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: «Двунаправленные 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 указатели на следующий и на предыдущий узлы.

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(), которая удаляет узел с заданным номером из связанного списка. Если указанный индекс находится за пределами списка или узел является единственным элементом списка, функция возвращает указатель на головной узел без изменений. В противном случае, функция удаляет узел, обновляет ссылки на предыдущий и следующий узлы, освобождает память, занимаемую удаляемым узлом, и возвращает указатель на головной узел после удаления.

6. Реализовать функцию void add\_Node(), которая добавляет новый в связанный список перед указанным номером. Если указанный номер равен 1, то новый узел будет добавлен перед головным элементом списка. Если указанный номер больше длины списка, функция выведет сообщение об ошибке. В остальных случаях, функция создаст новый узел, установит ссылки на предыдущий и следующий элементы, обновит указатели на предыдущий и следующий элементы у соседних узлов и добавит новый узел в список.

7. Реализовать функцию int find\_key\_index(), которая осуществляет поиск номера узла с указанным ключевым словом. Функция запрашивает ввод ключевого слова от пользователя, затем проходит по всем узлам списка, сравнивая ключевое слово с ключами узлов. Если ключевое слово найдено, функция возвращает индекс этого узла в списке. Если ключевое слово не найдено, функция выводит сообщение об ошибке и возвращает -1.

8. Реализовать функцию Node\* delete\_list()Функция delete\_list, которая осуществляет удаление всех узлов в связанном списке. Функция сначала подсчитывает количество узлов в списке, пройдя по всем узлам с помощью цикла. Затем, с использованием функции del\_Node(), функция последовательно удаляет все узлы, начиная с последнего и до первого, в цикле, передавая соответствующий индекс каждого узла в функцию del\_Node(). В результате, все узлы списка удаляются, и функция возвращает указатель на новый начальный узел списка после удаления всех узлов.

9. В функции int main() провести тестирование всех функций по постановке задачи.

**Блок-схемы**

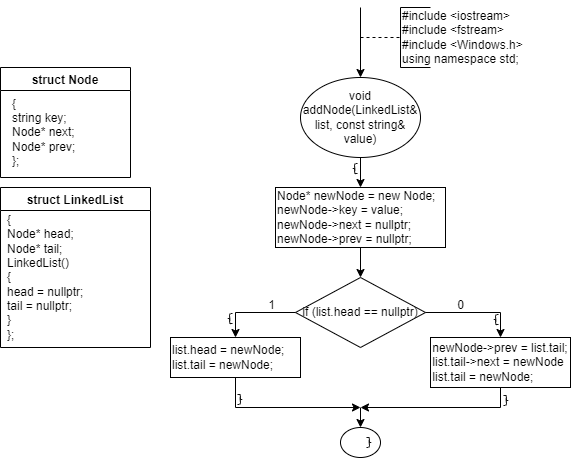
****

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

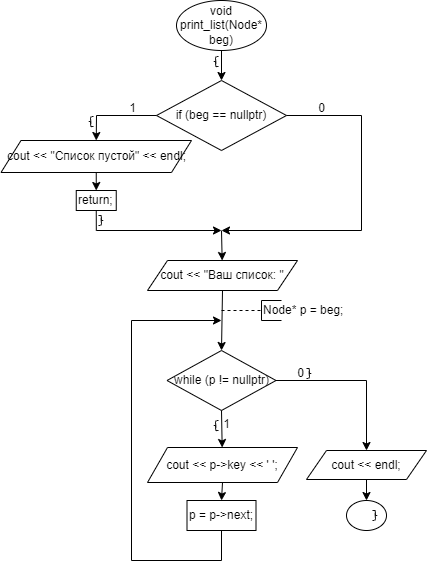
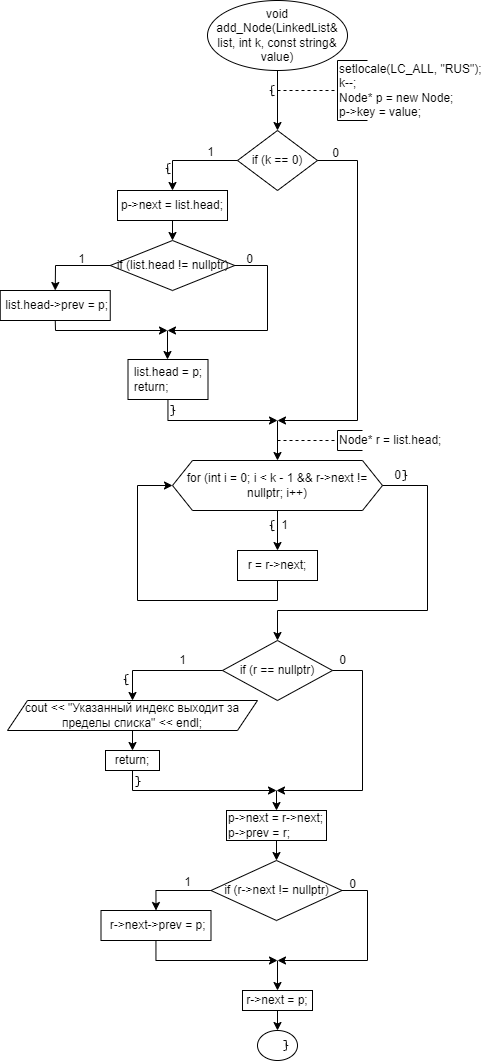


Рисунок 2 – Функция void print\_list()

Рисунок 3 – Функция void add\_Node()

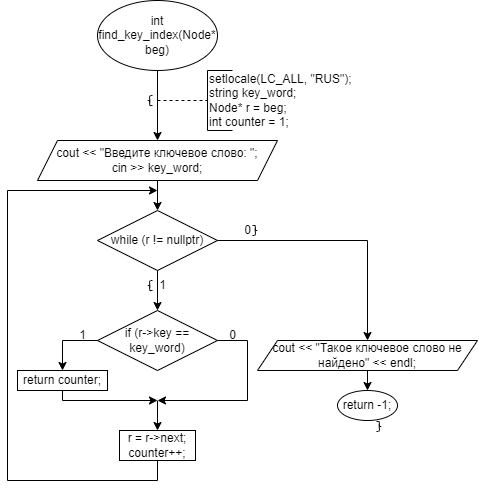


Рисунок 4 – Функция int find\_key\_index()

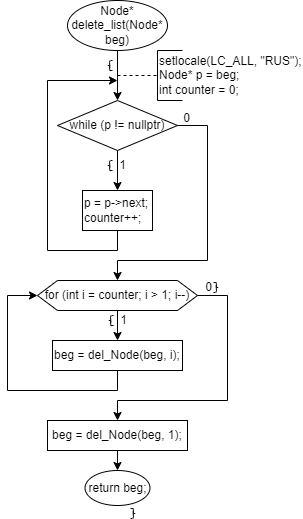


Рисунок 5 – Функция Node\* delete\_list()

**Код программы**

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

struct Node

{

string key;

Node\* next;

Node\* prev; //Указатель на предыдущий элемент

};

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;

newNode->prev = nullptr; //Инициализируем указатель на предыдущий элемент

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

{

list.head = newNode;

list.tail = newNode;

}

else

{

newNode->prev = list.tail; //Указываем на предыдущий элемент

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

if (r->next != nullptr) //Если удаляемый элемент не последний

r->next->prev = p; //Обновляем указатель на предыдущий элемент

delete r;

r = p->next; //Встать на следующий

return beg;

}

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

if (list.head != nullptr)

list.head->prev = p; //Установить указатель на предыдущий элемент для головного элемента

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 != nullptr; i++, r = r->next);

if (r == nullptr) {

cout << "Указанный индекс выходит за пределы списка" << endl;

return;

}

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

p->prev = r; //Связать p с r в обратном направлении

if (r->next != nullptr)

r->next->prev = p; //Установить указатель на предыдущий элемент для следующего элемента после r

r->next = p;

}

//Функция для поиска индекса ключевого слова

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

{

setlocale(LC\_ALL, "RUS");

cout << "Введите ключевое слово: ";

string key\_word;

cin >> key\_word;

Node\* r = beg;

int counter = 1;

while (r != nullptr)

{

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

return counter;

r = r->next;

counter++;

}

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

return -1;

}

//Функция для удаления списка

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

{

setlocale(LC\_ALL, "RUS");

Node\* p = beg;

int counter = 0;

while (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;

}

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;

return 0;

}

**Вывод программы**

