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: «Однонаправленные списки»

Выполнил студент ИВТ-22-2б:

Афонин Артем Александрович

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

Записи в линейном списке содержат ключевое поле типа string (строка символов). Сформировать однонаправленный список. Удалить элемент с заданным номером. Добавить по К элементов перед и после элемента с заданным ключом.

**Алгоритм решения**

1. Определить структуру узла списка struct Node. В узле содержатся ключевое слово типа string и указатель на следующий узел.

2. Определить структуру однонаправленного списка с указателями на начало и конец списка.

3. Реализовать функцию void addNode() для добавления элементов в конец списка.

4. Реализовать функцию void print\_list() для вывода списка на экран.

5. Реализовать функцию Node\* del\_Node(), в которой создается новый узел, указатель устанавливается на текущий элемент, текущий элемент удаляется и ссылка на текущий элемент перемещается на следующий.

6. Реализовать функцию void add\_Node для добавления элементов справа и слева от ключевого слова. Функция при этом учитывает граничные случаи, такие как вставка перед головным элементом и вставка в конец списка.

7. Реализовать функцию int find\_key\_index(), которая осуществляет поиск индекса позиции узла с заданным ключевым словом в списке и возвращает позицию, если ключевое слово найдено и -1 в противном случае.

8. Реализовать функцию Node\* delete\_list(), которая полностью удаляет все узлы в связанном списке, начиная с узла beg, используя функцию Node\* del\_Node() для удаления узлов.

9. Реализовать функцию void printINfile(), которая записывает информацию из связанного списка, начиная с узла beg, в текстовый файл с указанным путем. Если файл не удалось открыть, выводится сообщение об ошибке.

10. Реализовать функцию Node\* readFROMfile(), которая считывает информацию из текстового файла с указанным путем и создает связанный список, содержащий прочитанные значения. Если файл не удалось открыть, выводится сообщение об ошибке.

11. В функции int main() произвести тестирование всех функций по постановке задачи.

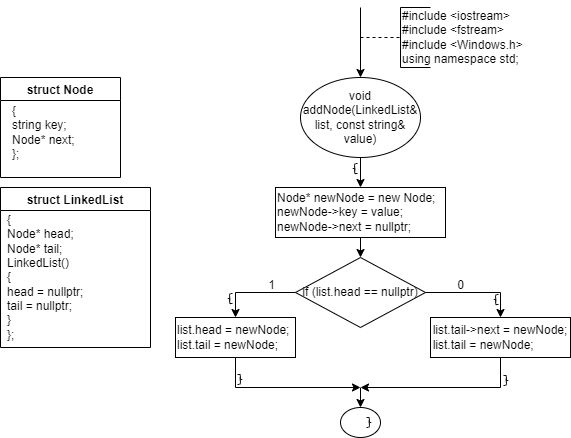
**Блок-схемы**

Рисунок 1 – Структура узла struct Node, структура списка struct LinkedList, функция void addNode()

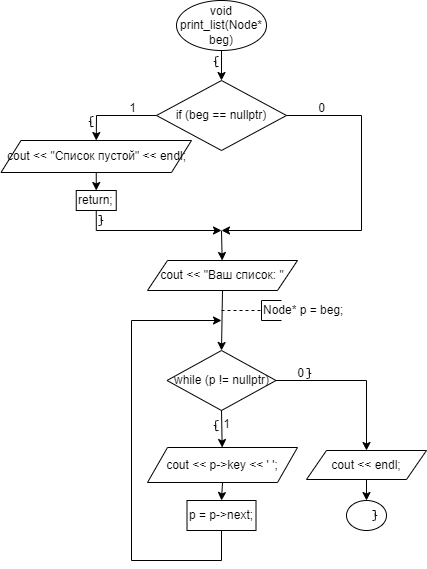
****

Рисунок 2 – Функция void print\_list()

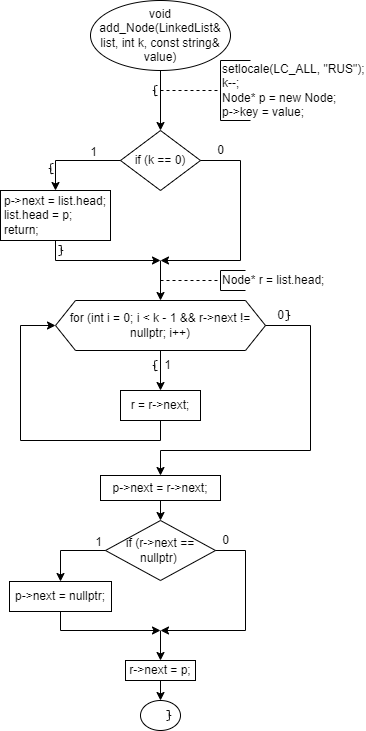


Рисунок 3 – Функция void add\_Node()

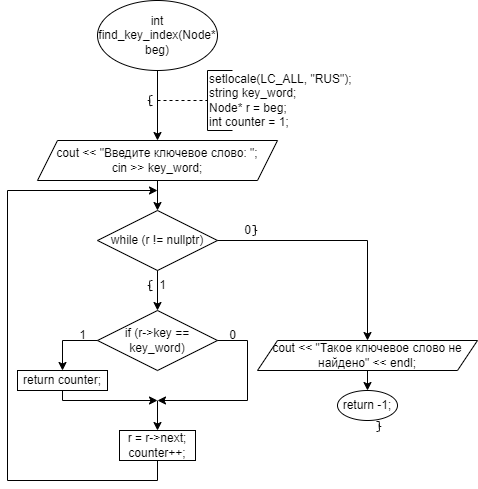


Рисунок 4 – Функция int find\_key\_index()

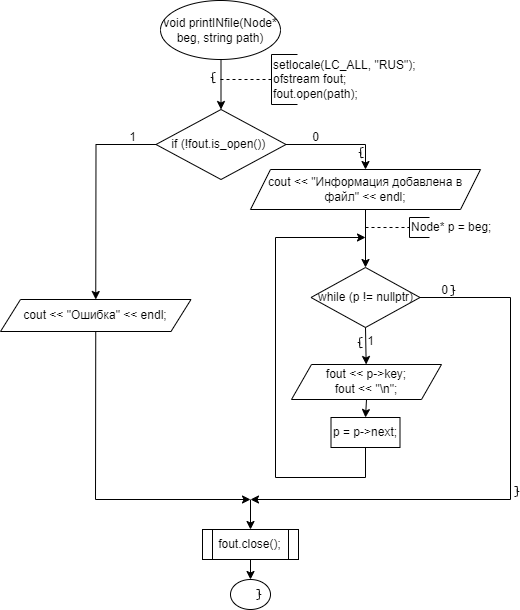


Рисунок 5 – Функция void printINfile()

**Код программы**

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

struct Node

{

string key;

Node\* next;

};

struct LinkedList

{

Node\* head; // указатель на начало списка

Node\* tail; // указатель на конец списка

// Конструктор

LinkedList()

{

head = nullptr;

tail = nullptr;

}

};

// Функция для добавления элемента в конец списка

void addNode(LinkedList& list, const string& value)

{

Node\* newNode = new Node;

newNode->key = value;

newNode->next = nullptr;

if (list.head == nullptr) // если список пустой, новый элемент становится головным и хвостовым

{

list.head = newNode;

list.tail = newNode;

}

else

{

list.tail->next = newNode; // добавляем новый элемент после текущего хвостового элемента

list.tail = newNode; // обновляем указатель на хвостовой элемент

}

}

// Функция для вывода списка

void print\_list(Node\* beg)

{

if (beg == nullptr) // если список пустой

{

cout << "Список пустой" << endl;

return;

}

cout << "Ваш список: ";

Node\* p = beg;

while (p != nullptr) // пока не конец списка

{

cout << p->key << ' ';

p = p->next; // перейти на следующий элемент

}

cout << endl;

}

// Функция для удаления узла

Node\* del\_Node(Node\* beg, int k)

{

setlocale(LC\_ALL, "RUS");

k--;

Node\* p = beg;

if (k == 0)

{

beg = beg->next;

if (beg == nullptr) return nullptr;

delete p;

return beg;

}

for (int i = 0; i < k - 1 && p != nullptr; i++, p = p->next);

if (p == nullptr || p->next == nullptr) return beg;

Node\* r = p->next; // встать на удаляемый элемент

p->next = r->next; // изменить ссылку

delete r;

r = p->next; // встать на следующий

return beg;

}

// Функция для добавления элементов слева и справа от ключевого слова

void add\_Node(LinkedList& list, int k, const string& value)

{

setlocale(LC\_ALL, "RUS");

k--;

Node\* p = new Node;

p->key = value;

if (k == 0)

{

p->next = list.head; // добавить перед головным элементом

list.head = p; // обновить указатель на головной элемент

return;

}

Node\* r = list.head; // встать на начало списка

for (int i = 0; i < k - 1 && r->next != nullptr; i++) {

r = r->next;

}

p->next = r->next; // связать r с концом списка

// если элемент не последний, то связать конец списка с r

if (r->next == nullptr)

p->next = nullptr;

r->next = p;

}

int find\_key\_index(Node\* beg)

{

setlocale(LC\_ALL, "RUS");

string key\_word;

Node\* r = beg;

int counter = 1;

cout << "Введите ключевое слово: ";

cin >> key\_word;

while (r != nullptr)

{

if (r->key == key\_word)

return counter;

r = r->next;

counter++;

}

cout << "Такое ключевое слово не найдено" << endl;

return -1; // возвращаем -1 для обозначения отсутствия найденного ключевого слова

}

Node\* delete\_list(Node\* beg)

{

setlocale(LC\_ALL, "RUS");

Node\* p = beg;

int counter = 0; // Изменяем начальное значение counter на 0

while (p != nullptr) // Изменяем условие цикла на p != nullptr

{

p = p->next;

counter++;

}

for (int i = counter; i > 1; i--)

{

beg = del\_Node(beg, i); // Используем функцию del\_Node() для удаления узлов

}

beg = del\_Node(beg, 1); // Удаляем первый узел

return beg;

}

void printINfile(Node\* beg, string path) {

setlocale(LC\_ALL, "RUS");

ofstream fout;

fout.open(path);

if (!fout.is\_open())

cout << "Ошибка" << endl;

else {

cout << "Информация добавлена в файл" << endl;

Node\* p = beg;

while (p != nullptr) {

fout << p->key;

fout << "\n";

p = p->next;

}

}

fout.close();

}

//Функция для восстановления списка из файла

Node\* readFROMfile(string path) {

setlocale(LC\_ALL, "RUS");

ifstream fin;

fin.open(path);

Node\* beg = nullptr;

if (!fin.is\_open())

cout << "Ошибка" << endl;

else {

cout << "Информация была восстановлена из файла" << endl;

string word;

Node\* p = nullptr;

Node\* r = nullptr;

fin >> word;

while (!fin.eof()) {

if (p == nullptr) {

p = new(Node);

p->key = word;

p->next = nullptr;

beg = p;

}

else {

r = new(Node);

r->key = word;

p->next = r;

r->next = nullptr;

p = r;

}

fin >> word;

}

}

fin.close();

return beg;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int c;

string value;

LinkedList list; // Создаем экземпляр связного списка

cout << "Введите количество элементов списка: "; cin >> c;

for (int i = 1; i <= c; i++) {

cout << "Введите элемент спика: "; cin >> value;

addNode(list, value);

}

Node\* first = list.head; // Устанавливаем начало списка на головной элемент

print\_list(first); // Выводим список на экран

cout << "Номер элемента, который вы хотите удалить: ";

int n;

cin >> n;

first = del\_Node(first, n); // Удаляем элемент из списка

print\_list(first); // Выводим обновленный список на экран

int num = find\_key\_index(first); // Ищем индекс ключевого слова в списке

cout << "Введите количество, добавляемых слева и справа элементов: ";

int k;

string add;

cin >> k;

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

cout << "Введите добавлемяый элемент: "; cin >> add;

add\_Node(list, num + i, add); // Добавляем элементы слева от ключевого слова

}

num += k;

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

cout << "Введите добавлемяый элемент: "; cin >> add;

add\_Node(list, num + i + 1,add); // Добавляем элементы справа от ключевого слова

}

print\_list(list.head); // Выводим обновленный список на экран

cout << endl;

printINfile(first, "output.txt");

list.head = delete\_list(first);

cout << "Список удалён: "; print\_list(list.head);

Node\* recovery = readFROMfile("output.txt");

list.head = recovery;

print\_list(list.head);

return 0;

}

**Вывод программы**

