МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



**ОТЧЁТ**

**по лабораторной работе № 4-5**

**«Линейные списки*»***

по дисциплине:

**«Программирование»**

|  |  |
| --- | --- |
| Выполнил(а):  Дашкин С.М. Группа АТ-34 | Проверил:  Кухто А.В. |

Новосибирск

2024

**ЦЕЛЬ РАБОТЫ:**

**Вариант 6.**

 Разработайте программу для:

1. Создания элементов линейного списка (односвязный нециклический, хранит строки размерностью не более 5 символов) в динамической памяти, ввода значений списка из файла (текстовый; данные представлены в виде набора хранимых строк), установки связей между элементами.

2. Выполнения действий над списком (Вставка элемента в список – новые элементы добавляются с сохранением упорядоченности. Удаление элементов – по заданной подстроке. Поиск элемента - по вводимой подстроке. Распечатка всех элементов (на экран)) (в любое время и в любой последовательности, т.е. выбор действий – по меню).

**ХОД РАБОТЫ:**

1. Открытие файла «input.txt» для чтения. Если файл не существует, он создается и затем открывается для чтения.

2. Чтение строк из файла «input.txt» и вставка их в упорядоченный список с помощью функции insertNode.

3. Закрытие файла «input.txt».

4. Отображение меню действий для пользователя.

5. Получение выбора пользователя.

6. В зависимости от выбора пользователя выполняются следующие действия:

- Вставка нового элемента в список.

- Удаление элемента из списка по заданной подстроке с помощью функции deleteNodeBySubstring.

- Поиск элемента в списке по заданной подстроке с помощью функции searchNodeBySubstring.

- Печать списка с помощью функции printList.

- Выход из программы.

7. Повторение шага 5-6, пока пользователь не выберет выход из программы.

8. Освобождение памяти, выделенной под элементы списка с помощью функции freeList.

9. Завершение программы.

**ЛИСТИНГ:**

#define \_CRT\_SECURE\_NO\_DEPRECATE

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Структура для представления элемента списка

struct Node {

char data[6]; // Хранит строки размерностью не более 5 символов

struct Node\* next; // Указатель на следующий элемент списка

};

// Функция для вставки элемента в упорядоченный список

void insertNode(struct Node\*\* head, char newData[]) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

strcpy(newNode->data, newData);

if (\*head == NULL || strcmp(newNode->data, (\*head)->data) < 0) {

newNode->next = \*head;

\*head = newNode;

}

else {

struct Node\* current = \*head;

while (current->next != NULL && strcmp(newNode->data, current->next->data) > 0) {

current = current->next;

}

newNode->next = current->next;

current->next = newNode;

}

}

// Функция для удаления элемента из списка по заданной подстроке

void deleteNodeBySubstring(struct Node\*\* head, char substring[]) {

struct Node\* temp = \*head;

struct Node\* prev = NULL;

while (temp != NULL) {

if (strstr(temp->data, substring) != NULL) {

if (prev == NULL) {

\*head = temp->next;

}

else {

prev->next = temp->next;

}

free(temp);

temp = prev != NULL ? prev->next : \*head;

}

else {

prev = temp;

temp = temp->next;

}

}

}

// Функция для поиска элемента по вводимой подстроке

void searchNodeBySubstring(struct Node\* head, char substring[]) {

int found = 0;

while (head != NULL) {

if (strstr(head->data, substring) != NULL) {

printf(“Element found: %s\n”, head->data);

found = 1;

}

head = head->next;

}

if (!found) {

printf(“Element s podstrokoy \”%s\” not found.\n”, substring);

}

}

// Функция для печати списка

void printList(struct Node\* head) {

while (head != NULL) {

printf(“%s “, head->data);

head = head->next;

}

printf(“\n”);

}

// Функция освобождения памяти

void freeList(struct Node\* head) {

struct Node\* temp;

while (head != NULL) {

temp = head;

head = head->next;

free(temp);

}

}

int main() {

struct Node\* head = NULL;

FILE\* inputFile = fopen(“input.txt”, “r”);

if (inputFile == NULL) {

// Создание файла, если его нет

inputFile = fopen(“input.txt”, “w”);

fclose(inputFile);

inputFile = fopen(“input.txt”, “r”);

if (inputFile == NULL) {

printf(“Error.\n”);

return 1;

}

char buffer[6];

while (fscanf(inputFile, “%5s”, buffer) == 1) {

insertNode(&head, buffer);

}

fclose(inputFile);

int choice;

char searchSubstr[6];

char deleteSubstr[6];

do {

printf(“\nMenu:\n”);

printf(“1. Paste element\n”);

printf(“2. Delete element po podstroke\n”);

printf(“3. Found element po podstroke\n”);

printf(“4. Print spisok\n”);

printf(“5. Exit\n”);

printf(“Chose action: “);

scanf(“%d”, &choice);

switch (choice) {

case 1:

printf(“Vvedite new element: “);

scanf(“%5s”, buffer);

insertNode(&head, buffer);

break;

case 2:

printf(“Vvedite podstroku for delete: “);

scanf(“%5s”, deleteSubstr);

deleteNodeBySubstring(&head, deleteSubstr);

break;

case 3:

printf(“Vvedite podstroku for found: “);

scanf(“%5s”, searchSubstr);

searchNodeBySubstring(head, searchSubstr);

break;

case 4:

printf(“Spisok: “);

printList(head);

break;

case 5:

break;

default:

printf(“Error. Try again\n”);

}

} while (choice != 5);

// Освобождение памяти

freeList(head);

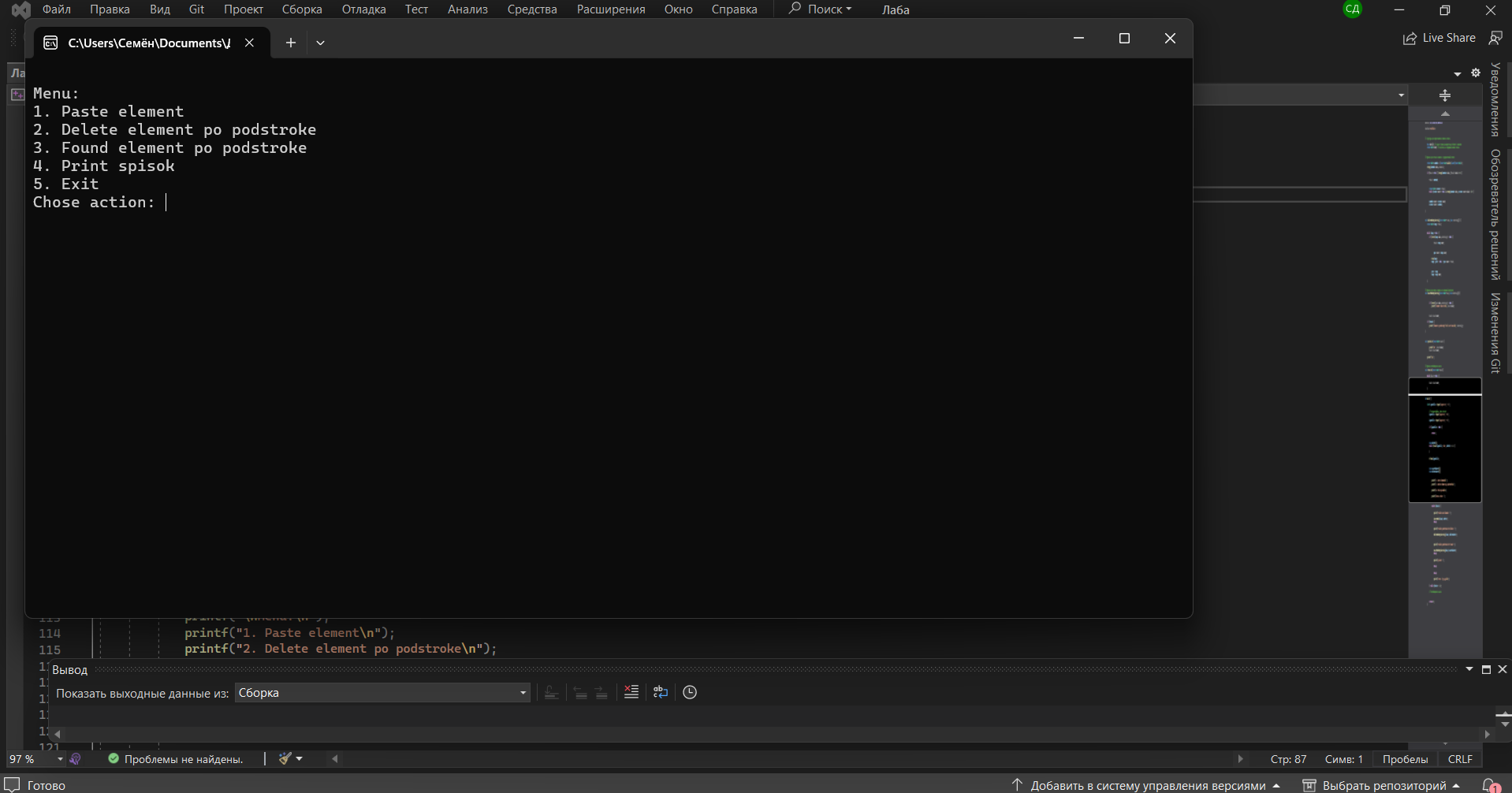
return 0;

}

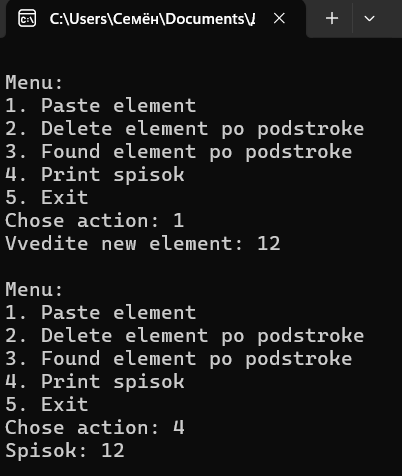
}

**ТЕСТИРОВАНИЕ:**

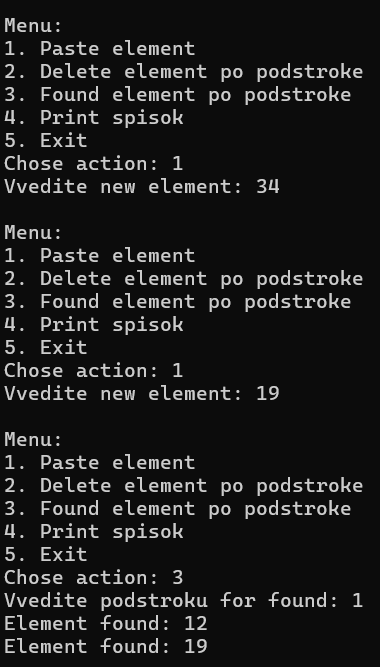
Запуск программы:



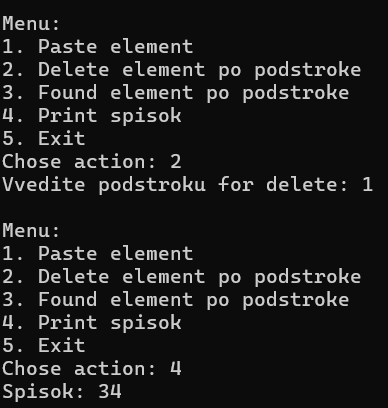
Выберем действие 1 и добавим элемент “12”, а после выведем наш список (действие 4), который будет состоять из элемента “12”.

****

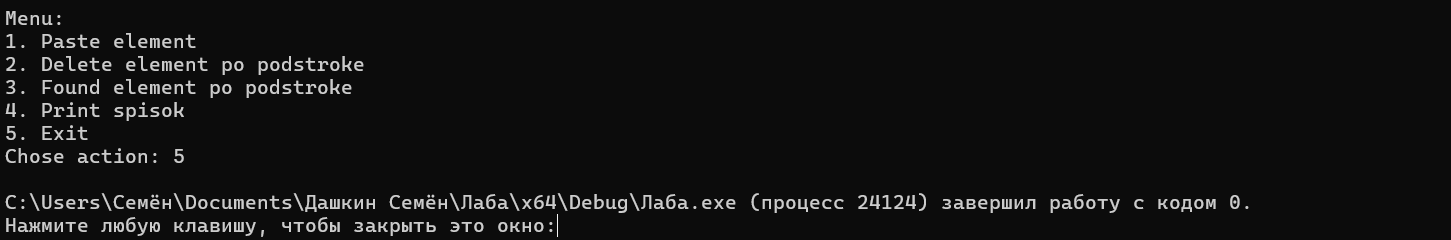
Теперь добавим в наш список еще два элемента – “34” и “19”, и выполним поиск элемента (действие 3) по подстроке “1”:



Далее удалим элементы по подстроке “1” (действие 2) и выведем список:

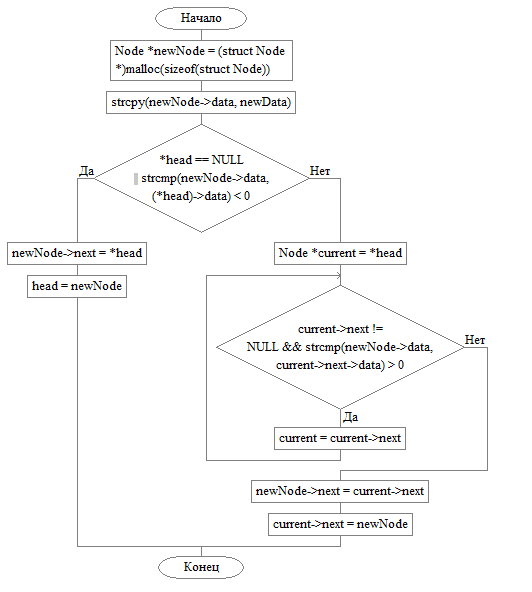


Выйдем из программы(действие 5):

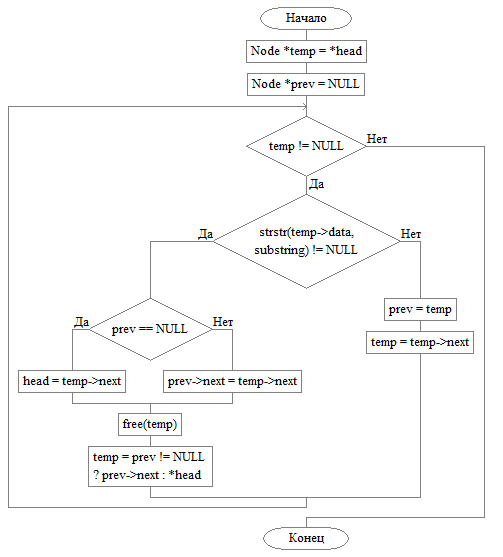


**БЛОК СХЕМЫ:**

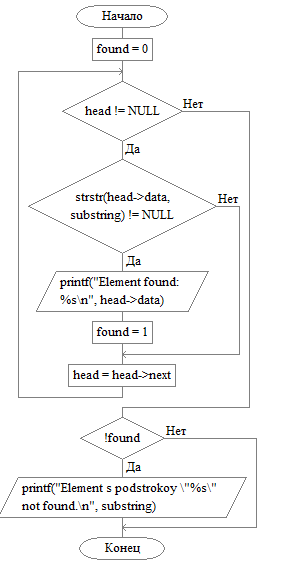
Блок-схема функции insertNode:



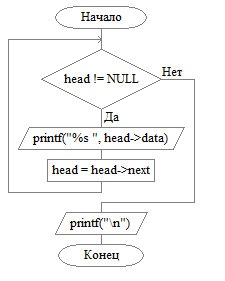
Блок-схема функции deleteNodeBySubstring:



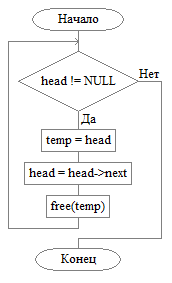
Блок-схема функции searchNodeBySubstring:



Блок-схема функции printList:



Блок-схема функции freeList:



Блок-схема функции main:

