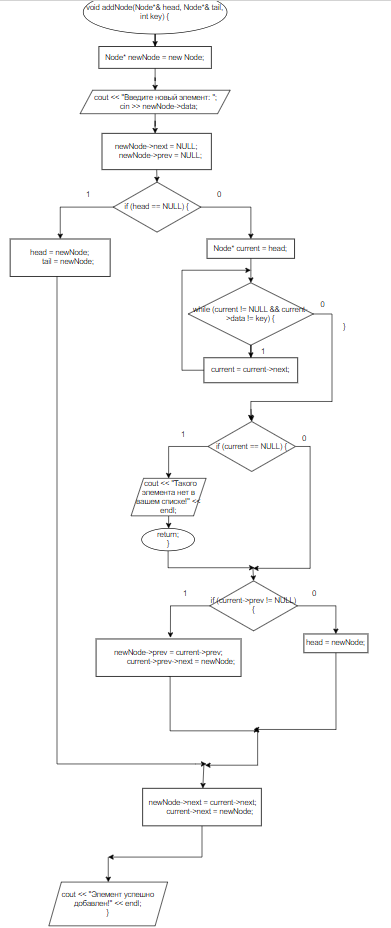


Рис. 5 – Блок-схема функции void Addnode()

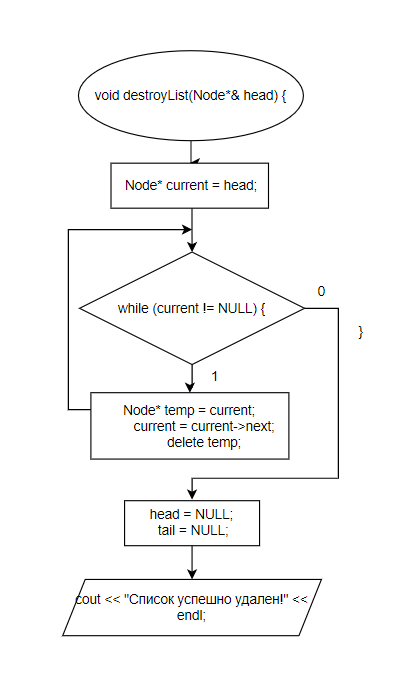


Рис. 6 – Блок-схема функции void destroyList()

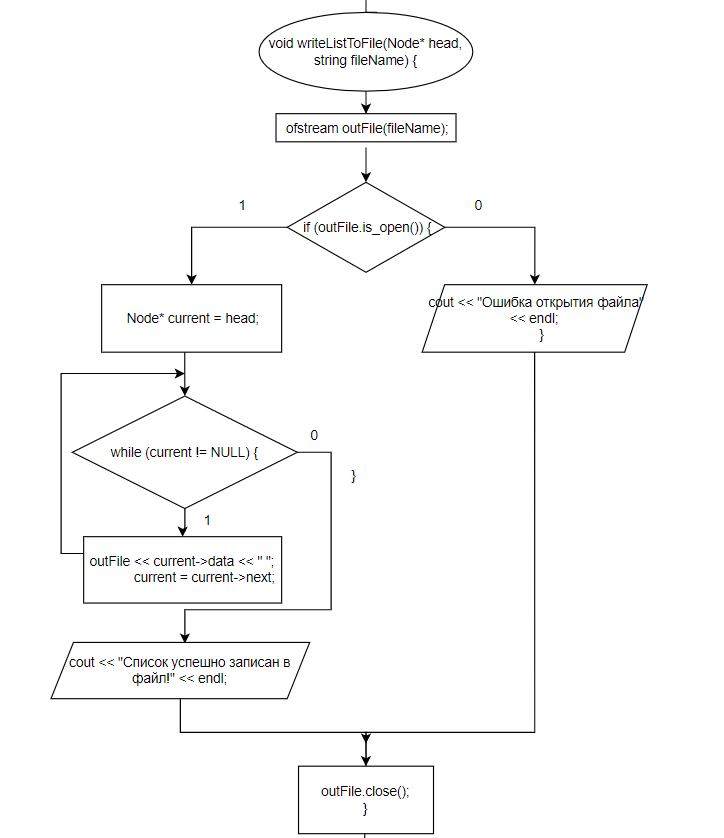


Рис. 7 – Блок-схема функции void writeListToFile

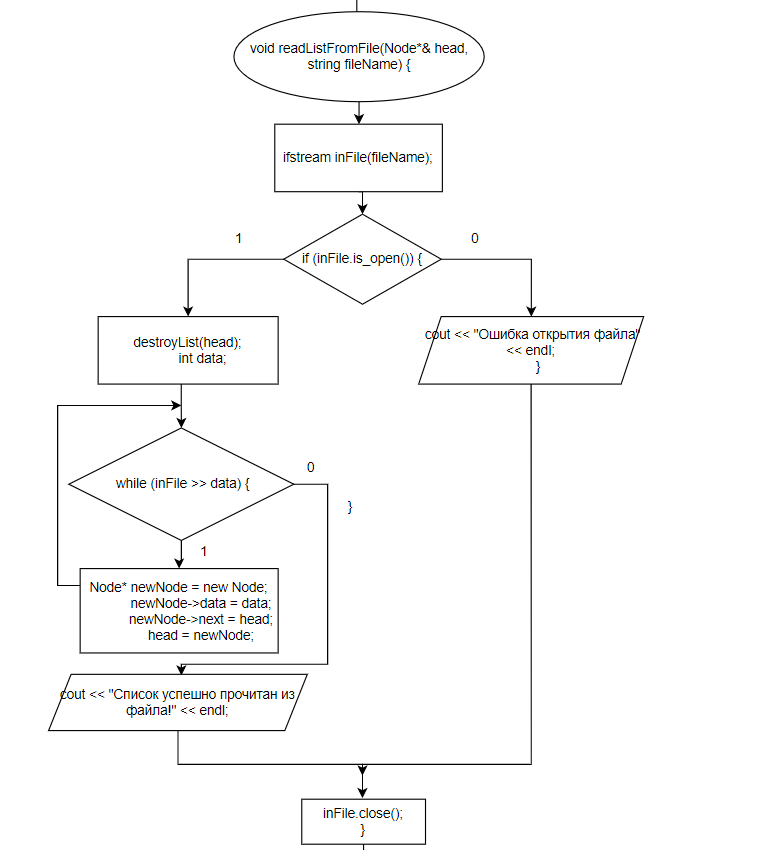


Рис. 8 – Блок-схема функции void readListFromFile

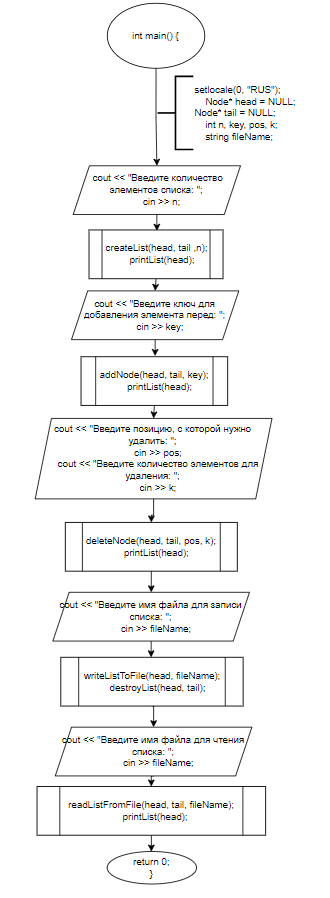


Рис. 9 – Блок-схема функции main

**Код программы**

#include <iostream>

#include <fstream>

#include <ctime>

using namespace std;

// Структура записи списка

struct Node {

int key;

Node\* next;

Node\* prev;

};

// Функция создания списка

Node\* createList(int n) {

Node\* head = nullptr;

Node\* tail = nullptr;

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

cin >> n;

srand(time(NULL));

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

Node\* temp = new Node;

temp->key = rand() % 100;

temp->next = nullptr;

temp->prev = tail;

if (head == nullptr) {

head = temp;

}

else {

tail->next = temp;

}

tail = temp;

}

return head;

}

// Функция добавления элемента в список

void addElement(Node\*& head, int num, int key) {

Node\* temp = new Node;

cout << "Enter an element to add: ";

cin >> key;

cout << "Enter key to add an element: ";

cin >> num;

temp->key = key;

temp->next = nullptr;

temp->prev = nullptr;

if (num == 0) {

temp->next = head;

head->prev = temp;

head = temp;

}

else {

Node\* current = head;

for (int i = 0; i < num - 1; i++) {

current = current->next;

}

temp->next = current->next;

temp->prev = current;

current->next->prev = temp;

current->next = temp;

}

}

// Функция удаления элементов из конца списка

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

Node\* current = head;

cout << "Enter the node of elements to delete: ";

cin >> k;

for (int i = 0; i < k - 1; i++) {

current = current->next;

}

if (current->next == nullptr) {

// Удаляем последний элемент

current->prev->next = nullptr;

delete current;

}

else {

// Удаляем несколько элементов из конца списка

Node\* temp = current->next;

while (temp != nullptr) {

temp = temp->next;

delete current->next;

current->next = temp;

if (temp != nullptr) {

temp->prev = current;

}

}

}

}

// Функция печати списка

void printList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

cout << current->key << " ";

current = current->next;

}

cout << endl;

}

// Функция записи списка в файл

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

ofstream file;

fileName = "fileName.txt";

file.open("fileName.txt");

Node\* current = head;

while (current != nullptr) {

file << current->key << " ";

current = current->next;

}

file.close();

}

// Функция восстановления списка из файла

Node\* loadListFromFile(string fileName) {

Node\* head = nullptr;

Node\* tail = nullptr;

ifstream file;

fileName = "fileName.txt";

file.open("fileName.txt");

int key;

while (file >> key) {

Node\* temp = new Node;

temp->key = key;

temp->next = nullptr;

temp->prev = tail;

if (head == nullptr) {

head = temp;

}

else {

tail->next = temp;

}

tail = temp;

}

file.close();

return head;

}

// Функция уничтожения списка

void destroyList(Node\*& head) {

Node\* current = head;

while (current != nullptr) {

Node\* temp = current->next;

delete current;

current = temp;

}

head = nullptr;

}

int main() {

// Создание списка

Node\* list = createList(10);

// Печать списка

printList(list);

// Добавление элемента в список

addElement(list, 3, 50);

// Печать списка

printList(list);

// Удаление элементов из конца списка

deleteKElements(list, 3);

// Печать списка

printList(list);

}