Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Отчет №14

По лабораторной работе

Тема:«Динамические структуры данных: списки и деревья»

Выполнил:

Студ. гр.ИИ-23

Романюк А.П.

Проверил:

Брест 2023

Вариант №8

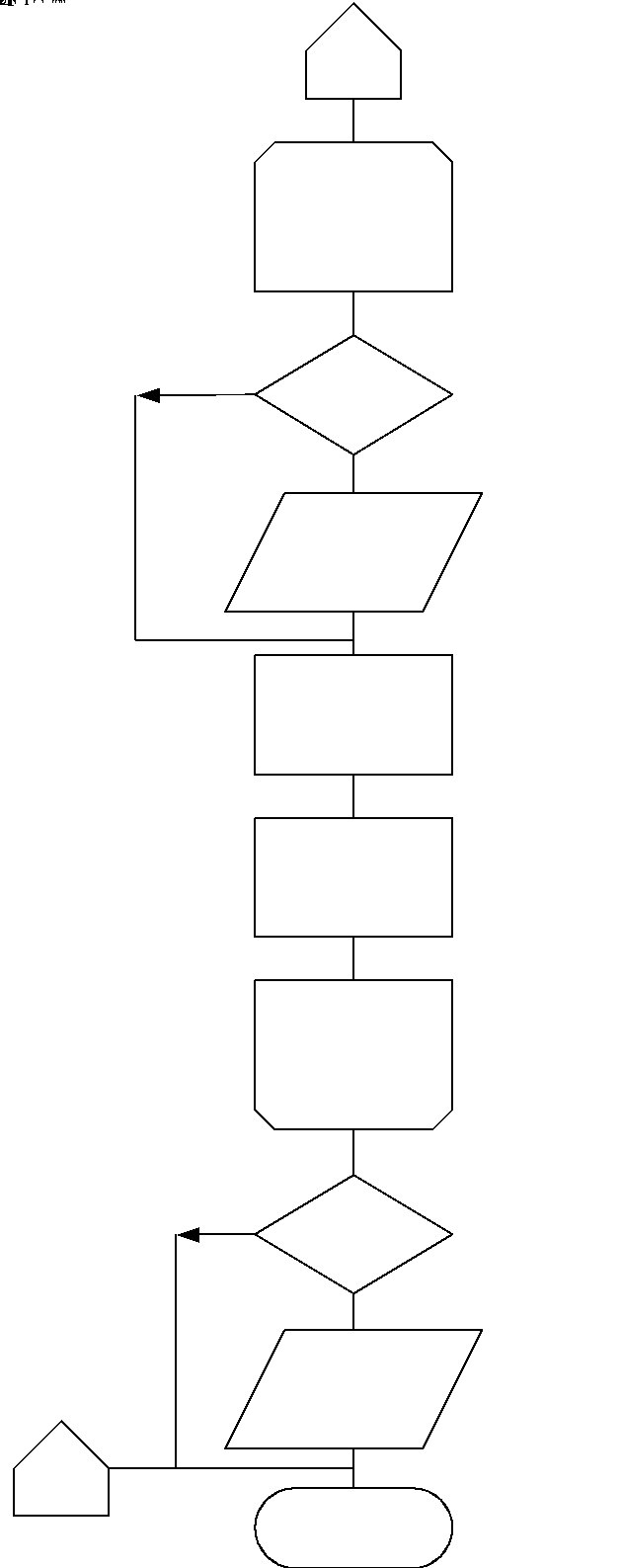
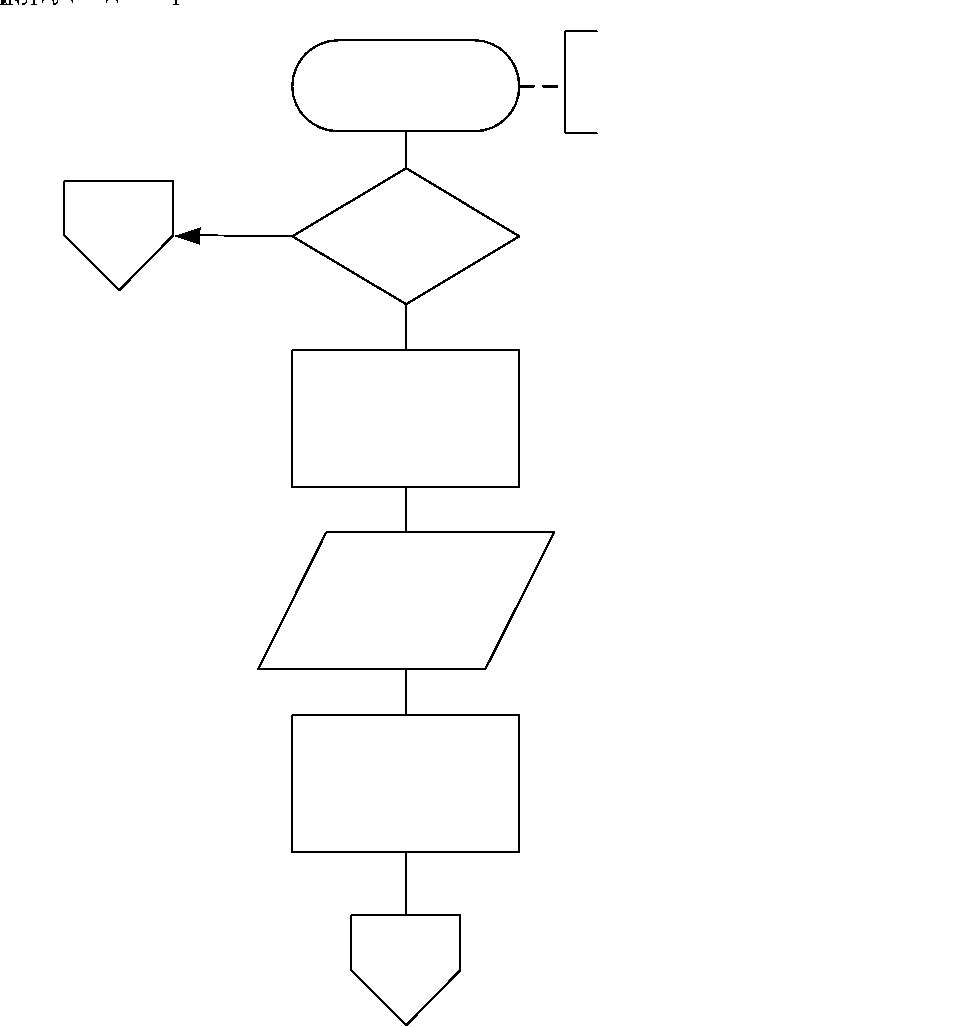
Цель работы: приобретение навыков работы с динамической памятью и указателями на С/C++; изучение принципов работы с динамическими структурами данных: списками и деревьями.

Текст задания:

Общие требования: в начале программы вывести задание; в процессе работы выводить подсказки пользователю (что ему нужно ввести, чтобы продолжить выполнение программы). Основные алгоритмы, ввод/вывод списков реализовать в виде функций с необходимыми параметрами. После работы программы вся динамически выделенная память должна быть освобождена. Взаимодействие с пользователем организовать в виде простого меню, обеспечивающего возможность переопределения исходных данных и завершение работы программы.

8. Написать программу, обеспечивающую работу с однонаправленным нециклическим списком: добавление/удаление элементов в голову, просмотр списка, инверсию списка

Блок-схема:

****

Текст программы:

**#include <iostream>**

**using namespace std;**

**struct list {**

**int data;**

**list\* next;**

**};**

**void menu(list\* head);**

**void create\_list(list\*& head);**

**void create\_node(list\*& head);**

**void add(list\*& head);**

**void del(list\*& head);**

**void output(list\* head);**

**void reverce(list\*& head);**

**bool empty(list\* head);**

**int main() {**

**list\* head;**

**head = NULL;**

**menu(head);**

**}**

**void menu(list \*head) {**

**int choise;**

**while (true) {**

**cout << "выберите действие:\n";**

**cout << "1.создание списка\n";**

**cout << "2.добавление элемента\n";**

**cout << "3.удаление элемента\n";**

**cout << "4.просмотр списка\n";**

**cout << "5.инверсия списка\n";**

**cout << "6.выход\n";**

**cin >> choise;**

**switch (choise) {**

**case 1:**

**system("cls");**

**create\_list(head);**

**break;**

**case 2:**

**system("cls");**

**add(head);**

**break;**

**case 3:**

**system("cls");**

**del(head);**

**break;**

**case 4:**

**system("cls");**

**output(head);**

**break;**

**case 5:**

**system("cls");**

**reverce(head);**

**break;**

**case 6:**

**return;**

**break;**

**}**

**}**

**}**

**void create\_list(list\*&head) {**

**head = NULL;**

**cout << "введите колво элементов списка\n";**

**int n;**

**cin >> n;**

**cout << "введите "<<n<<" элементов списка\n";**

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

**create\_node(head);**

**}**

**if (n > 0) {**

**output(head);**

**}**

**}**

**void create\_node(list\*& head) {**

**list \*new\_node = new list;**

**cin >> new\_node->data;**

**new\_node->next = NULL;**

**list\* q = head;**

**if (q == NULL) {**

**new\_node->next = head;**

**head = new\_node;**

**}**

**else {**

**while (q->next) {**

**q = q->next;**

**}**

**new\_node->next = q->next;**

**q->next = new\_node;**

**}**

**}**

**void del(list\*& head) {**

**if (empty(head) == 1) {**

**output(head);**

**cout << "введите элемент для удаления";**

**int d;**

**cin >> d;**

**list\* q = head;**

**bool find = false;**

**while (q) {**

**if (d == q->data) {**

**find = true;**

**if (q != NULL) {**

**if (q == head) {**

**head = head->next;**

**delete q;**

**q = head;**

**}**

**else {**

**list\* temp = head;**

**while (temp->next != q) {**

**temp = temp->next;**

**}**

**temp->next = q->next;**

**delete q;**

**q = temp;**

**}**

**}**

**}**

**else q = q->next;**

**}**

**if (find == false) {**

**cout << "Данный элемент не найден.\n";**

**}**

**else {**

**output(head);**

**cin.ignore();**

**}**

**}**

**}**

**void output(list\* head) {**

**if (empty(head) == 1) {**

**list\* q = head;**

**cout << "список\n";**

**while (q) {**

**cout << q->data<<" ";**

**q = q->next;**

**}**

**cout << endl;**

**}**

**}**

**void reverce(list\*& head) {**

**if (empty(head) == 1) {**

**list\* reversed\_list=NULL;**

**for (list\* node = head, \*next\_node; node != NULL; node = next\_node) {**

**next\_node = node->next;**

**node->next = reversed\_list;**

**reversed\_list = node;**

**}**

**head = reversed\_list;**

**output(head);**

**}**

**}**

**bool empty(list\* head) {**

**if (head != NULL) return 1; else {**

**cout << "список пуст\n";**

**return 0;**

**}**

**}**

**void add(list\*& head) {**

**int choise;**

**do {**

**cout << "введите новый элемент списка: ";**

**create\_node(head);**

**cout << "хотите добавить еще элемент? (1 - да, 2 - нет)\n";**

**cin >> choise;**

**if (choise != 1 && choise != 2) {**

**cout << "Вы ввели неправильный номер.\nВвод будет прекращен.\n";**

**cin.ignore();**

**cin.ignore();**

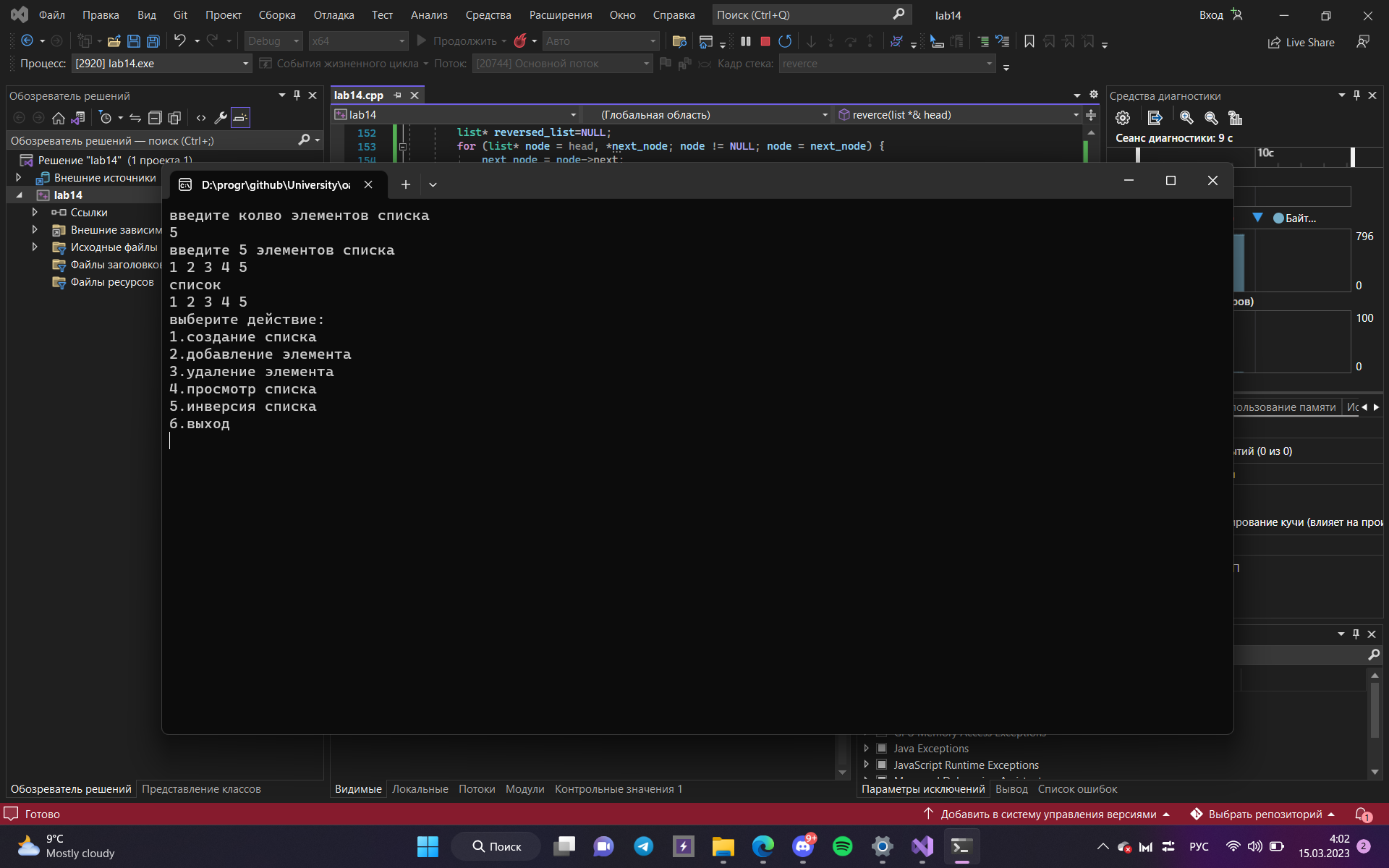
**}**

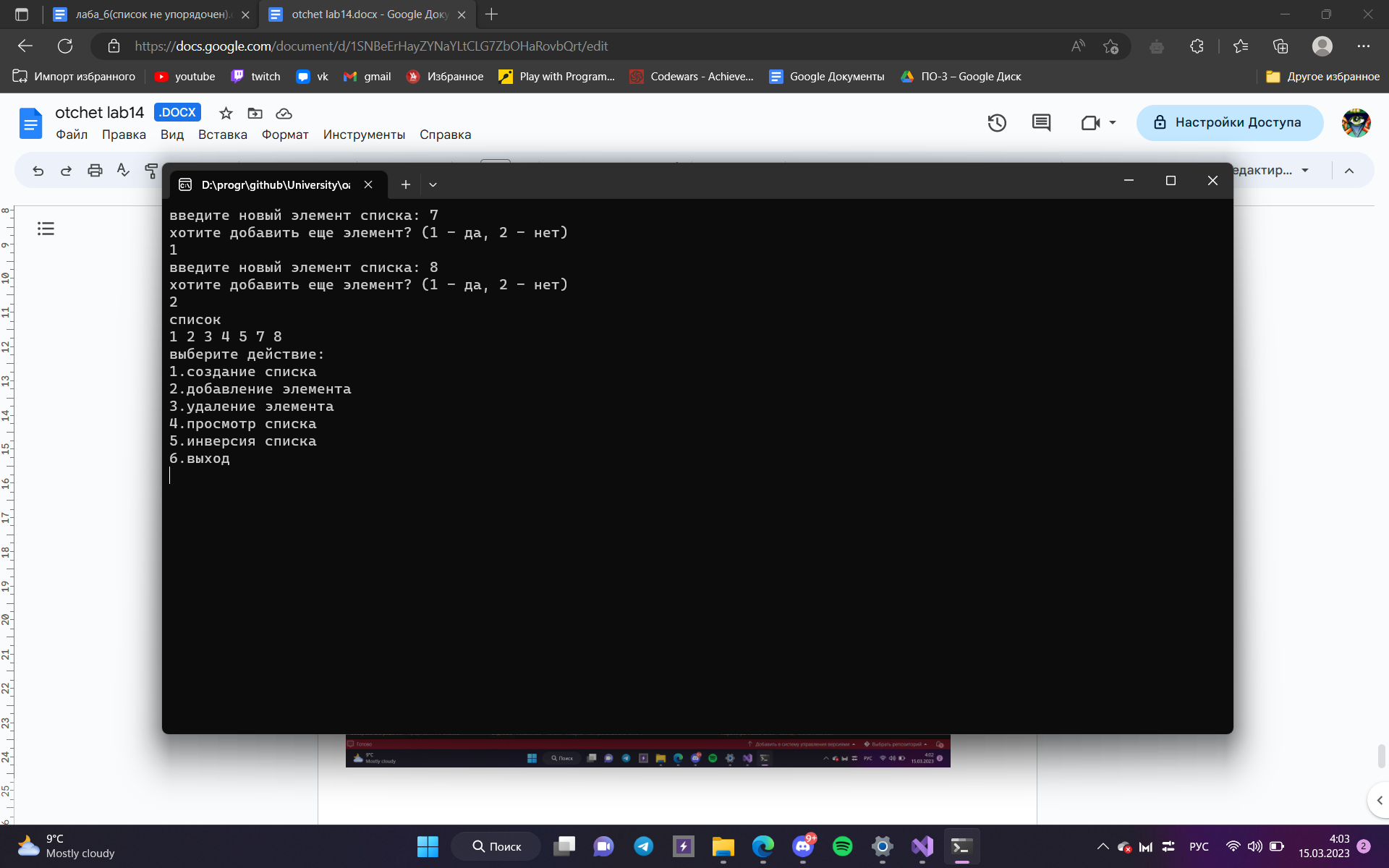
**} while (choise != 2);**

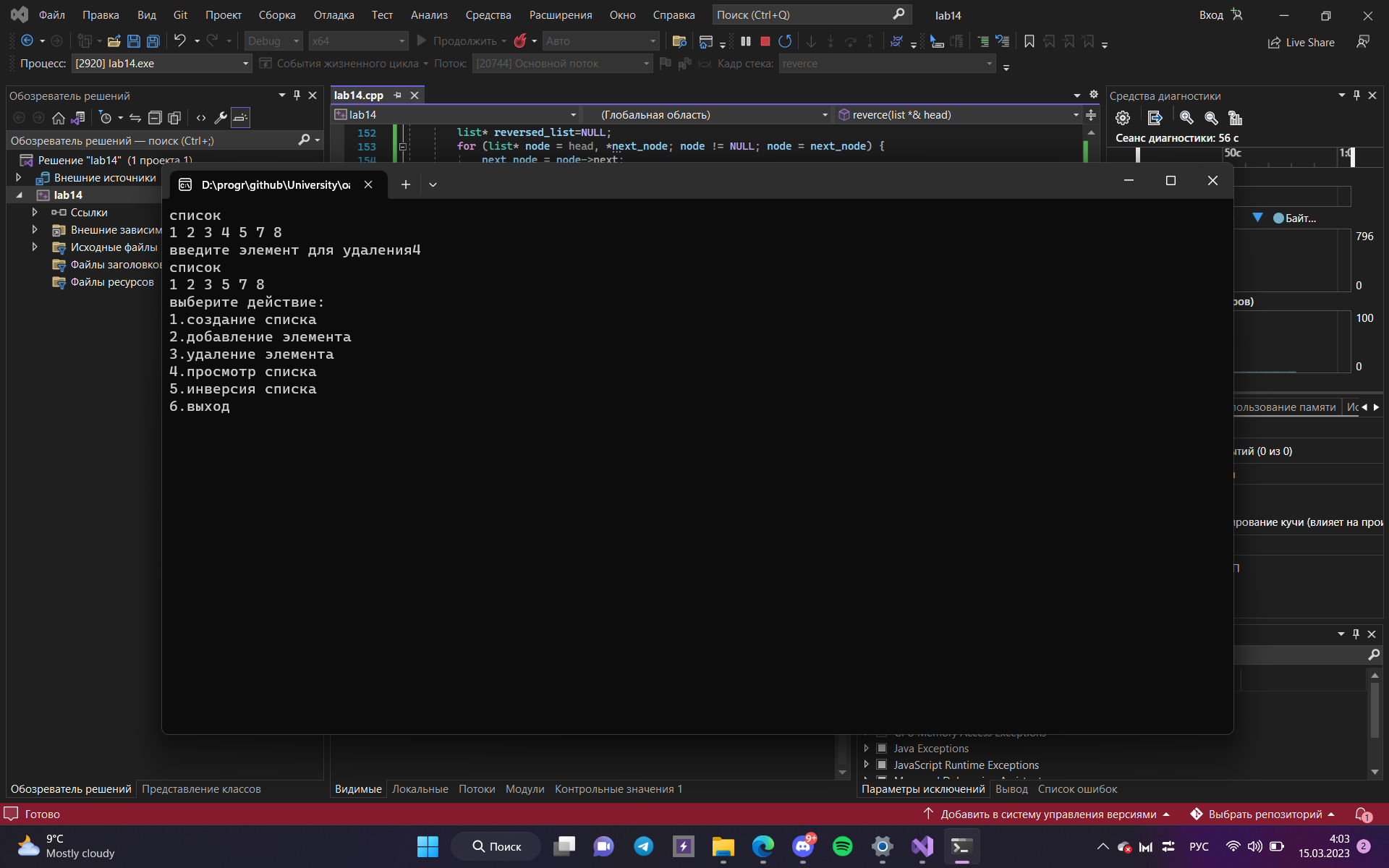
**output(head);**

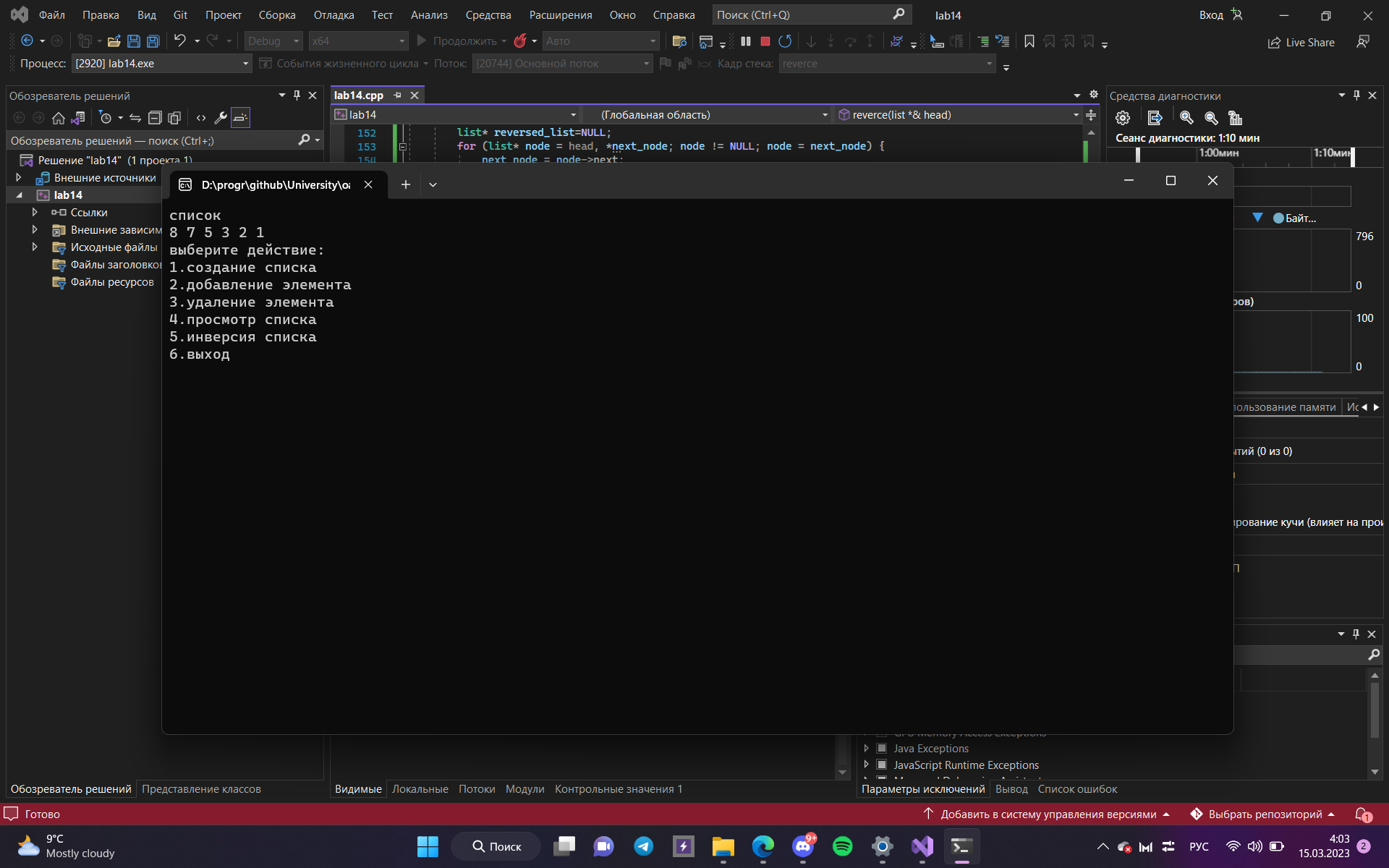
**}**

Результаты выполнения программы:









Вывод: приобрел навык работы с динамической памятью и указателями на С/C++; изучил принципы работы с динамическими структурами данных: списками и деревьями.