**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

Тема: С мира по нитки

|  |  |  |
| --- | --- | --- |
| Студент гр. |  | Косенко А.Р. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

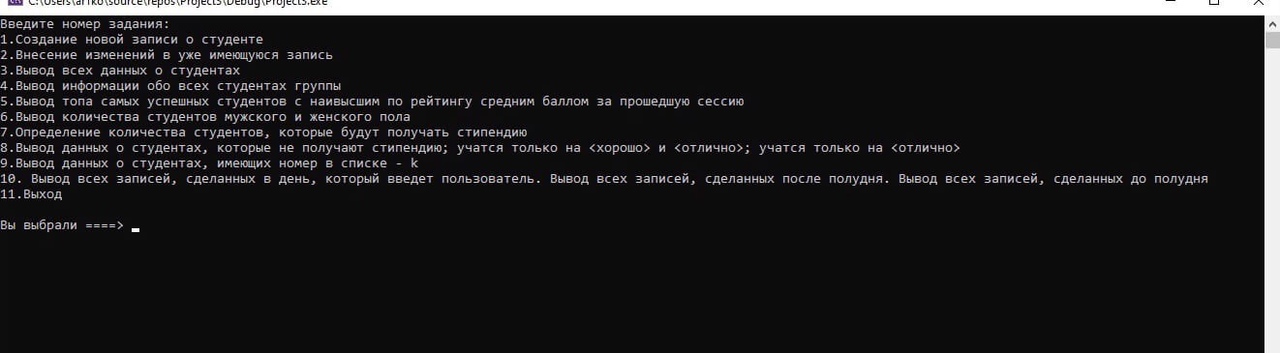
2021

Первый вариант курсовой работы - 3 практических работы, объединенных в единый интерфейс.

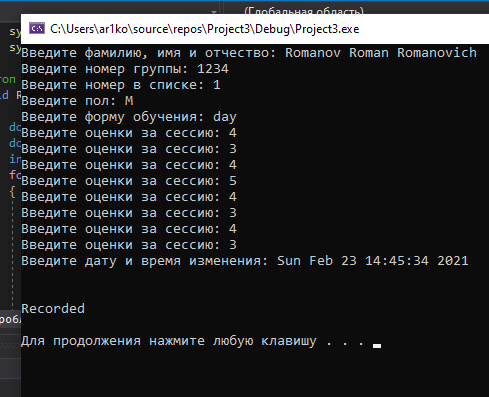
* Максимальный балл за работу - 100 баллов. 33 балла за каждую правильно выполненную работу.  
  > 50 баллов - оценка "3"  
  > 85 баллов - оценка "4"  
  Работы выполненные не в полном объеме оцениваются пропорционально объему проделанной работы.
* Для работы самостоятельно оформляется отчёт по всем выполненным работам.
* 4 - максимальная оценка за данный вариант курсовой работы.
* К сдаче курсовой работы варианта №1 допускаются все студенты, сдавшие не менее 2 работ. В ином случае студенту будет выставлена отметка "неудовлетворительно"

## Выполнение работы

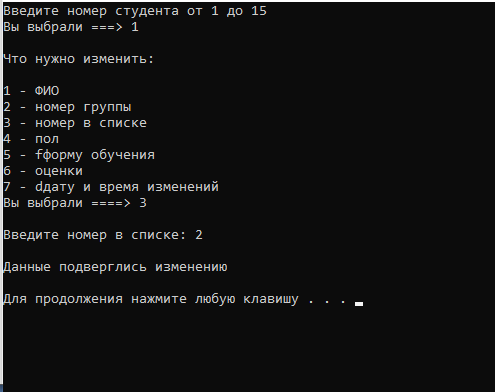
1 работа  
1)Визуал Меню



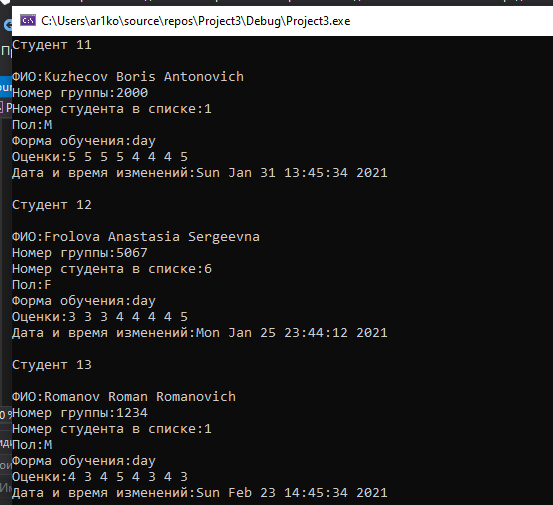
2)Пример работы первого задания.



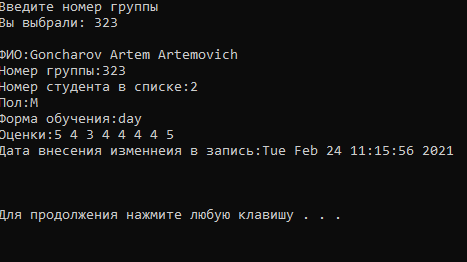
3) Пример работы второго задания

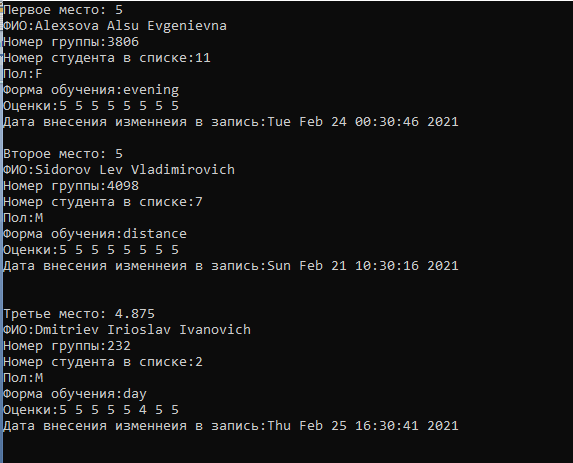
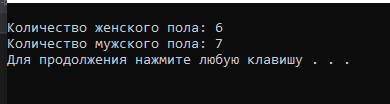


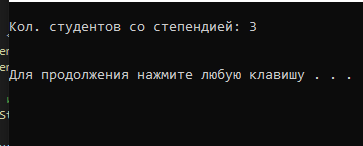
4)Пример работы 3 задания

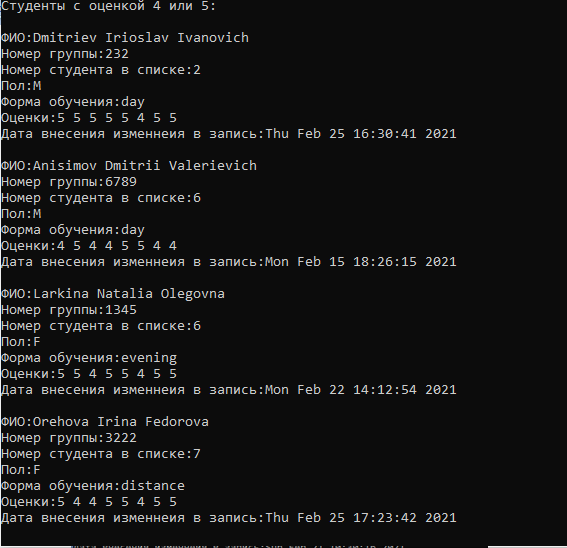
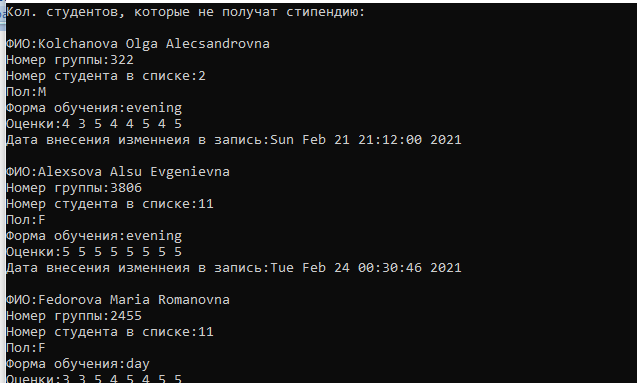
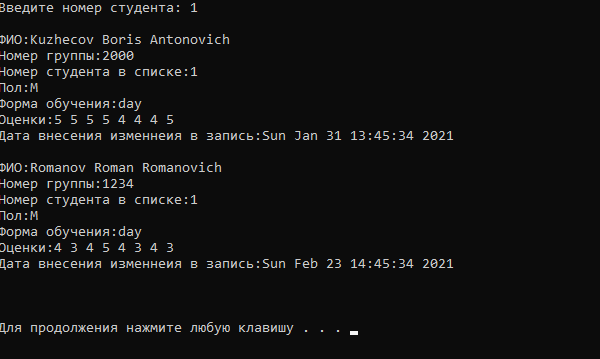


5)Пример решения четвертого задания



6)Пример работы пятого задания  
****7)Пример работы шестого задания  


8)Пример работы седьмого задания  


9)Пример работы восьмого задания  
****10)Пример работы 9 задания  
****11)Пример работы 10 задания  
****  
**  
Вывод**

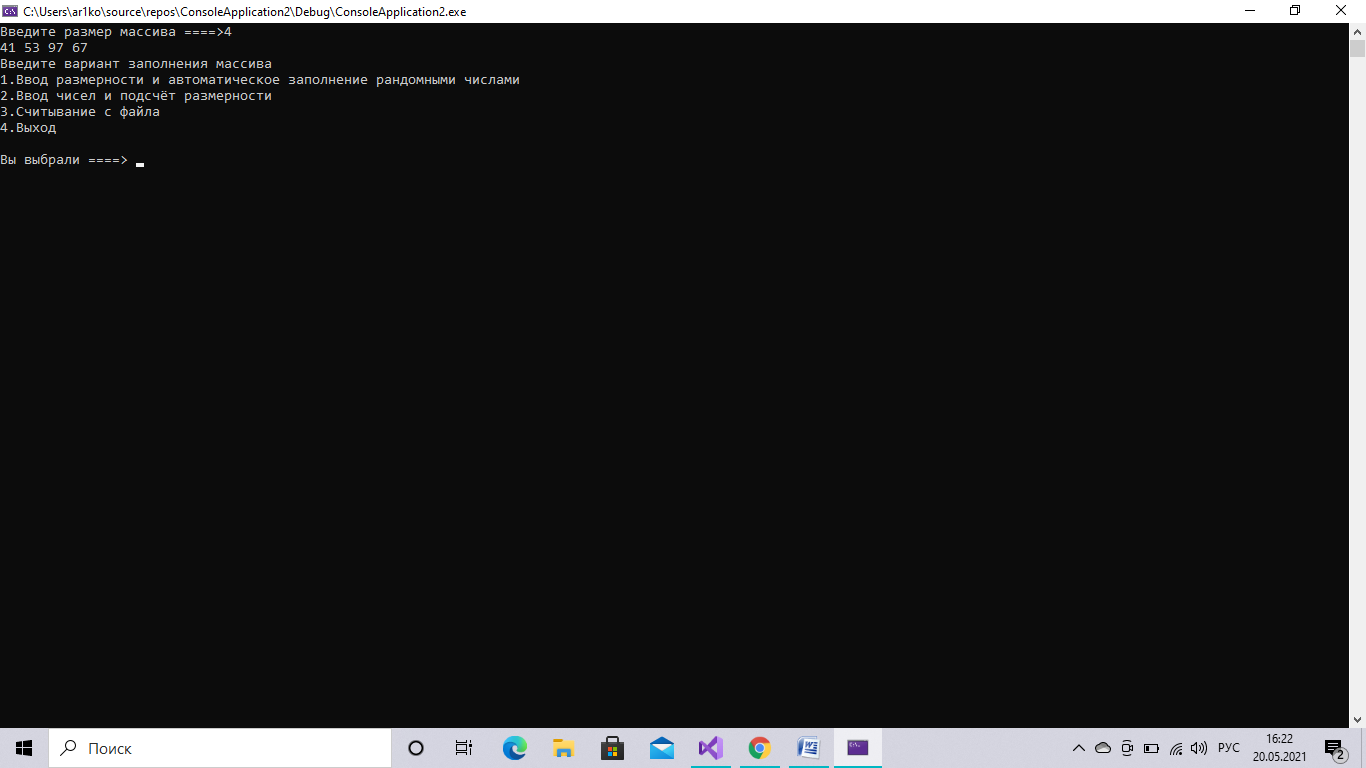
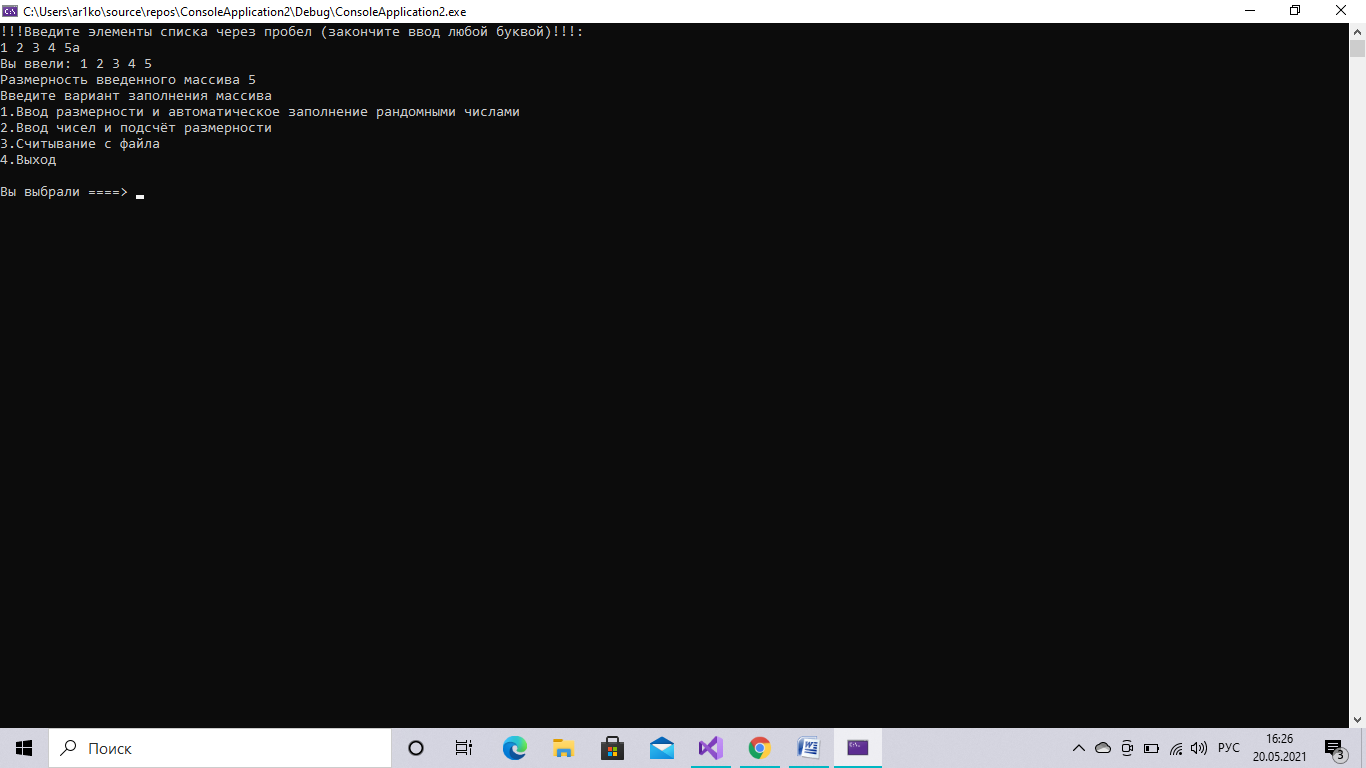
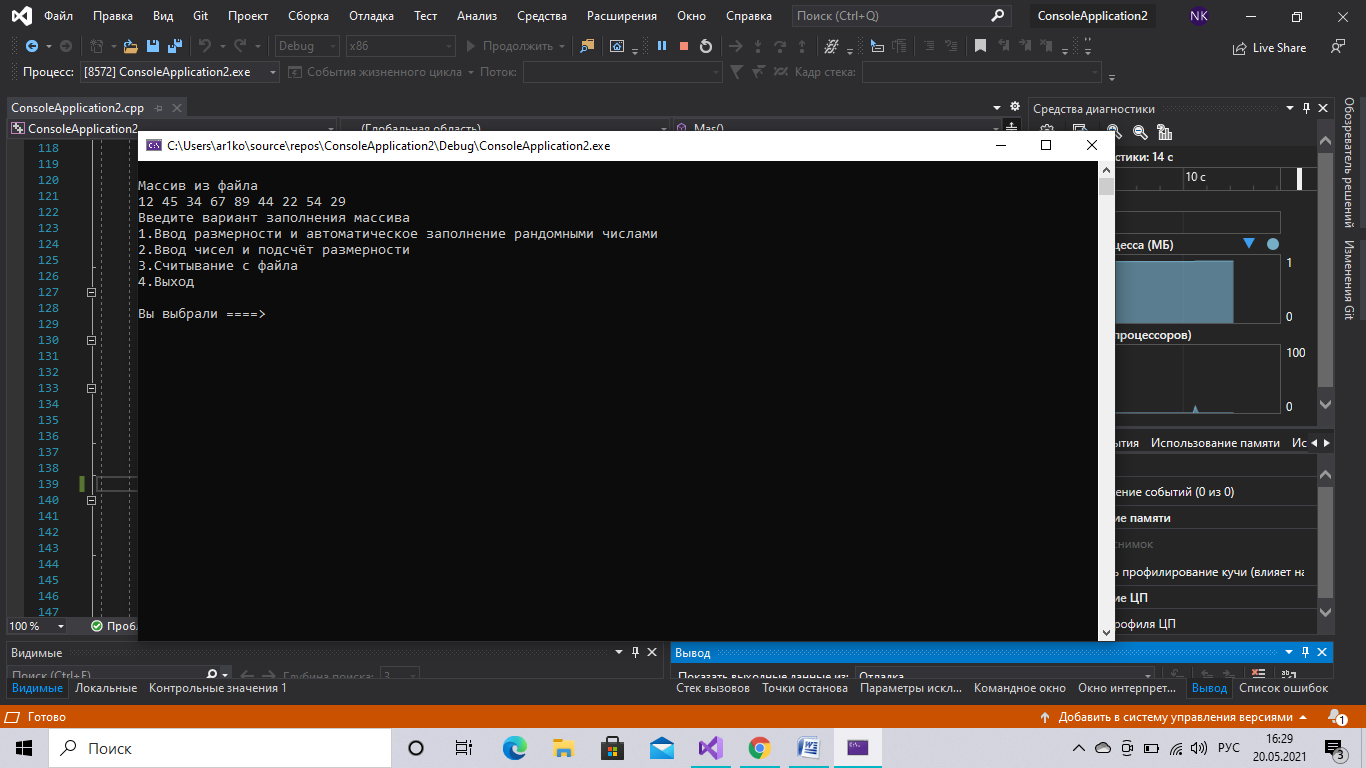
Была проведена работа со структурами.

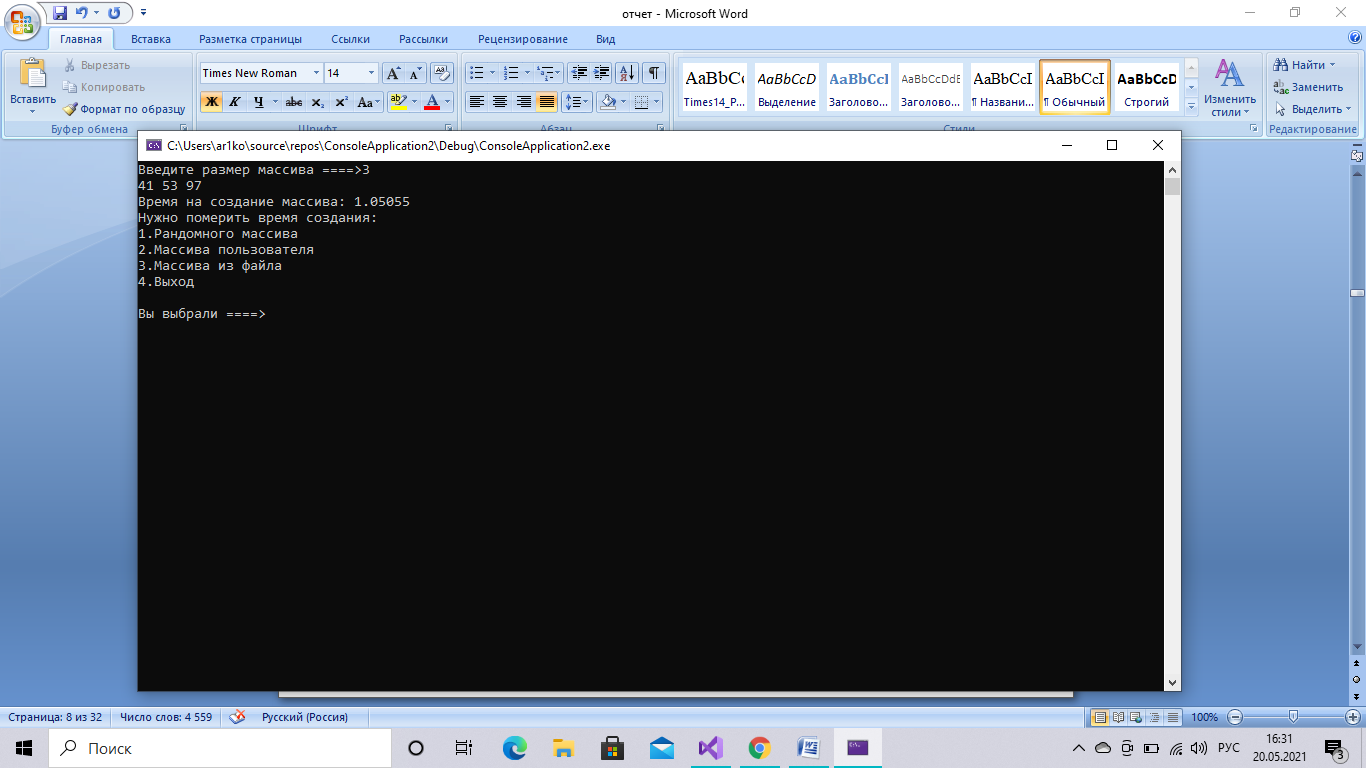
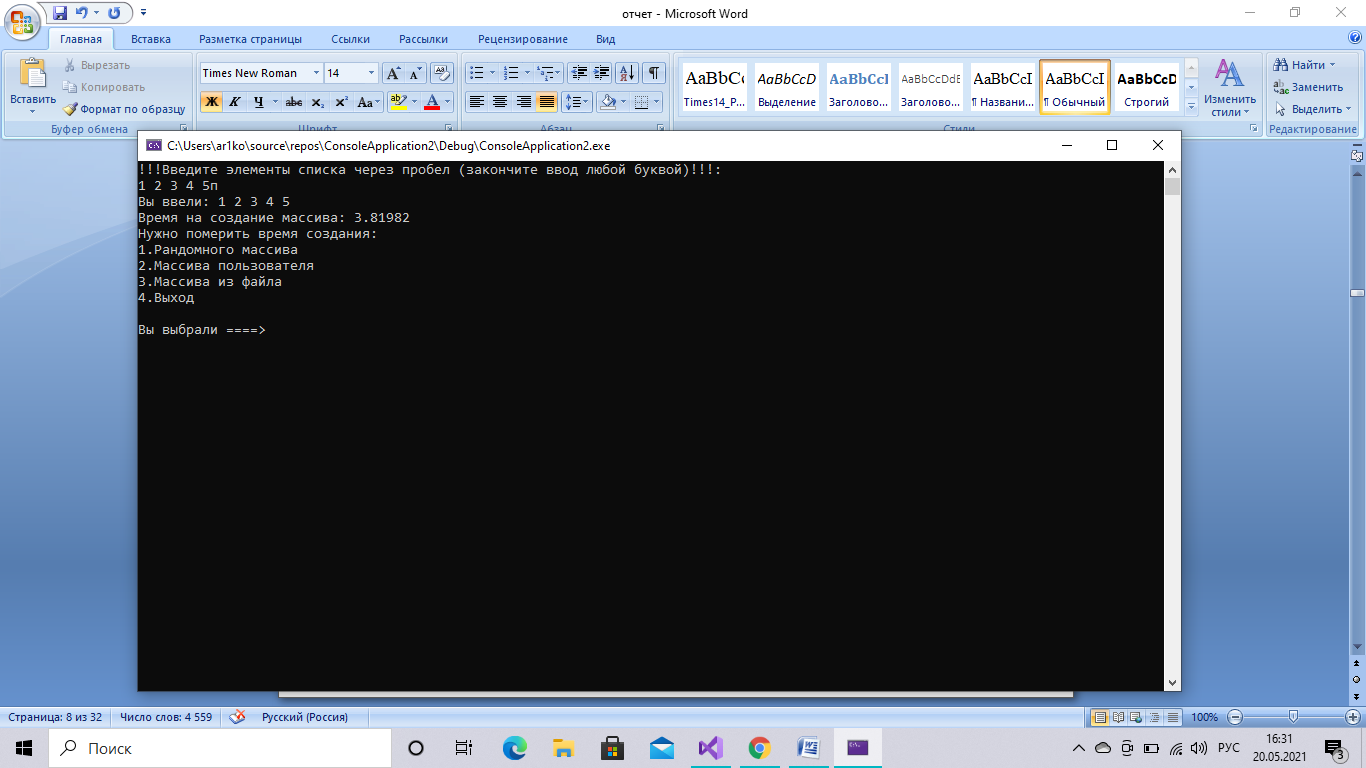
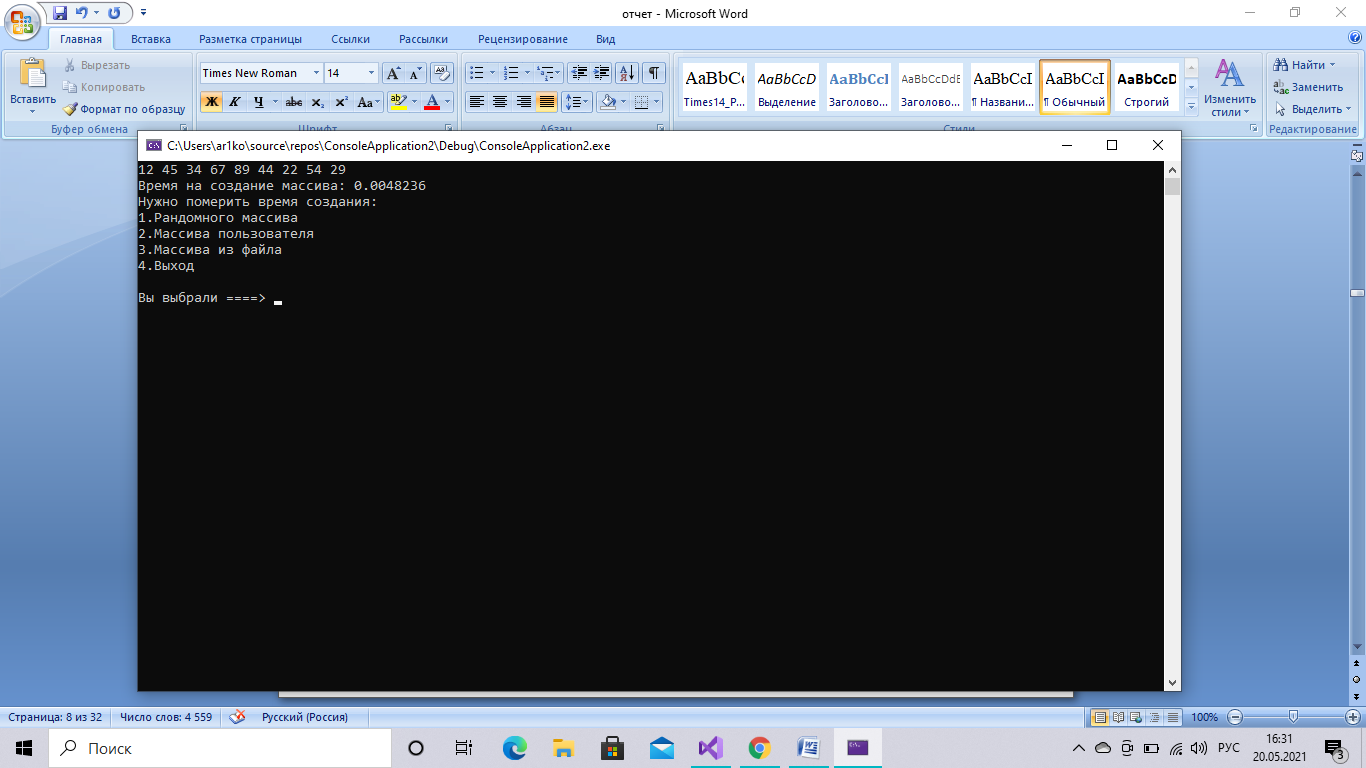
Структуры полезны, когда надо объединить несколько переменных с разными типами под одним именем. Это делает программу более компактной и более гибкой для внесения изменений. Также структуры незаменимы, когда необходимо сгруппировать некоторые данные, к примеру запись из базы данных.

2 работа.  
**Выполнение работы.**

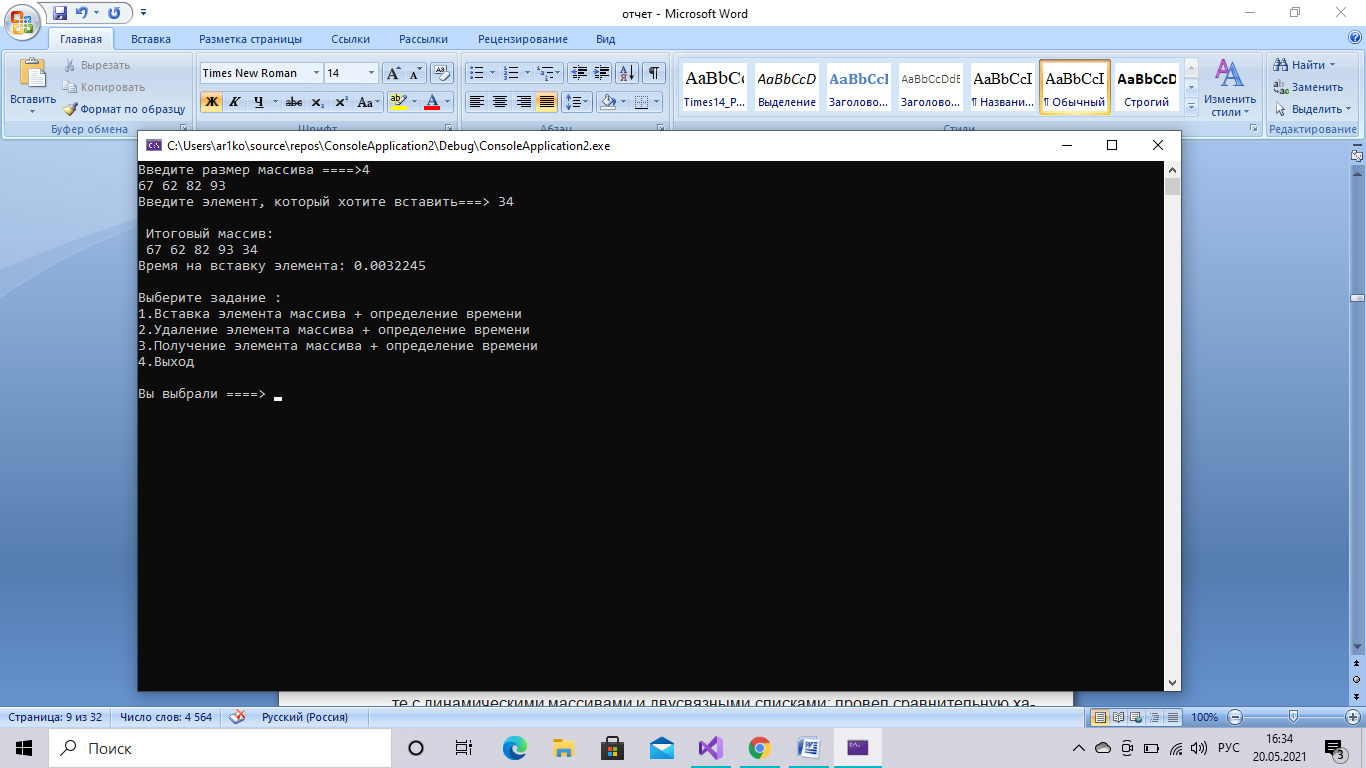
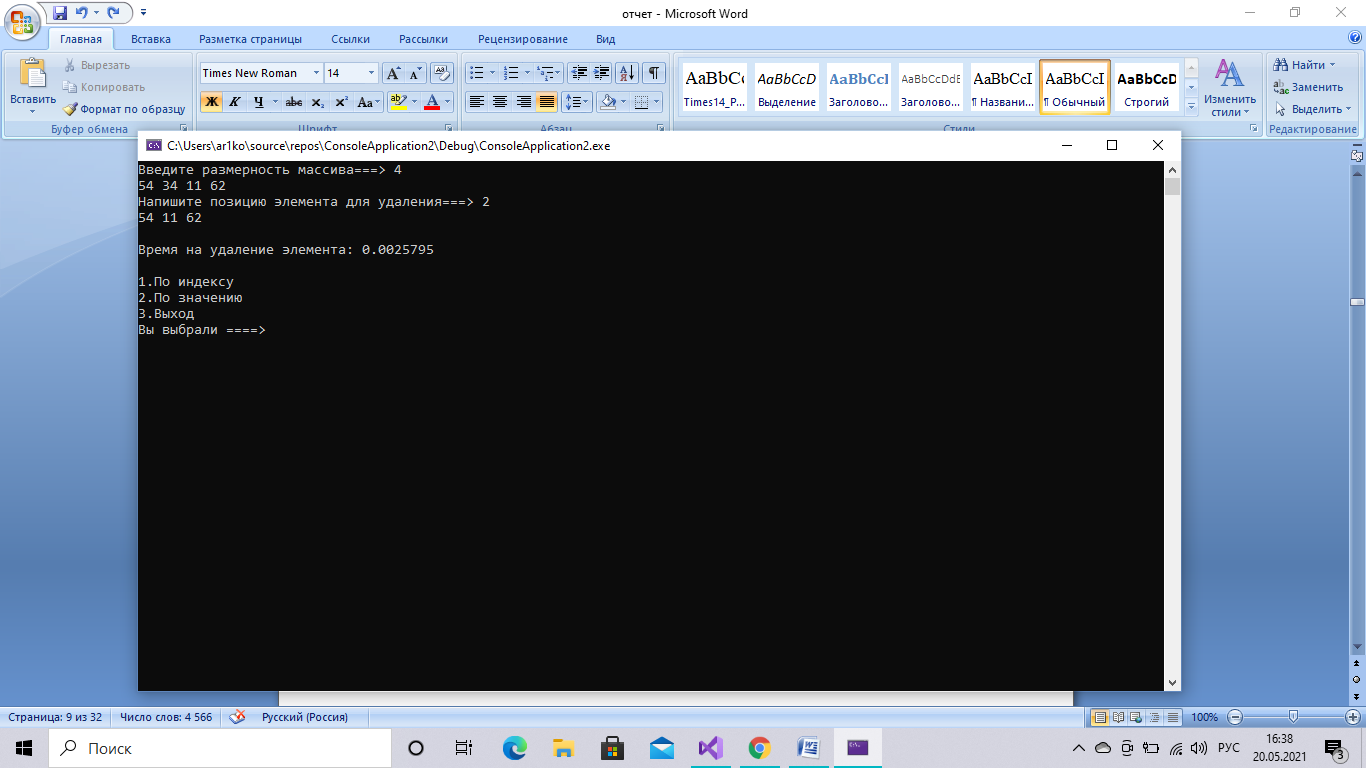
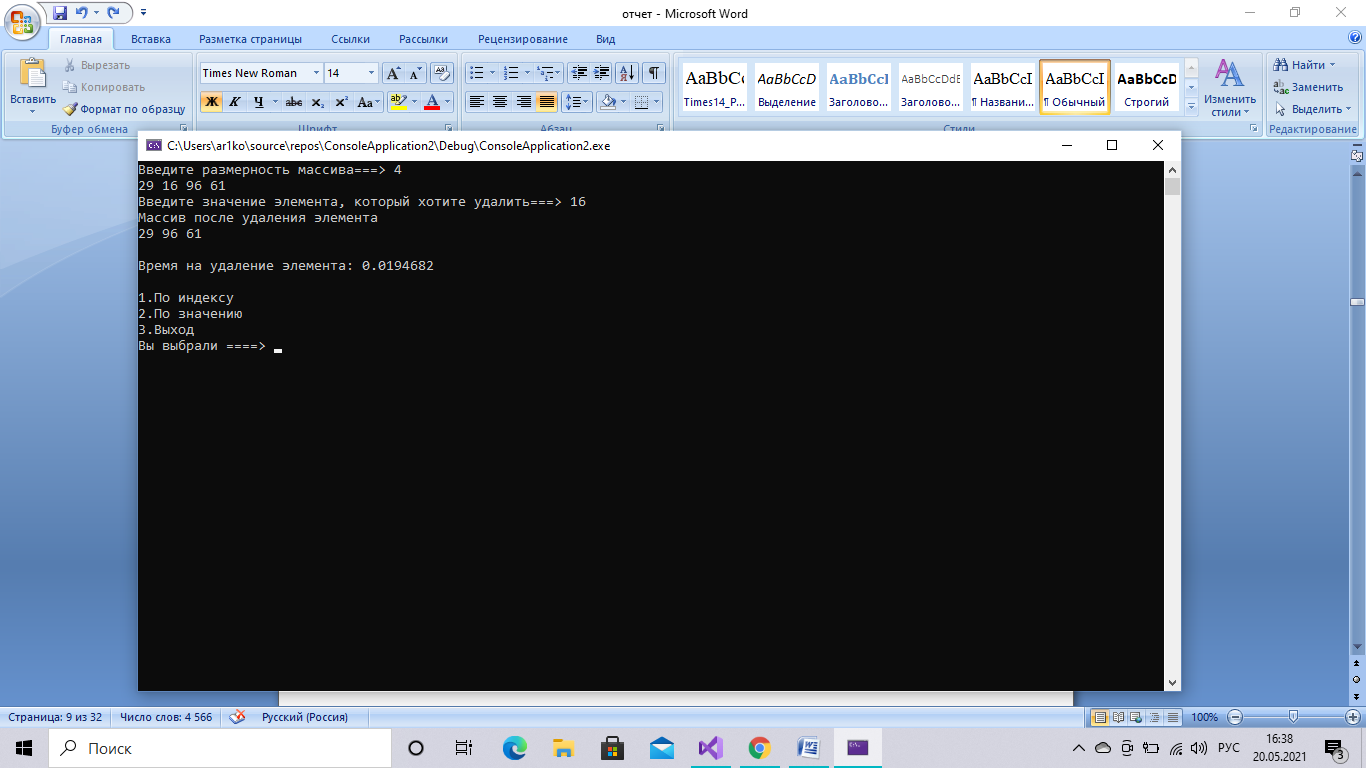
**1 задание.**

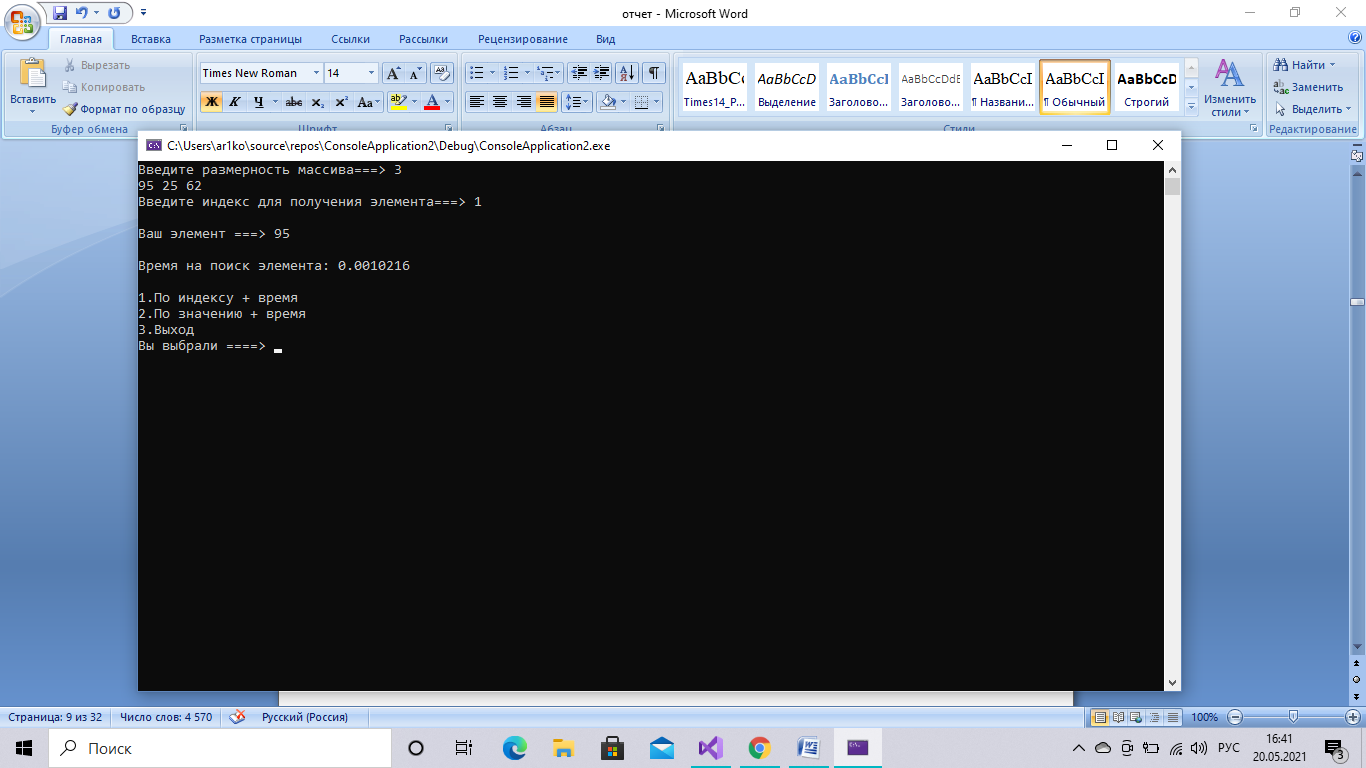
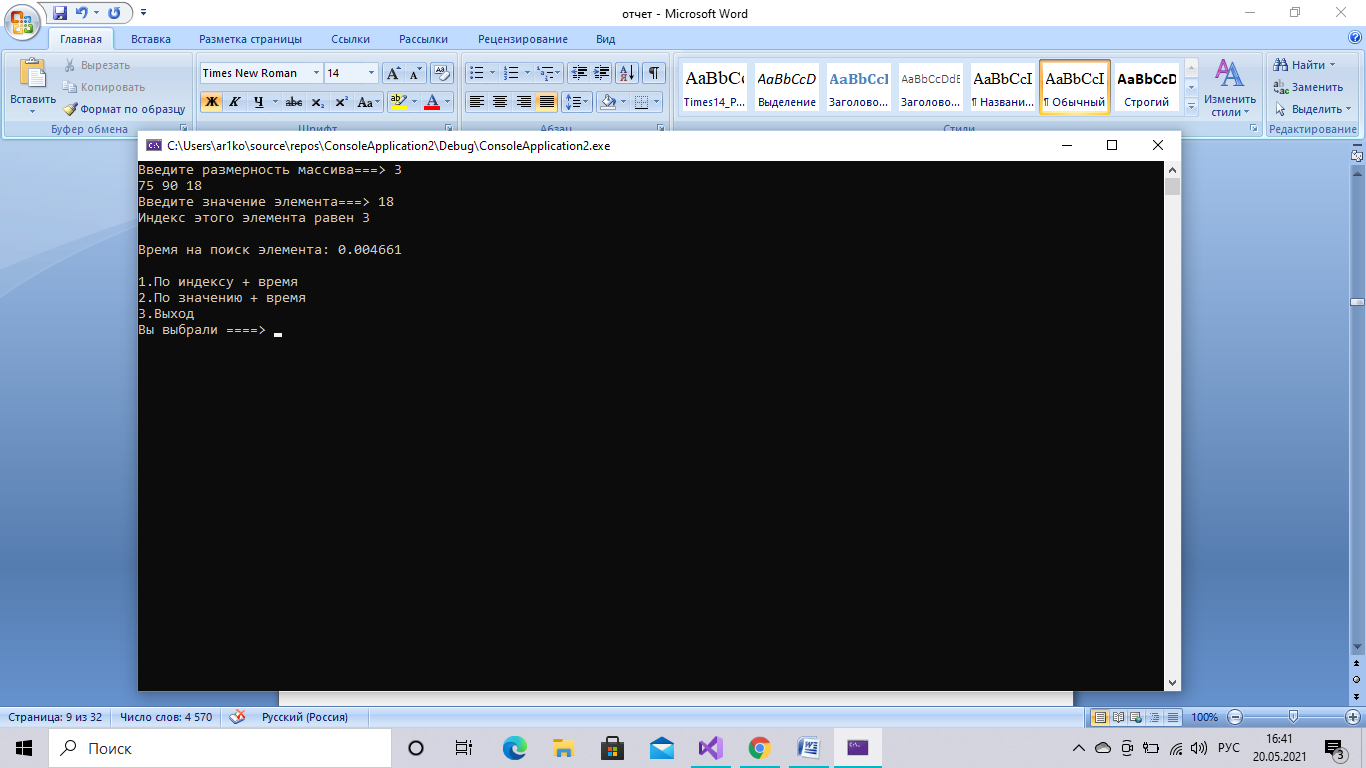
**А)**

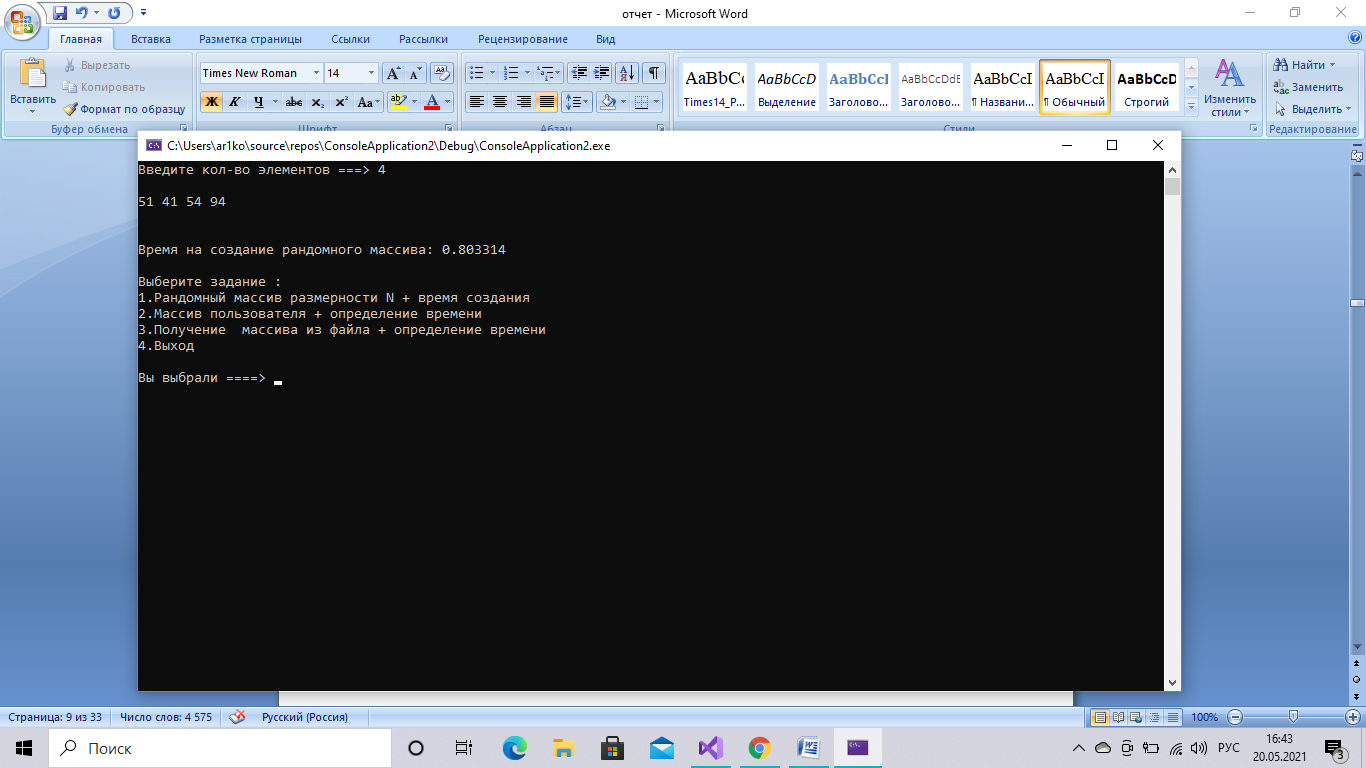
**  
б)  
  
в)  
  
2 задание**

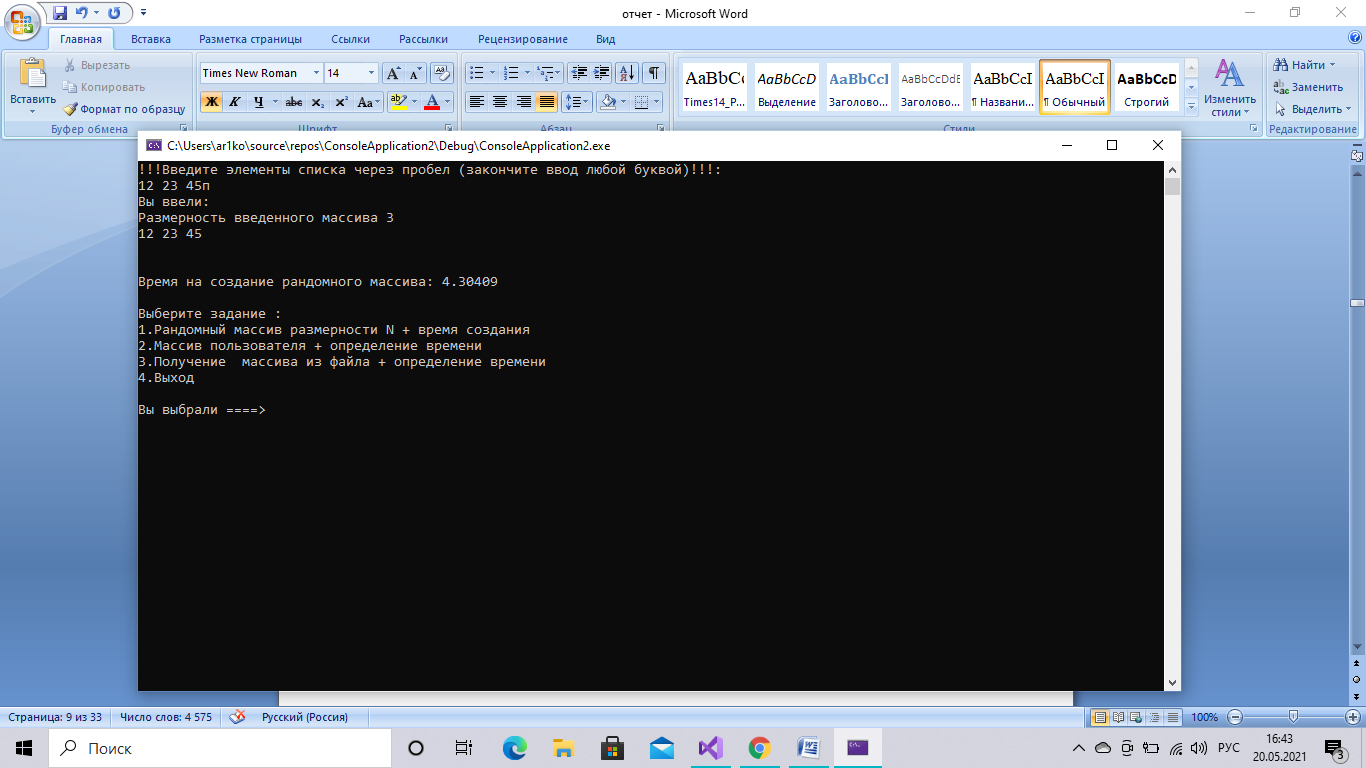
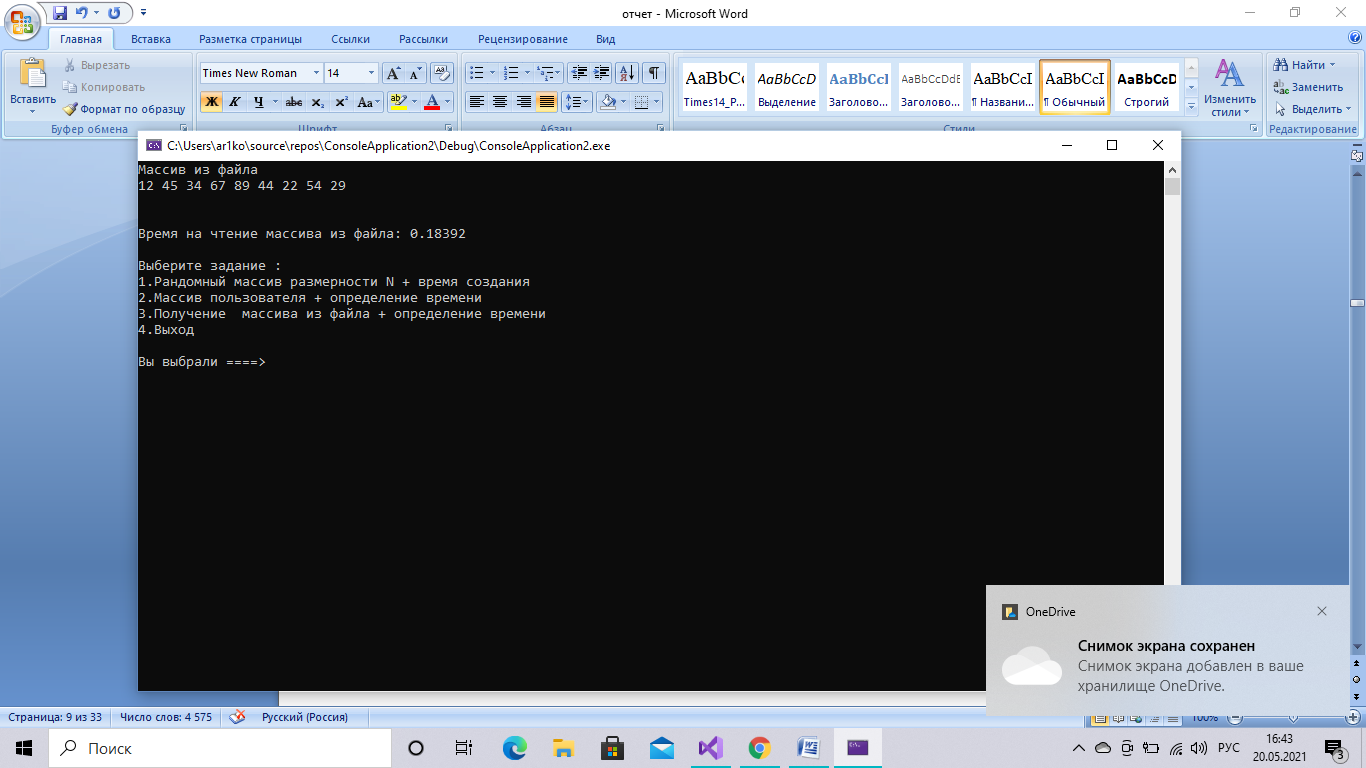
**А)  
  
б)  
  
в)  
**

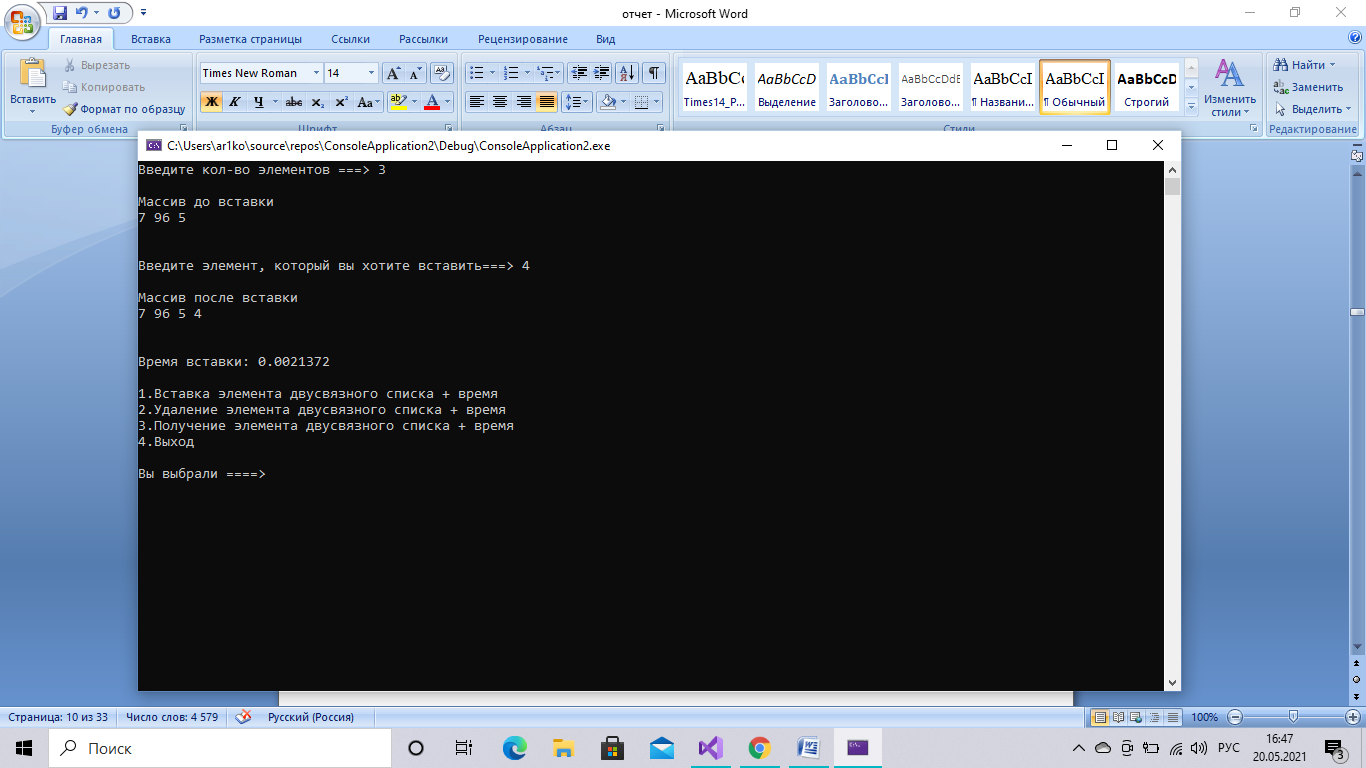
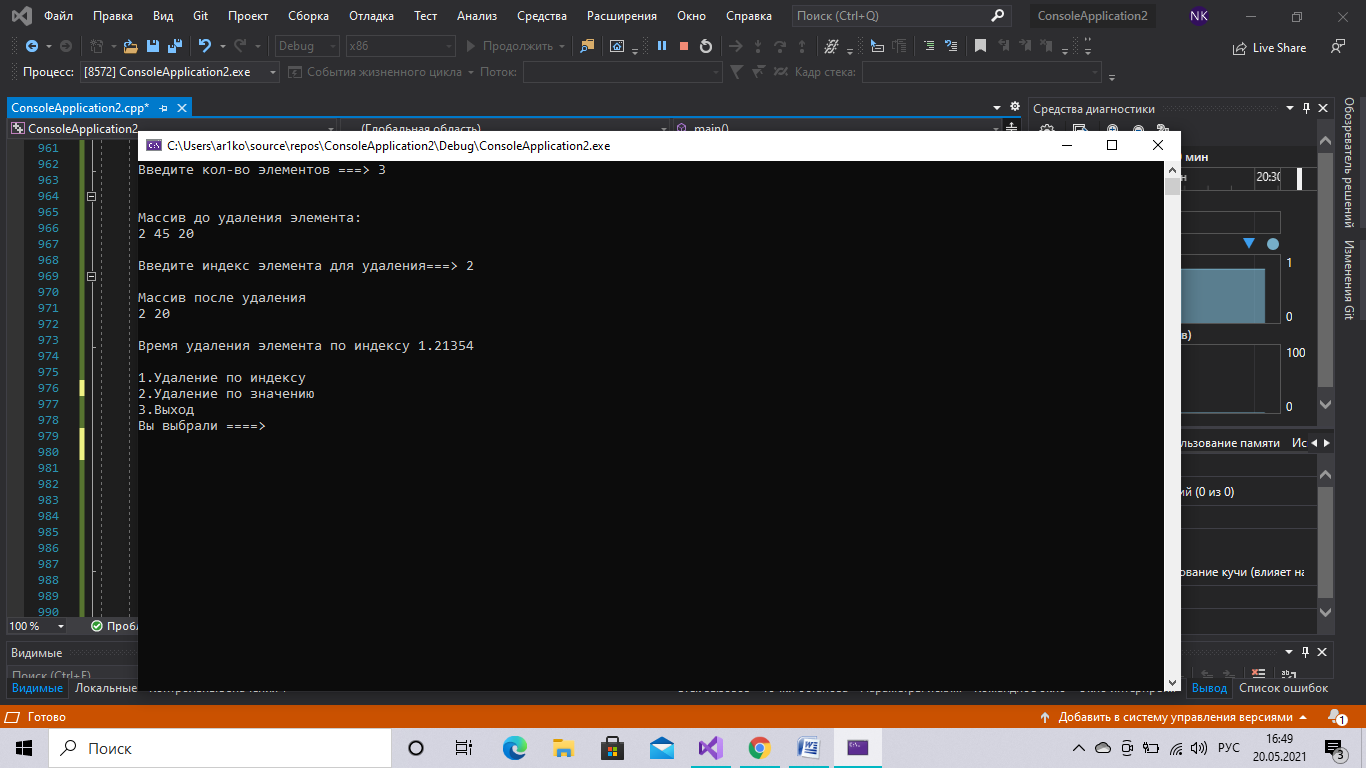
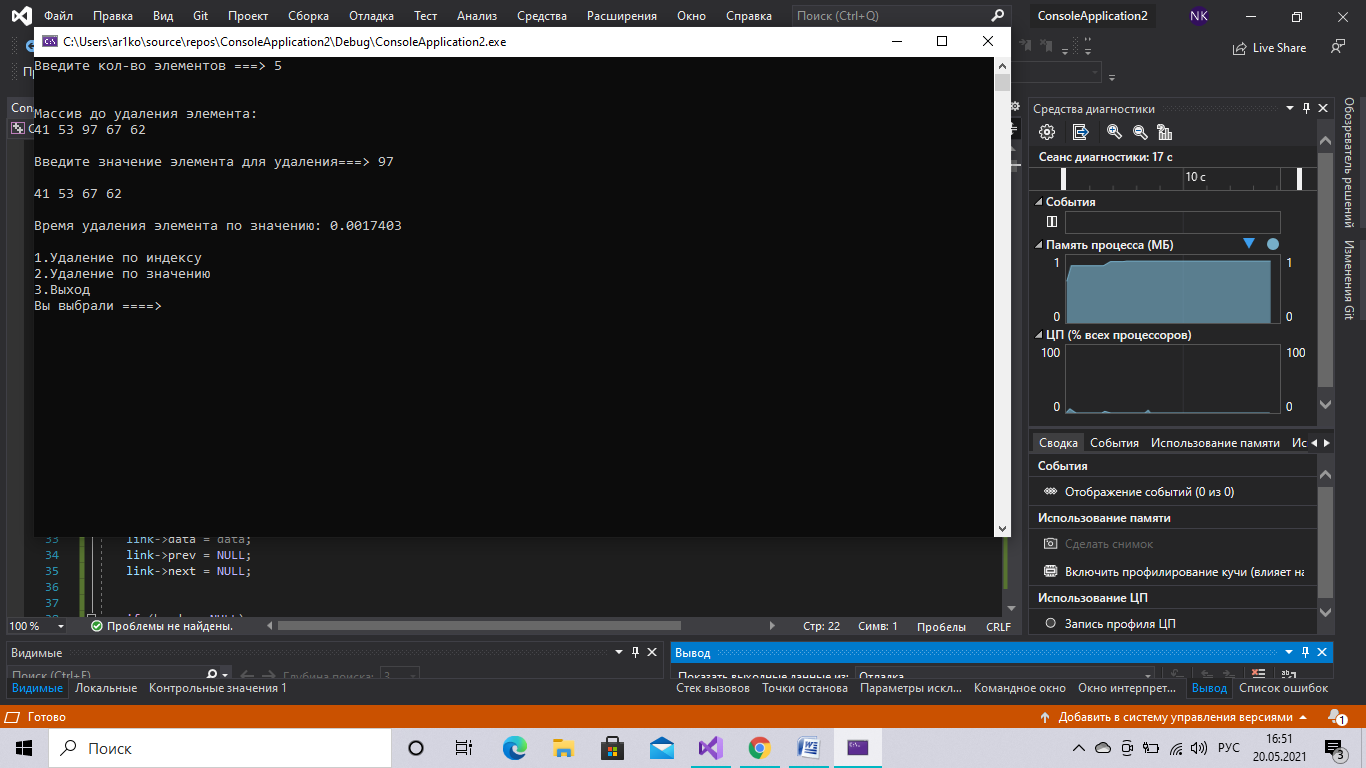
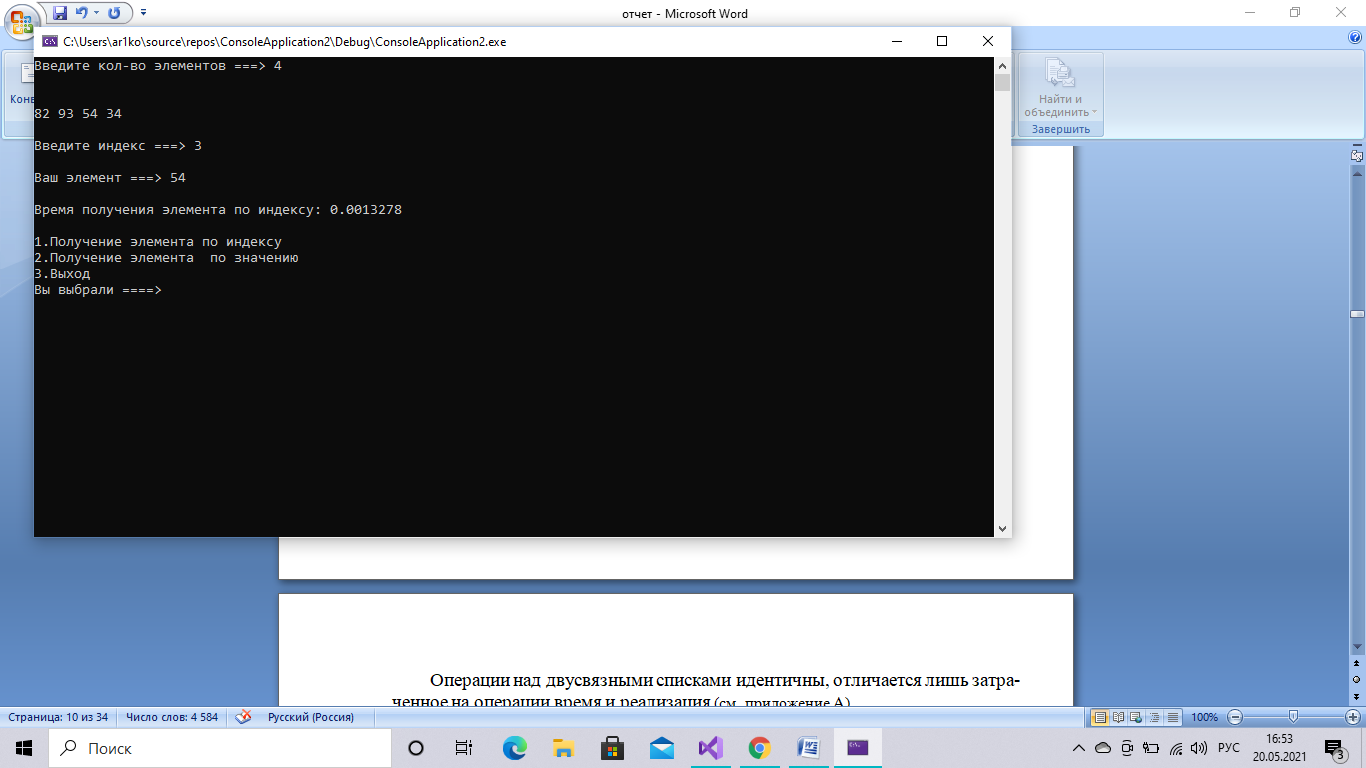
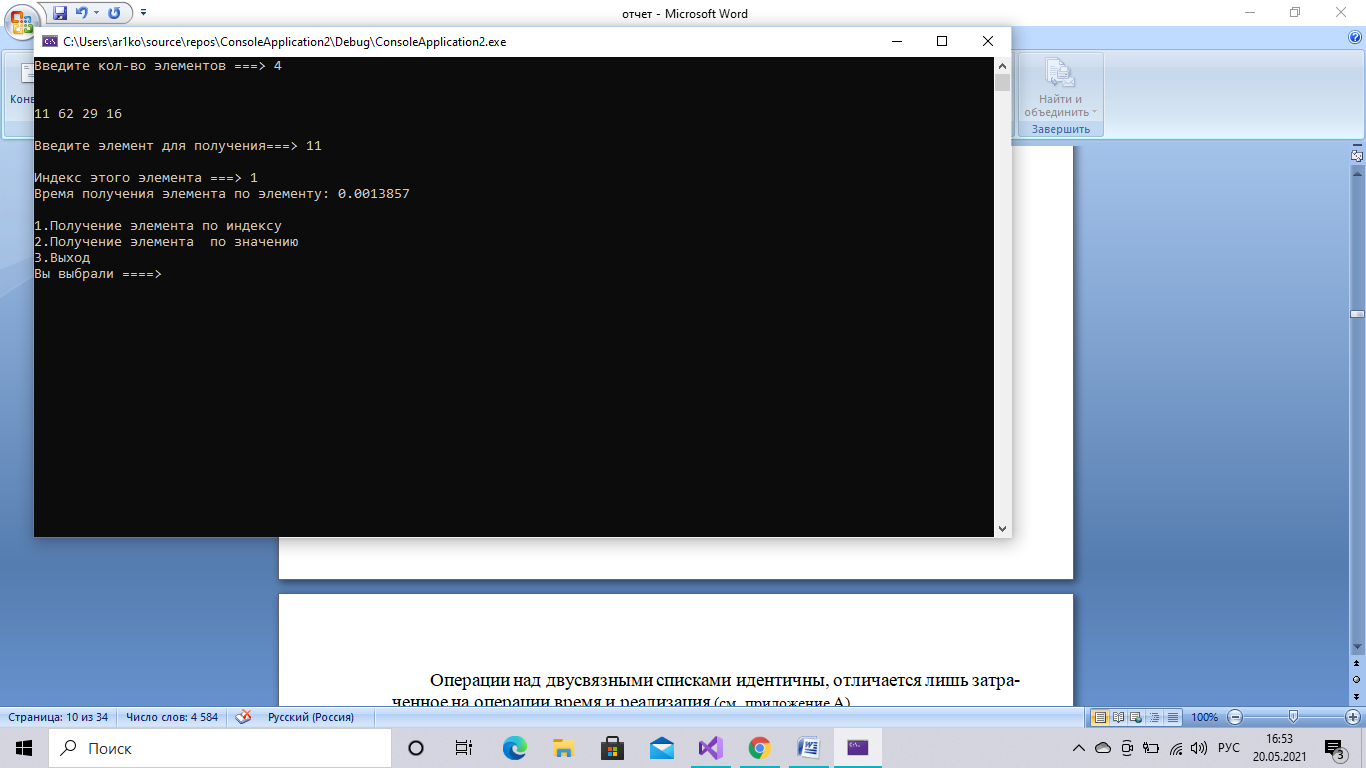
**3 задание +4 задание**

**1)  
  
2)  
а)  
  
б)  
  
3)  
а)**

**  
б)  
  
 5+6 задание**

**1)  
  
2)**

**  
3)  
  
7+8 задание**

**1)  
  
2)  
а)  
  
б)  
  
3)  
а)  
  
б)  
**

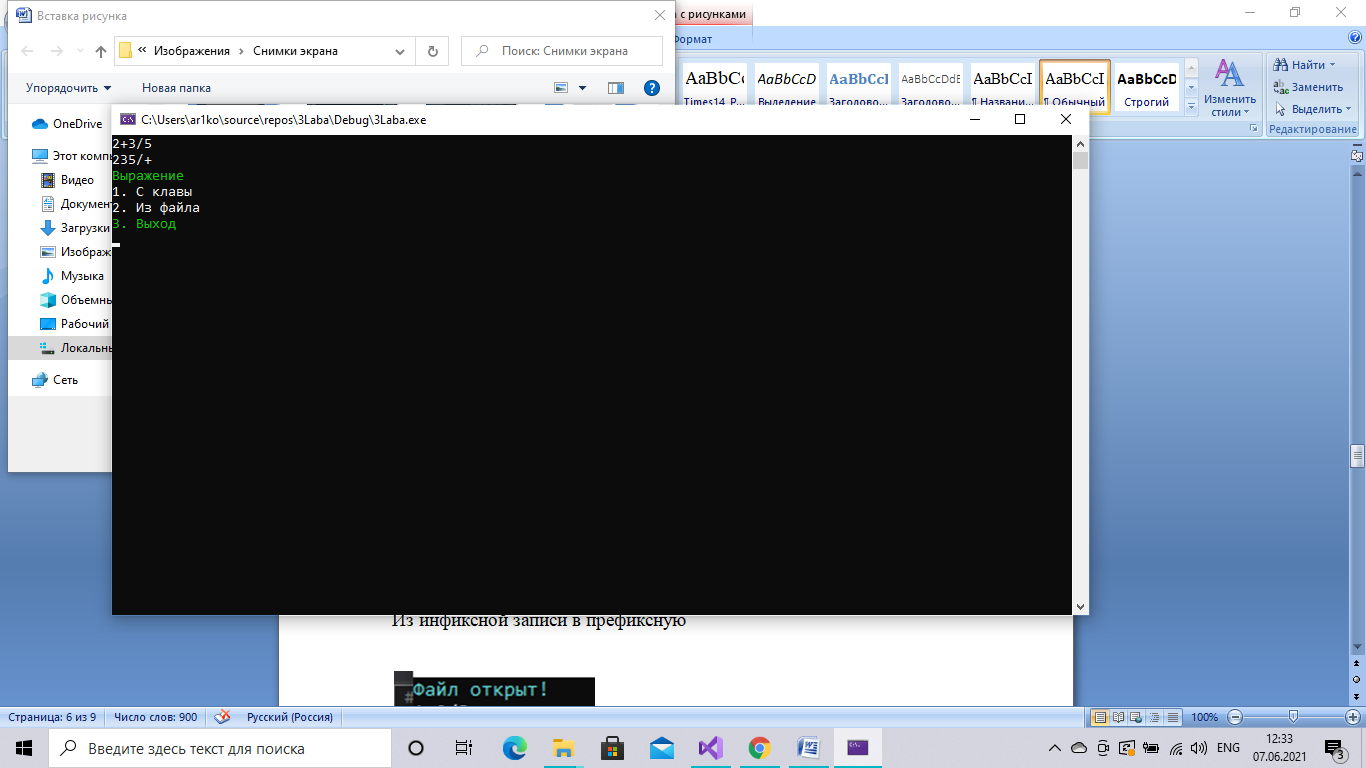
Операции над двусвязными списками идентичны, отличается лишь затраченное на операции время и реализация (см. приложение A)

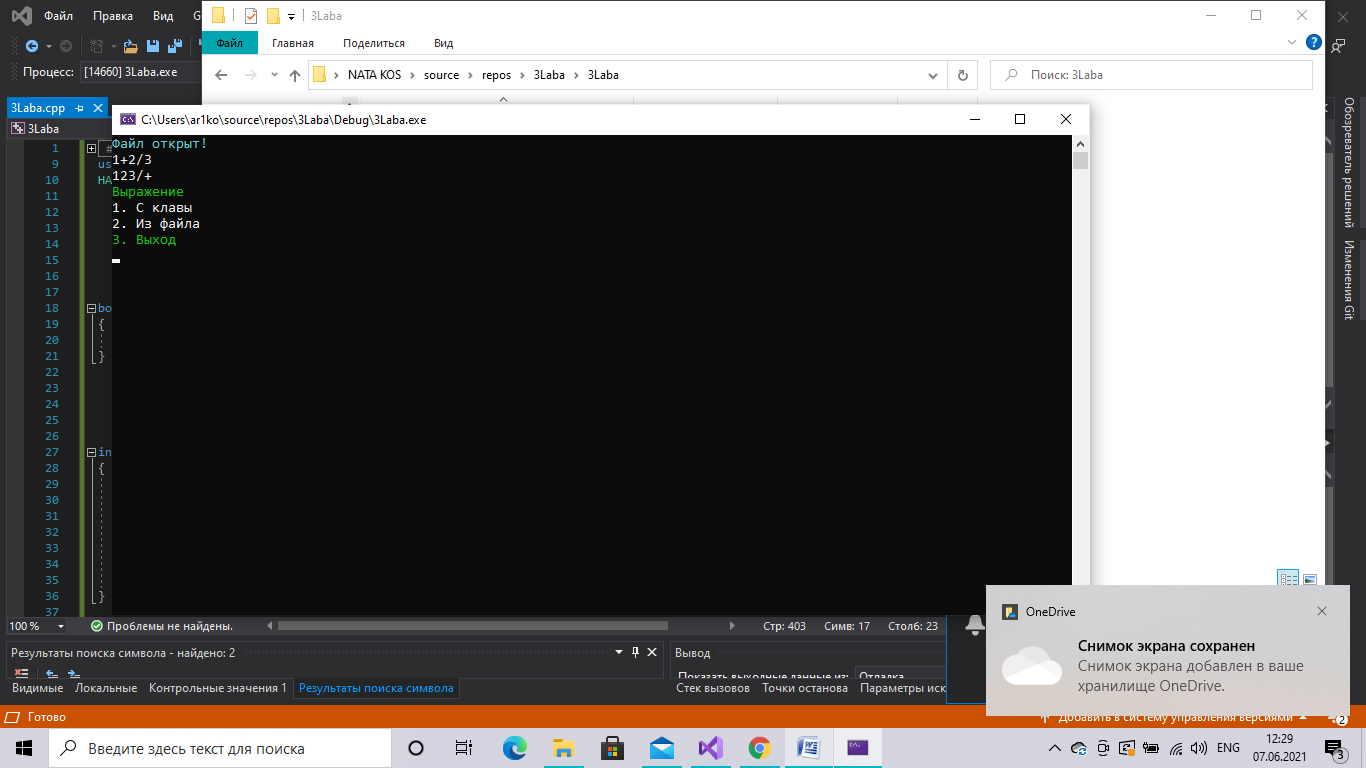
**Выводы**.

В ходе выполнения практической работы я изучила свойства и организацию динамических массивов и двусвязных списков; получил практические навыки в работе с динамическими массивами и двусвязными списками; провел сравнительную характеристику скорости вставки, получения и удаления элементов из них.

3 работа.  
Задание 1

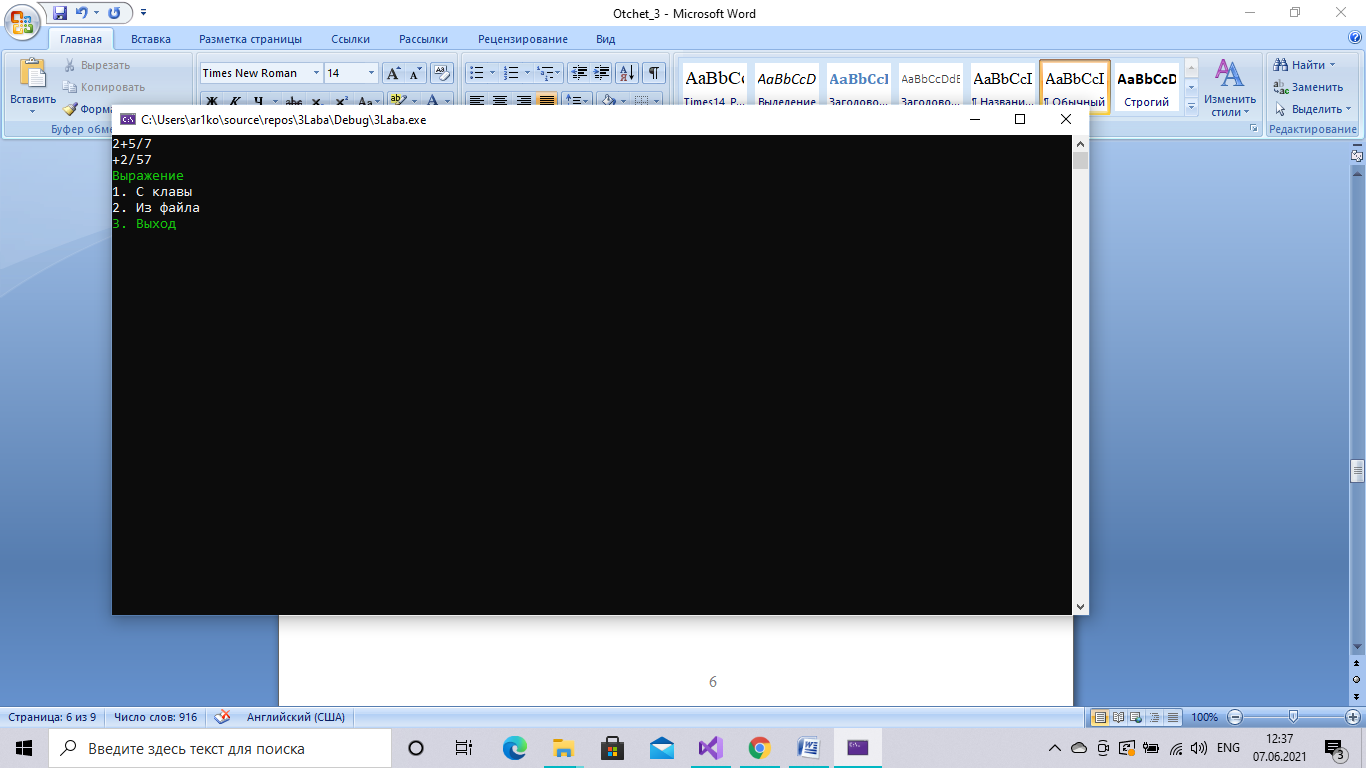
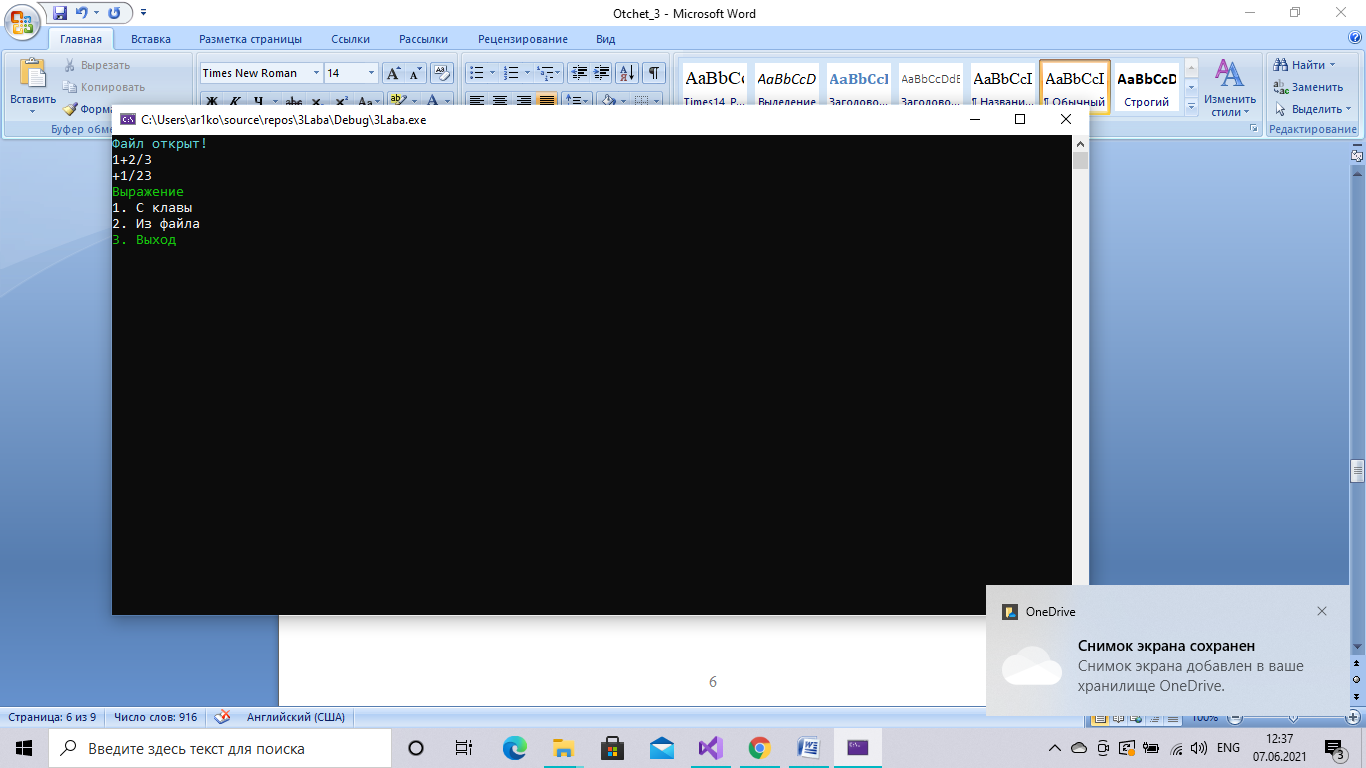
Из инфиксной записи в постфиксную



****

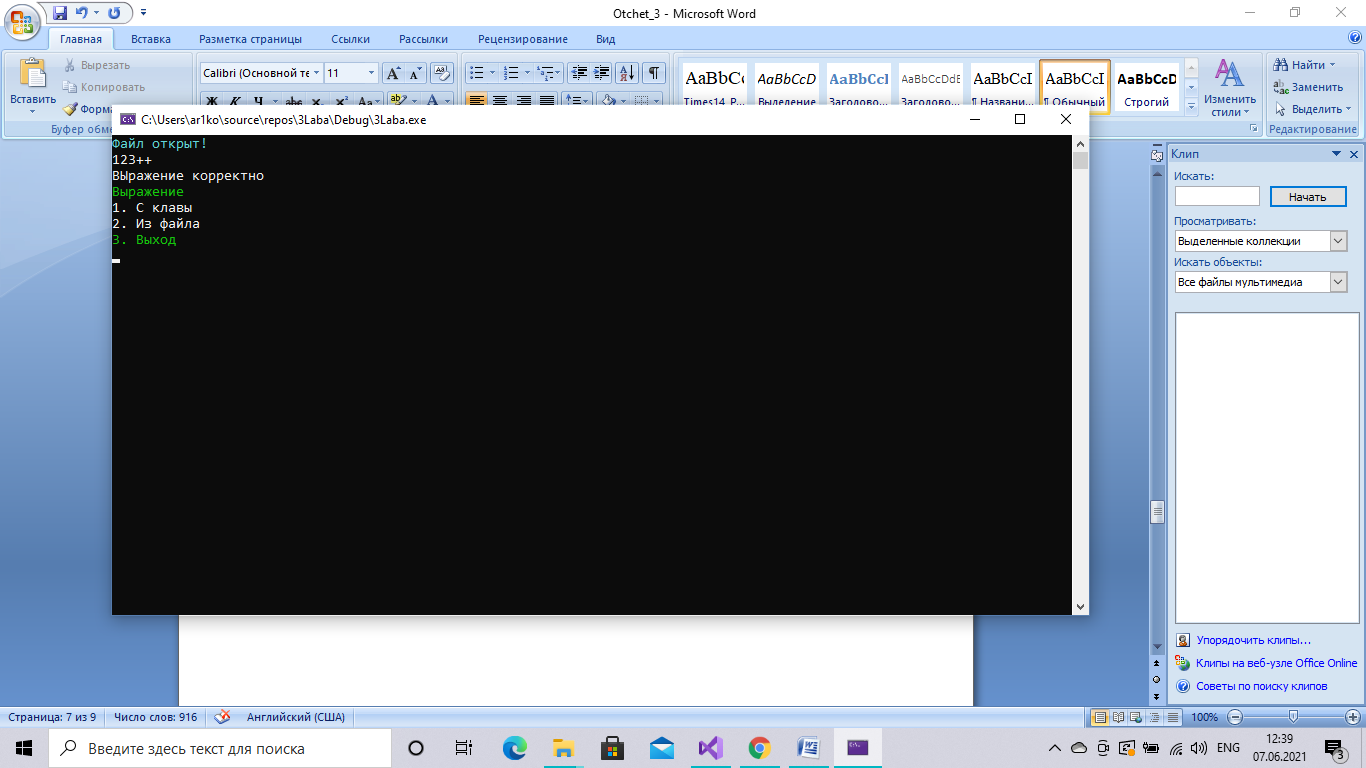
Задание 1

Из инфиксной записи в префиксную

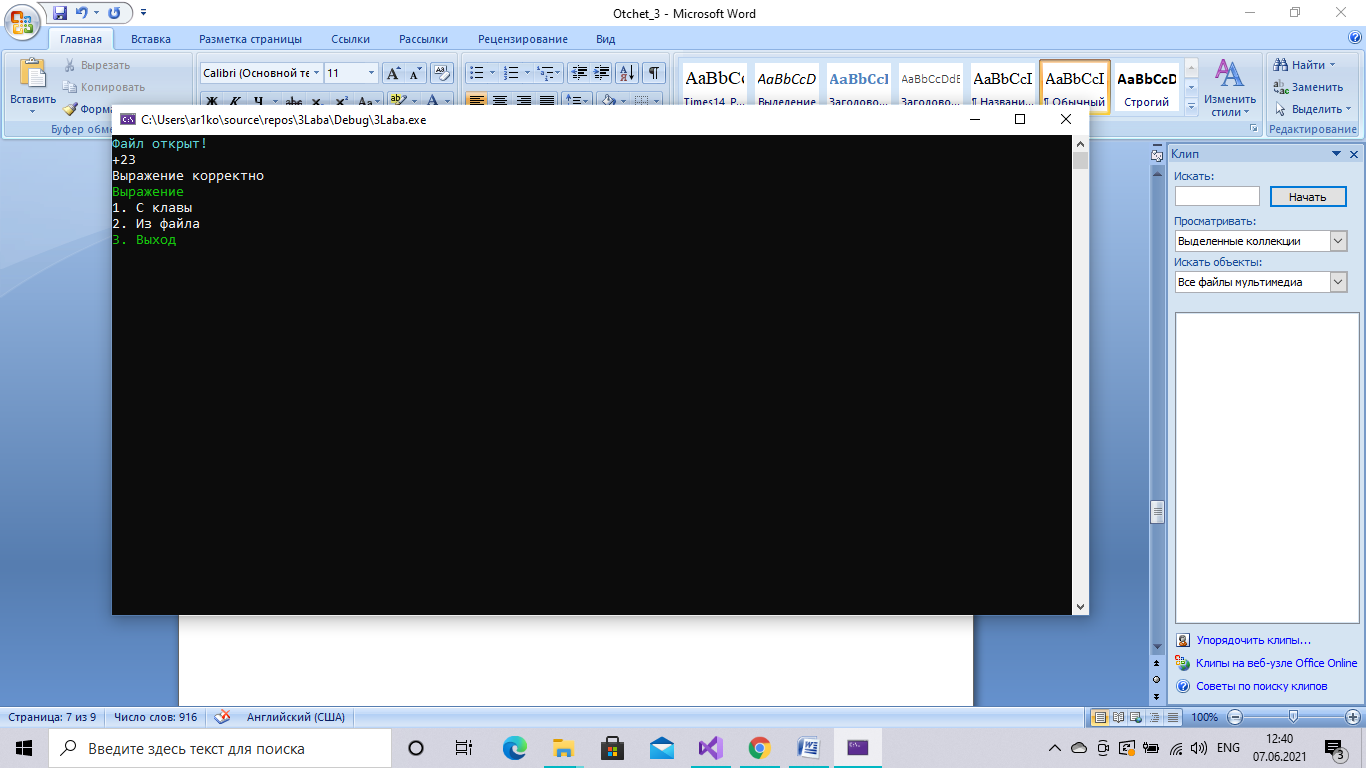
  


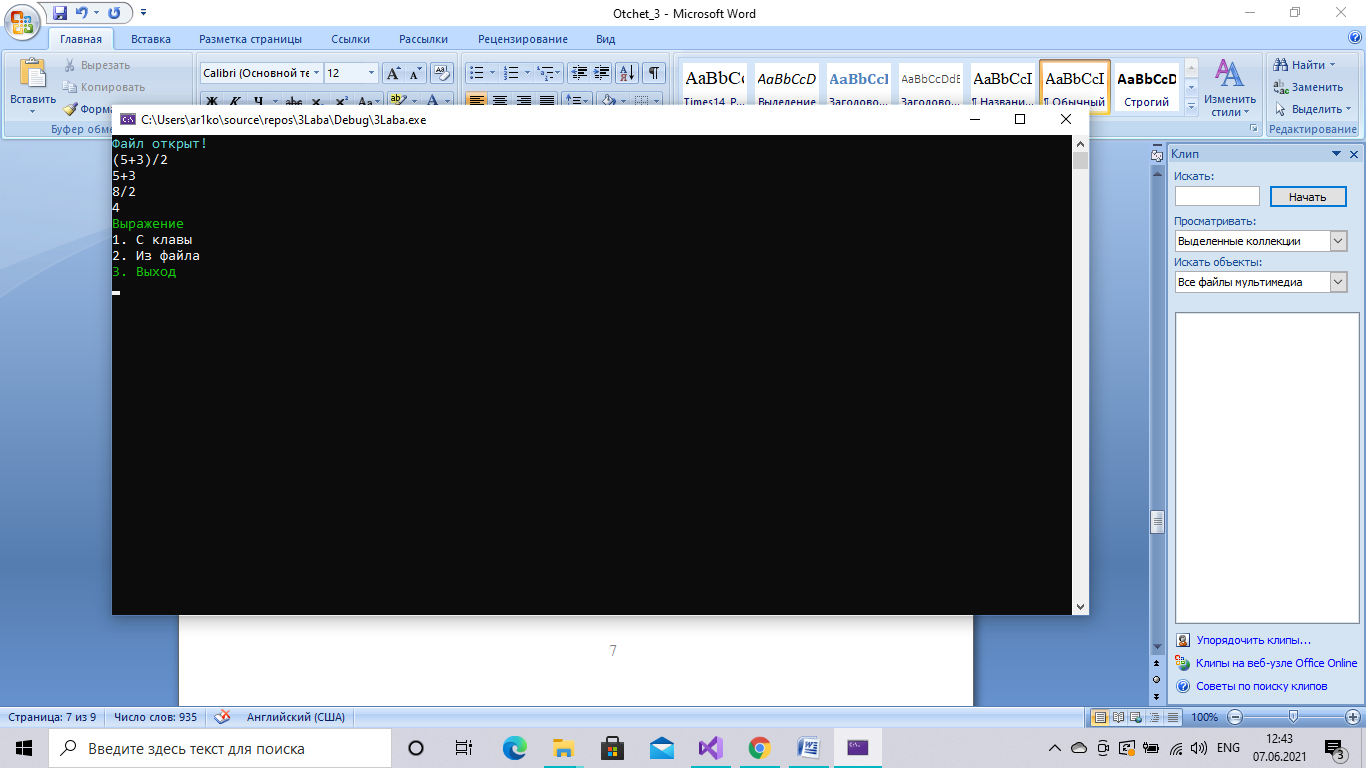
Задание 2

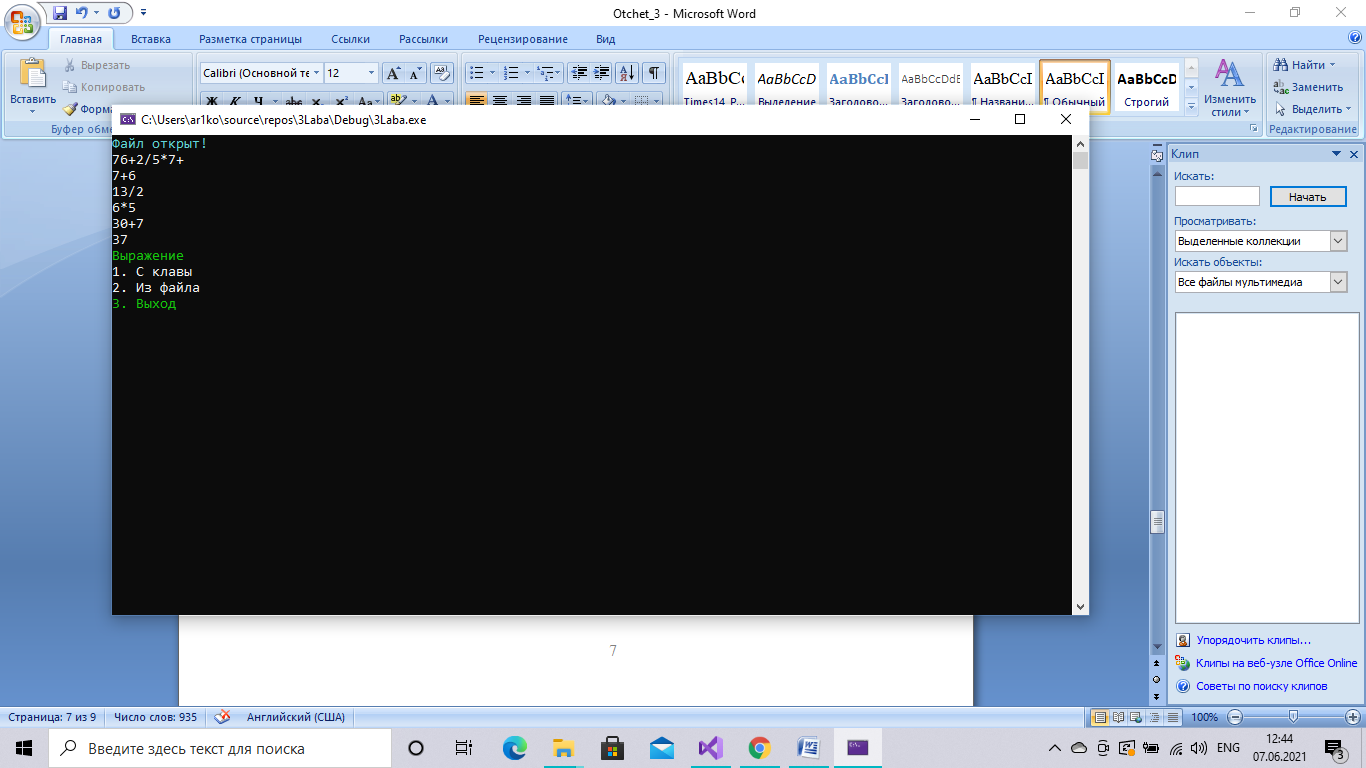
Проверка выражения, записанного в инфиксной нотации  


Проверка выражения, записанного в ОПН  


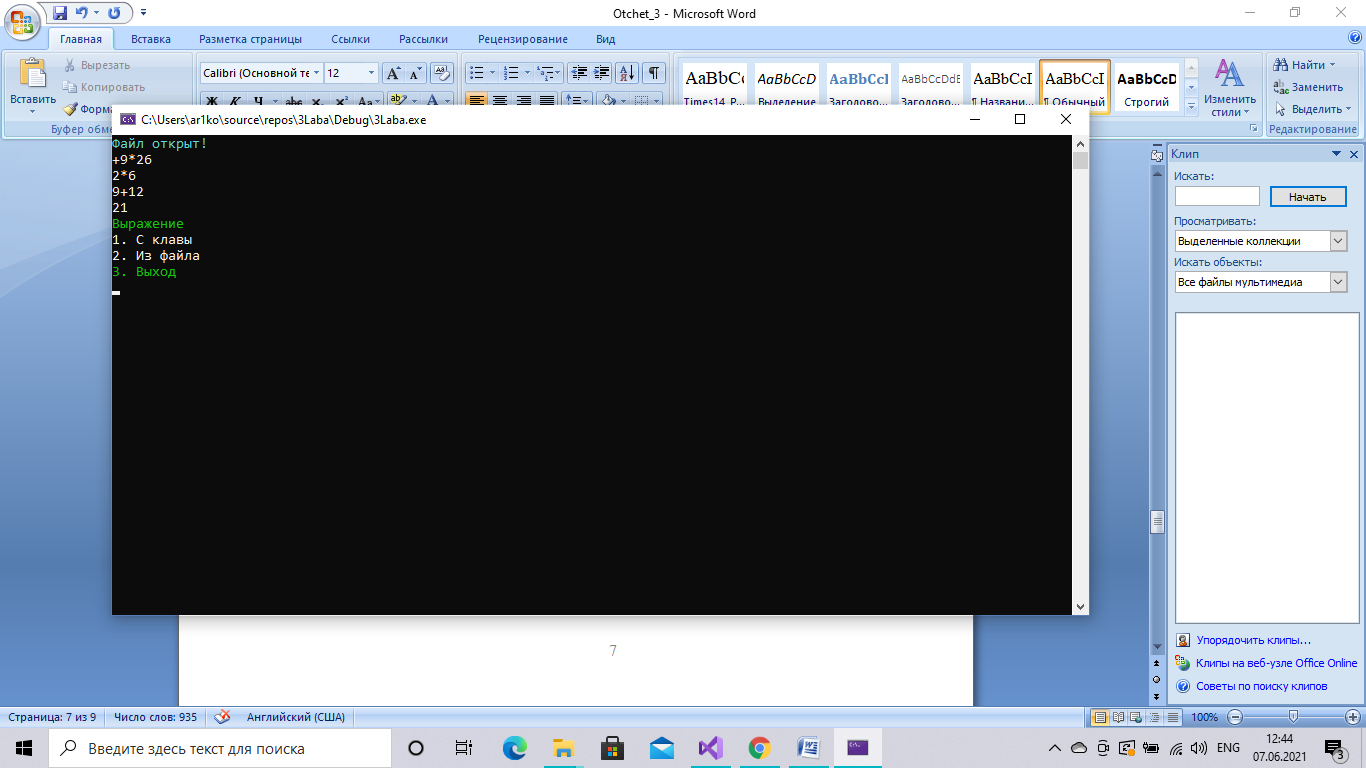
Проверка выражения, записанного в ПН



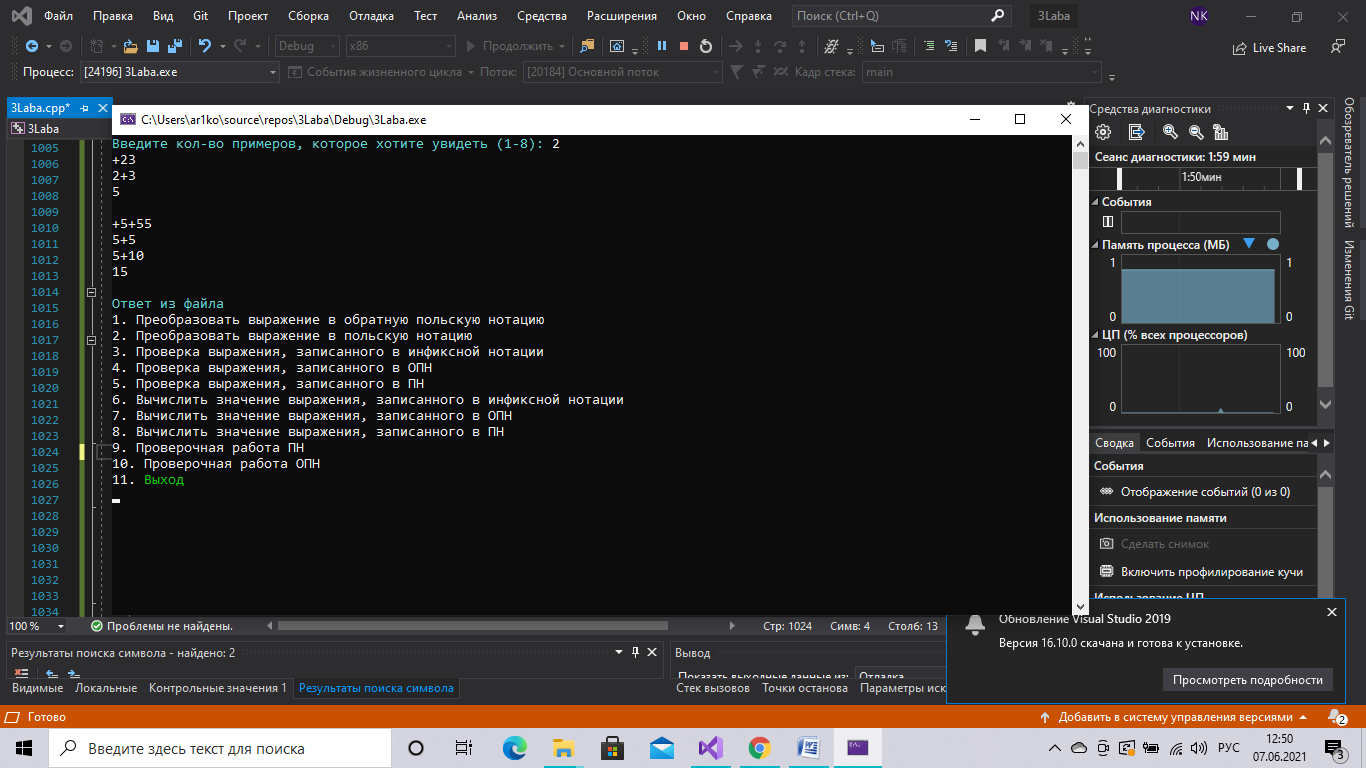
Задание 3  
Вычислить значение выражения, записанного в инфиксной нотации  


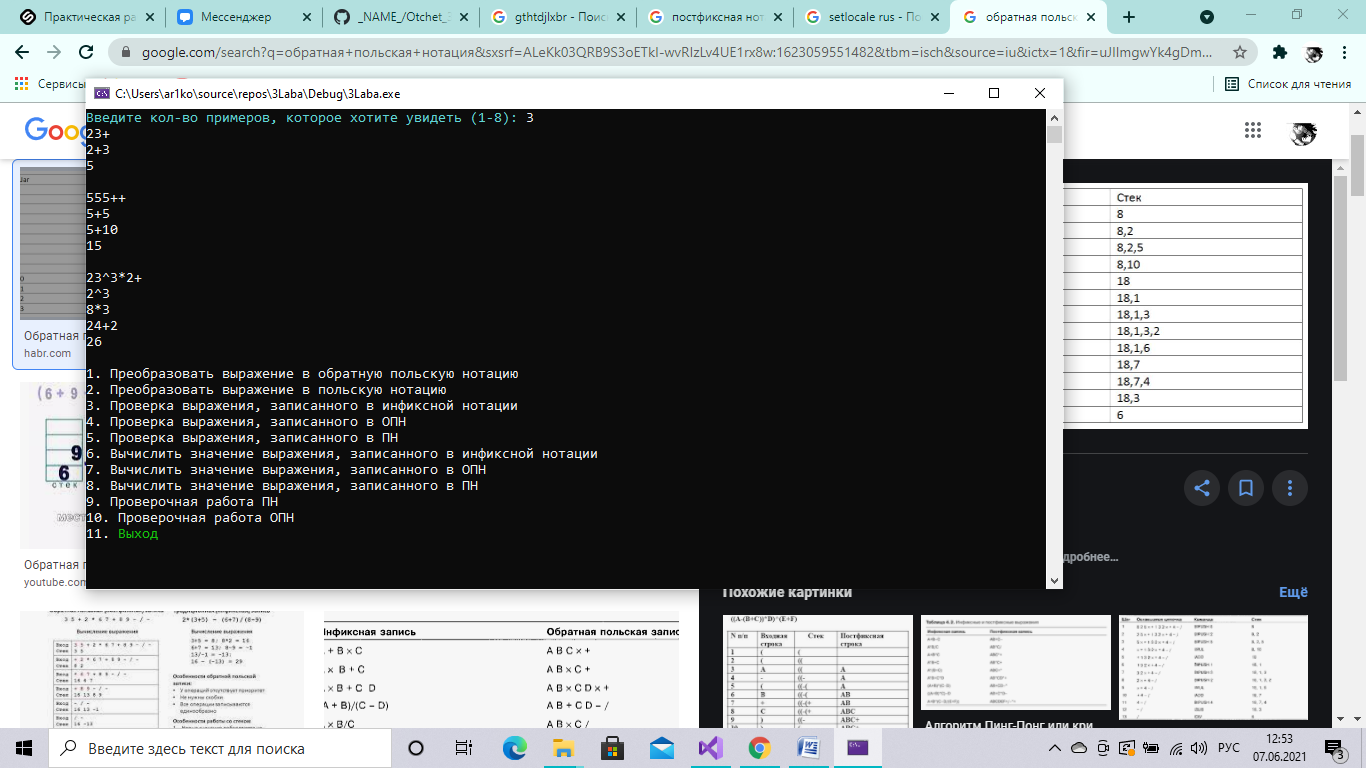
Вычислить значение выражения, записанного в ОПН  


Вычислить значение выражения, записанного в ПН



Задание 4

****

****

**Выводы**.

## Была проведена работа со стеками, изучены прямая и обратная польские нотации.

## КОД

## #include <iostream>

## #include <iterator>

## #include <stack>

## #include <sstream>

## #include <vector>

## #include <fstream>

## #include <Windows.h>

## #include <math.h>

## #include <string>

## #include <cstring>

## #include <Ratings.h>

## #include <chrono>

## #include <cstdlib>

## using namespace std;

## HANDLE N = GetStdHandle(STD\_OUTPUT\_HANDLE);

## struct students

## {

## string fio;

## string pol;

## int group;

## int n\_v\_spiske;

## int ratings[8];

## string forma\_obuch;

## string data\_izm;

## };

## //студенты с индексом i

## void Show(vector<students> people, int n, int i)

## {

## cout << "ФИО:" << people[i].fio << "\n";

## cout << "Номер группы:" << people[i].group << "\n";

## cout << "Номер студента в списке:" << people[i].n\_v\_spiske << "\n";

## cout << "Пол:" << people[i].pol << "\n";

## cout << "Форма обучения:" << people[i].forma\_obuch << "\n";

## cout << "Оценки:";

## for (int j = 0; j < 8; j++) cout << people[i].ratings[j] << " "; cout << "\n";

## cout << "Дата внесения изменнеия в запись:" << people[i].data\_izm << "\n\n";

## }

## //вывод всех студентов

## void VseStudents(vector<students> people, int n)

## {

## int stud;

## cout << "Все студенты \n\n";

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

## {

## stud = i + 1;

## cout << "Студент " << stud << "\n\n";

## cout << "ФИО:" << people[i].fio << "\n";

## cout << "Номер группы:" << people[i].group << "\n";

## cout << "Номер студента в списке:" << people[i].n\_v\_spiske << "\n";

## cout << "Пол:" << people[i].pol << "\n";

## cout << "Форма обучения:" << people[i].forma\_obuch << "\n";

## cout << "Оценки:";

## for (int j = 0; j < 8; j++) cout << people[i].ratings[j] << " "; cout << "\n";

## cout << "Дата и время изменений:" << people[i].data\_izm << "\n\n";

## }

## cout << "\n\n";

## system("pause");

## system("cls");

## }

## //добавление студента

## void NovStud(vector<students>& people, int& n)

## {

## n++;

## people.resize(n);

## int i = n - 1;

## ofstream fi;

## fi.open("students.txt", fstream::app);

## if (!fi.is\_open())

## {

## cout << "\nВ открытии файла произошла ошибка";

## }

## else {

## cout << "Введите фамилию, имя и отчество: ";

## cin.ignore();

## getline(cin, people[i].fio);

## fi << people[i].fio << "\n";

## cout << "Введите номер группы: ";

## cin >> people[i].group;

## fi << people[i].group << "\n";

## cout << "Введите номер в списке: ";

## cin >> people[i].n\_v\_spiske;

## fi << people[i].n\_v\_spiske << "\n";

## cout << "Введите пол: ";

## cin.ignore();

## getline(cin, people[i].pol);

## fi << people[i].pol << "\n";

## cout << "Введите форму обучения: ";

## getline(cin, people[i].forma\_obuch);

## fi << people[i].forma\_obuch << "\n";

## for (int j = 0; j < 8; j++) { cout << "Введите оценки за сессию: "; cin >> people[i].ratings[j]; }

## for (int j = 0; j < 8; j++) fi << people[i].ratings[j] << " ";

## fi << "\n";

## cout << "Введите дату и время изменения: ";

## cin.ignore();

## getline(cin, people[i].data\_izm);

## fi << people[i].data\_izm << "\n";

## }

## fi.close();

## cout << "\n\nRecorded\n\n";

## system("pause");

## system("cls");

## }

## //запись в файл

## void Recording(vector<students>& people, int& n)

## {

## ofstream Rec;

## Rec.open("students.txt");

## if (!Rec.is\_open())

## {

## cout << "\nВ открытии файла произошла ошибка";

## }

## else {

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

## {

## Rec << people[i].fio << "\n";

## Rec << people[i].group << "\n";

## Rec << people[i].n\_v\_spiske << "\n";

## Rec << people[i].pol << "\n";

## Rec << people[i].forma\_obuch << "\n";

## for (int j = 0; j < 8; j++) Rec << people[i].ratings[j] << " ";

## Rec << "\n";

## Rec << people[i].data\_izm << "\n";

## }

## }

## Rec.close();

## }

## //изменение определенных данных

## void Izmenenia(vector<students>& people, int& n)

## {

## int num, choose;

## cout << "Введите номер студента от 1 до " << n << "\n";

## cout << "Вы выбрали ===> ";

## cin >> num;

## num--;

## cout << "\nЧто нужно изменить:\n\n";

## cout << "1 - ФИО\n";

## cout << "2 - номер группы\n";

## cout << "3 - номер в списке\n";

## cout << "4 - пол\n";

## cout << "5 - fформу обучения\n";

## cout << "6 - оценки\n";

## cout << "7 - dдату и время изменений\n";

## cout << "Вы выбрали ====> ";

## cin >> choose;

## cout << "\n";

## switch (choose)

## {

## case 1:

## cout << "Введите ФИО: "; cin.ignore();

## getline(cin, people[num].fio);

## break;

## case 2:

## cout << "ВВедите номер группы: ";

## cin >> people[num].group;

## break;

## case 3:

## cout << "Введите номер в списке: ";

## cin >> people[num].n\_v\_spiske;

## break;

## case 4:

## cout << "Введите пол: "; cin.ignore();

## getline(cin, people[num].pol);

## break;

## case 5:

## cout << "Введите форму обучения: "; cin.ignore();

## getline(cin, people[num].forma\_obuch);

## break;

## case 6:

## for (int j = 0; j < 8; j++) { cout << "Введите оценки: "; cin >> people[num].ratings[j]; }

## break;

## case 7:

## cout << "Введите дату и время: "; cin.ignore();

## getline(cin, people[num].data\_izm);

## break;

## }

## Recording(people, n);

## cout << "\nДанные подверглись изменению\n\n";

## system("pause");

## system("cls");

## }

## //данные студентов определенной группы

## void StudentsVgr(vector<students> people, int n)

## {

## int num, check = 0;

## cout << "Введите номер группы";

## cout << "\nВы выбрали: ";

## cin >> num; cout << "\n";

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

## {

## if (people[i].group == num)

## {

## Show(people, n, i);

## check = 1;

## }

## }

## if (check == 0) cout << "Такой группы нет";

## cout << "\n\n";

## system("pause");

## system("cls");

## }

## //топ 3

## void Rating(vector<students> people, int n)

## {

## double\* rating = new double[n];

## double max1 = 0, max2 = 0, max3 = 0, dub = 0.0;

## int top1 = 0, top2 = 0, top3 = 0;

## for (int i = 0; i < n; i++) //Перебираем студентов

## {

## for (int j = 0; j < 8; j++) //Перебираем оценки

## dub += people[i].ratings[j]; //суммируем оценки

## dub /= 8; //находим сред.значение

## rating[i] = dub;

## dub = 0;

## }

## for (int i = 0; i < n; i++) { if (rating[i] > max1) { max1 = rating[i]; top1 = i; } }

## for (int i = 0; i < n; i++) { if (rating[i] > max2 && people[i].fio != people[top1].fio) { max2 = rating[i]; top2 = i; } }

## for (int i = 0; i < n; i++) { if (rating[i] > max3 && people[i].fio != people[top1].fio && people[i].fio != people[top2].fio) { max3 = rating[i]; top3 = i; } }

## cout << "Первое место: " << max1 << "\n"; Show(people, n, top1);

## cout << "Второе место: " << max2 << "\n"; Show(people, n, top2);

## cout << "\nТретье место: " << max3 << "\n"; Show(people, n, top3);

## delete[] rating;

## cout << "\n\n";

## system("pause");

## system("cls");

## }

## //количество мужчин и женщин

## void MF(vector<students> people, int n)

## {

## int countM = 0, countF = 0;

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

## {

## if (people[i].pol == "F")

## countF++; //счетчик женщин

## if (people[i].pol == "M")

## countM++;//счетчик мужчин

## }

## cout << "\nКоличество женского пола: " << countF << "\n";

## cout << "Количество мужского пола: " << countM << "\n";

## system("pause");

## system("cls");

## }

## //количество степендиатов

## void Stepuxa(vector<students> people, int n)

## {

## int check = 0, count = 0;

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

## {

## for (int j = 0; j < 8; j++)

## if (people[i].ratings[j] == 3 || people[i].ratings[j] == 2) check = 1; //проверка на оценки выше 3

## if (people[i].forma\_obuch == "day" && check == 0) count++; //Проверка на очное обучение и оценки выше 4

## check = 0;

## }

## cout << "\nКол. студентов со степендией: " << count << "\n\n\n";

## system("pause");

## system("cls");

## }

## //оценки и стипендии

## void BezStepuxi(vector<students> people, int n)

## {

## int uslov = 0, count = 0, otlichnik = 0, ocenka\_5 = 0, ocenka\_4 = 0;

## cout << "Кол. студентов, которые не получат стипендию:\n\n";

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

## {

## for (int j = 0; j < 8; j++)

## if (people[i].ratings[j] == 3 || people[i].ratings[j] == 2)

## uslov = 1; //Перебираем студентов, которые имеют оценки 2 или 3

## if (people[i].forma\_obuch == "evening" || uslov == 1) //Перебираем людей, которые на вечерке или имеют 2 или 3

## {

## Show(people, n, i);

## }

## uslov = 0;

## }

## system("pause"); system("cls");

## cout << "Студенты с оценкой 4 или 5:\n\n";

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

## {

## for (int j = 0; j < 8; j++)

## {

## if (people[i].ratings[j] == 2) uslov = 1;

## if (people[i].ratings[j] == 3) uslov = 1;

## if (people[i].ratings[j] == 5) ocenka\_5++; //Ведем счет 5

## if (people[i].ratings[j] == 4) ocenka\_4++;//Ведем счет 4

## }

## if (uslov == 0 && ocenka\_5 < 8 && ocenka\_4 > 0) //Проверка наличия 5 и 4

## {

## Show(people, n, i);

## }

## ocenka\_5 = 0;

## uslov = 0;

## }

## system("pause"); system("cls");

## cout << "Люди, которые учатся только на 5:\n";

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

## {

## for (int j = 0; j < 8; j++)

## if (people[i].ratings[j] == 5) otlichnik++;

## if (otlichnik == 8)

## {

## Show(people, n, i);

## }

## otlichnik = 0;

## }

## cout << "\n\n";

## system("pause");

## system("cls");

## }

## //данные студентов с определенным номером

## void NomerVSpiske(vector<students> people, int n)

## {

## int num, check = 0;

## cout << "Введите номер студента: ";

## cin >> num; cout << "\n";

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

## {

## if (people[i].n\_v\_spiske == num) //Перебор всех студентов с таким же номером

## {

## Show(people, n, i);

## check = 1;

## }

## }

## if (check == 0) cout << "Таких студентов нет";

## cout << "\n\n";

## system("pause");

## system("cls");

## }

## //дата

## void Time(vector<students> people, int n)

## {

## string t[12] = { "12","13","14","15","16","17","18","19","20","21","22","23" };

## string t\_[12] = { "00","01","02","03","04","05","06","07","08","09","10","11" };

## string sub, dub, time\_; int p, check = 0;

## cout << "Введите день недели, месяц и время изменений данных: ";

## cin.ignore();

## getline(cin, time\_);

## for (int i = 0; i < n; i++)//Перебираем людей

## {

## sub = people[i].data\_izm;//Смотрим на дату изменений человека

## if (sub.find\_first\_not\_of(time\_) >= 10)//Проверка на символы с строке

## {

## Show(people, n, i);

## check = 1;

## }

## }

## if (check == 0) cout << "Изменений в эту дату не было\n\n";

## system("pause"); system("cls");

## cout << "Изменения после полудня\n\n";

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

## {

## dub = people[i].data\_izm;//смотрим на даты измнений

## if (dub.size() == 24) //смотрим на размер строчки

## p = 11;

## else

## p = 10;

## for (int j = 0; j < 12; j++)

## {

## if (t[j] == dub.substr(p, 2)) //возвращает подстроку строки string длиной length , начинающегося с start символа по счету. Если данные подходят к допустимым значениям t, то выводим

## {

## Show(people, n, i);

## }

## }

## }

## system("pause"); system("cls");

## cout << "Записи до полудня\n";

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

## {

## dub = people[i].data\_izm;

## if (dub.size() >= 24)

## p = 11;

## else

## p = 10;

## for (int j = 0; j < 12; j++)

## {

## if (t\_[j] == dub.substr(p, 2))//рассматриваем относительно строчки t\_

## {

## Show(people, n, i);

## }

## }

## }

## cout << "\n\n";

## system("pause");

## system("cls");

## }

## struct node

## {

## int data;

## struct node\* prev;

## struct node\* next;

## };

## struct node\* head = NULL;

## struct node\* last = NULL;

## struct node\* current = NULL;

## void display()

## {

## struct node\* ptr = head;

## while (ptr != NULL)

## {

## cout << ptr->data << " ";

## ptr = ptr->next;

## }

## cout << endl;

## }

## void insert(int data)

## {

## struct node\* link = (struct node\*)malloc(sizeof(struct node));

## link->data = data;

## link->prev = NULL;

## link->next = NULL;

## if (head == NULL)

## {

## head = link;

## return;

## }

## current = head;

## while (current->next != NULL)

## current = current->next;

## current->next = link;

## last = link;

## link->prev = current;

## }

## void Mas()

## {

## int variant;

## do

## {

## cout << "Введите вариант заполнения массива\n";

## cout << "1.Ввод размерности и автоматическое заполнение рандомными числами\n";

## cout << "2.Ввод чисел и подсчёт размерности\n";

## cout << "3.Считывание с файла\n";

## cout << "4.Выход\n";

## cout << "\n";

## cout << "Вы выбрали ====> ";

## cin >> variant;

## system("cls");

## switch (variant)

## {

## case 1:

## {

## int kol;

## cout << "Введите размер массива ====>";

## cin >> kol;

## if (kol == 0)

## cout << "В массиве нет элементов";

## int\* mas = new int[kol];

## for (int i = 0; i < kol; i++)

## {

## mas[i] = 0 + rand() % 99;

## cout << mas[i] << " ";

## }

## cout << "\n";

## delete[] mas;

## break;

## }

## case 2:

## {

## int\* arr = new int[0];

## int\* temp;

## int n = 0, num;

## std::cout << "!!!Введите элементы списка через пробел (закончите ввод любой буквой)!!!: " << '\n';

## std::cin >> num;

## while (!std::cin.fail()) {

## temp = new int[n + 1];

## for (int i = 0; i != n; i++) {

## temp[i] = arr[i];

## }

## temp[n] = num;

## delete[] arr;

## arr = temp;

## n++;

## std::cin >> num;

## }

## cout << "Вы ввели: ";

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

## {

## cout << arr[i] << " ";

## }

## if (n == 0)

## cout << "В массиве нет элементов";

## else

## cout << "\nРазмерность введенного массива " << n;

## cout << "\n";

## std::cin.clear();

## std::cin.ignore(32767, '\n');

## break;

## }

## case 3:

## {

## string arr[9];

## ifstream file("massiv.txt");

## if (file.is\_open())

## {

## for (int i = 0; i < 9; ++i)

## {

## file >> arr[i];

## }

## }

## for (int i = 0; i < 9; ++i)

## {

## cout << arr[i] << " ";

## }

## cout << endl;

## break;

## }

## case 4:

## break;

## }

## } while (variant != 4);

## }

## void deleteAllNodes() {

## node\* temp = new node();

## while (head != NULL) {

## temp = head;

## head = head->next;

## delete[] temp;

## }

## }

## void Delete(int l)//Удаление по значению

## {

## node\* temp = head;

## while ((temp != NULL) && (temp->data != l))

## temp = temp->next;

## if (temp == NULL)

## return;

## node\* prev = temp->prev;

## node\* next = temp->next;

## if (prev != NULL)

## prev->next = next;

## if (next != NULL)

## next->prev = prev;

## delete temp;

## if (prev == NULL)

## {

## cout << "Ошибка: Такого элемента нет в массиве\n";

## display();

## deleteAllNodes();

## }

## else

## { // found match

## cout << "\n";

## display();

## deleteAllNodes();

## }

## }

## int displayEle(struct node\* head, int x)

## {

## if (head == NULL)

## return -1;

## if (x == 0)

## return head->data;

## return displayEle(head->next, x - 1);

## }

## int search(node\* head, int x)

## {

## node\* temp = head;

## int pos = 0;

## while (temp->data != x

## && temp->next != NULL) {

## pos++;

## temp = temp->next;

## }

## if (temp->data != x)

## return -1;

## return (pos + 1);

## }

## void addElement(node\*& head, int newdata)

## {

## //Вставка узла в конец списка не должна вызывать затруднений: необходимо изменить

## //Tail-указатель прежнего последнего элемента на адрес нового узла, затем правильно

## //связать новый узел с двусвязным списком

## node\* tmp = new node;

## tmp->data = newdata;

## tmp->next = NULL;

## tmp->prev = NULL;

## if (!head)

## head = tmp;

## else

## {

## node\* last = head;

## while (last->next != NULL)

## last = last->next;

## last->next = tmp;

## tmp->prev = last;

## }

## }

## //для удаления первого элемента

## void delFirst()

## {

## struct node\* NodeToDel;

## NodeToDel = head;

## head = head->next;

## head->prev = NULL;

## free(NodeToDel);

## }

## //для удаления последнего элемента

## void delLast()

## {

## struct node\* NodeToDel;

## NodeToDel = last;

## last = last->prev;

## last->next = NULL;

## free(NodeToDel);

## }

## //удаление элементов по индексу

## void deleteNode(int pos)

## {

## struct node\* curNode;

## int i;

## curNode = head;

## for (i = 1; i < pos && curNode != NULL; i++)

## {

## curNode = curNode->next;

## }

## if (pos == 1)

## {

## delFirst();

## }

## else if (curNode == last)

## {

## delLast();

## }

## else if (curNode != NULL)

## {

## curNode->prev->next = curNode->next;

## curNode->next->prev = curNode->prev;

## free(curNode);

## }

## }

## bool bb(char c)

## {

## return (!isalpha(c) && !isdigit(c));

## }

## int prioritet(char C)

## {

## if (C == '-' || C == '+')

## return 1;

## else if (C == '\*' || C == '/')

## return 2;

## else if (C == '^')

## return 3;

## return 0;

## }

## string infixToPostfix(string infix)

## {

## infix = '(' + infix + ')';

## int l = infix.size();

## stack<char> st;

## string output;

## for (int i = 0; i < l; i++) {

## if (isalpha(infix[i]) || isdigit(infix[i]))

## output += infix[i];

## else if (infix[i] == '(')

## st.push('(');

## else if (infix[i] == ')') {

## while (st.top() != '(') {

## output += st.top();

## st.pop();

## }

## st.pop();

## }

## else

## {

## if (bb(st.top()))

## {

## if (infix[i] == '^')

## {

## while (prioritet(infix[i]) <= prioritet(st.top()))

## {

## output += st.top();

## st.pop();

## }

## }

## else

## {

## while (prioritet(infix[i]) < prioritet(st.top()))

## {

## output += st.top();

## st.pop();

## }

## }

## st.push(infix[i]);

## }

## }

## }

## return output;

## }

## string infixToPrefix(string infix)

## {

## int l = infix.size();

## reverse(infix.begin(), infix.end());

## for (int i = 0; i < l; i++) {

## if (infix[i] == '(') {

## infix[i] = ')';

## i++;

## }

## else if (infix[i] == ')') {

## infix[i] = '(';

## i++;

## }

## }

## string prefix = infixToPostfix(infix);

## reverse(prefix.begin(), prefix.end());

## return prefix;

## }

## int isPostfix(string c) {

## int len = c.length();

## if ((c[len - 1] == ')') || (c[len - 1] == '(')) {

## return 1;

## }

## if (bb(c[len - 1]) == 0) {

## return 1;

## }

## return -1;

## }

## int isPrefix(string c) {

## if ((c[0] == ')') || (c[0] == '(')) {

## return 1;

## }

## if (bb(c[0]) == 0) {

## return 1;

## }

## return -1;

## }

## int evaluatePostfixFile(string s) {

## stack<int> st;

## for (int i = 0; i < s.length(); i++) {

## if (!bb(s[i])) {

## st.push(s[i] - '0');

## }

## else {

## int op2 = st.top();

## st.pop();

## int op1 = st.top();

## st.pop();

## switch (s[i])

## {

## case '+':

## st.push(op1 + op2);

## break;

## case '-':

## st.push(op1 - op2);

## break;

## case '\*':

## st.push(op1 \* op2);

## break;

## case '/':

## st.push(op1 / op2);

## break;

## case '^':

## st.push(pow(op1, op2));

## break;

## }

## }

## }

## return st.top();

## }

## int evaluatePostfix(string s) {

## stack<int> st;

## for (int i = 0; i < s.length(); i++) {

## if (!bb(s[i])) {

## st.push(s[i] - '0');

## }

## else {

## int op2 = st.top();

## st.pop();

## int op1 = st.top();

## st.pop();

## switch (s[i])

## {

## case '+':

## st.push(op1 + op2);

## cout << op1;

## cout << "+";

## cout << op2 << endl;

## break;

## case '-':

## st.push(op1 - op2);

## cout << op1;

## cout << "-";

## cout << op2 << endl;

## break;

## case '\*':

## st.push(op1 \* op2);

## cout << op1;

## cout << "\*";

## cout << op2 << endl;

## break;

## case '/':

## st.push(op1 / op2);

## cout << op1;

## cout << "/";

## cout << op2 << endl;

## break;

## case '^':

## st.push(pow(op1, op2));

## cout << op1;

## cout << "^";

## cout << op2 << endl;

## break;

## }

## }

## }

## return st.top();

## }

## double evaluatePrefixFile(string s)

## {

## stack<int> st;

## for (int i = s.size() - 1; i >= 0; i--) {

## if (!bb(s[i]))

## st.push(s[i] - '0');

## else {

## double op1 = st.top();

## st.pop();

## double op2 = st.top();

## st.pop();

## switch (s[i]) {

## case '+':

## st.push(op1 + op2);

## break;

## case '-':

## st.push(op1 - op2);

## break;

## case '\*':

## st.push(op1 \* op2);

## break;

## case '/':

## st.push(op1 / op2);

## break;

## case '^':

## st.push(pow(op1, op2));

## break;

## }

## }

## }

## return st.top();

## }

## double evaluatePrefix(string s)

## {

## stack<int> st;

## for (int i = s.size() - 1; i >= 0; i--) {

## if (!bb(s[i]))

## st.push(s[i] - '0');

## else {

## double op1 = st.top();

## st.pop();

## double op2 = st.top();

## st.pop();

## switch (s[i]) {

## case '+':

## st.push(op1 + op2);

## cout << op1;

## cout << "+";

## cout << op2 << endl;

## break;

## case '-':

## st.push(op1 - op2);

## cout << op1;

## cout << "-";

## cout << op2 << endl;

## break;

## case '\*':

## st.push(op1 \* op2);

## cout << op1;

## cout << "\*";

## cout << op2 << endl;

## break;

## case '/':

## st.push(op1 / op2);

## cout << op1;

## cout << "/";

## cout << op2 << endl;

## break;

## case '^':

## st.push(pow(op1, op2));

## cout << op1;

## cout << "^";

## cout << op2 << endl;

## break;

## }

## }

## }

## return st.top();

## }

## int main()

## {

## int cursach = 0;

## int answer = 0;

## setlocale(LC\_ALL, "Russian");

## while (cursach != 4)

## {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "\n МЕНЮ" ;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "\n1-1 Лабараторная работа ";

## cout << "\n2-2 Лабараторная работа ";

## cout << "\n3-3 Лабараторная работа ";

## cout << "\n4-";

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Выход" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Введите номер работы ====> ";

## cin >> cursach;

## system("CLS");

## switch (cursach)

## {

## case 1:

## {

## ifstream fin;

## fin.open("students.txt");

## int n = 0, t = 0, task;

## while (!fin.eof()) {

## if (fin.get() == '\n')

## n++;

## }

## n = n / 7;// количество студентов

## vector<students>people(n);

## ifstream f;

## f.open("students.txt");

## if (!f.is\_open())

## {

## cout << "\nFile opening error";

## }

## else {

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

## {

## getline(f, people[i].fio);

## f >> people[i].group; f.ignore();

## f >> people[i].n\_v\_spiske; f.ignore();

## getline(f, people[i].pol);

## getline(f, people[i].forma\_obuch);

## for (int j = 0; j < 8; j++) { f >> people[i].ratings[j]; }

## f.ignore(2);

## getline(f, people[i].data\_izm);

## }

## }

## f.close();

## do

## {

## cout << "Введите номер задания:\n";

## cout << "1.Создание новой записи о студенте\n";

## cout << "2.Внесение изменений в уже имеющуюся запись\n";

## cout << "3.Вывод всех данных о студентах\n";

## cout << "4.Вывод информации обо всех студентах группы\n";

## cout << "5.Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию\n";

## cout << "6.Вывод количества студентов мужского и женского пола\n";

## cout << "7.Определение количества студентов, которые будут получать стипендию\n";

## cout << "8.Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»\n";

## cout << "9.Вывод данных о студентах, имеющих номер в списке – k\n";

## cout << "10. Вывод всех записей, сделанных в день, который введет пользователь. Вывод всех записей, сделанных после полудня. Вывод всех записей, сделанных до полудня\n";

## cout << "11.Выход\n";

## cout << "\n";

## cout << "Вы выбрали ====> ";

## cin >> task;

## system("cls");

## switch (task)

## {

## case 1:

## NovStud(people, n);

## break;

## case 2:

## Izmenenia(people, n);

## break;

## case 3:

## VseStudents(people, n);

## break;

## case 4:

## StudentsVgr(people, n);

## break;

## case 5:

## Rating(people, n);

## break;

## case 6:

## MF(people, n);

## break;

## case 7:

## Stepuxa(people, n);

## break;

## case 8:

## BezStepuxi(people, n);

## break;

## case 9:

## NomerVSpiske(people, n);

## break;

## case 10:

## Time(people, n);

## break;

## case 11:

## break;

## }

## } while (task != 11);

## break;

## }

## case 2:

## {

## std::chrono::system\_clock::time\_point start;

## std::chrono::system\_clock::time\_point end;

## std::chrono::duration <double> time;

## int task, nom, nam;

## int kol = 0;

## setlocale(LC\_ALL, "Russian");

## do

## {

## cout << "Введите номер задания:\n";

## cout << "1.Формирование целочисленного одномерного массива размерности N\n";

## cout << "2.Время создания массива\n";

## cout << "3.Вставка, удаление и получение элемента массива + определение времени\n";

## cout << "4.Формирование двусвязного списка по вариантам + определение времени\n";

## cout << "5.Вставка, удаление и получение элемента двусвязного списка + определение времени\n";

## cout << "6.Выход\n";

## cout << "\n";

## cout << "Вы выбрали ====> ";

## cin >> task;

## system("cls");

## switch (task)

## {

## case 1:

## Mas();

## break;

## case 2:

## {

## do

## {

## cout << "Нужно померить время создания:\n";

## cout << "1.Рандомного массива\n";

## cout << "2.Массива пользователя\n";

## cout << "3.Массива из файла\n";

## cout << "4.Выход\n";

## cout << "\n";

## cout << "Вы выбрали ====> ";

## cin >> nom;

## system("cls");

## switch (nom)

## {

## case 1:

## {

## start = std::chrono::system\_clock::now();

## int kol;

## cout << "Введите размер массива ====>";

## cin >> kol;

## if (kol == 0)

## cout << "В массиве нет элементов";

## int\* mas = new int[kol];

## for (int i = 0; i < kol; i++)

## {

## mas[i] = 0 + rand() % 99;

## cout << mas[i] << " ";

## }

## cout << "\n";

## delete[] mas;

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "Время на создание массива: " << time.count() << '\n';

## break;

## }

## case 2:

## {

## start = std::chrono::system\_clock::now();

## int\* arr = new int[0];

## int\* temp;

## int n = 0, num;

## std::cout << "!!!Введите элементы списка через пробел (закончите ввод любой буквой)!!!: " << '\n';

## std::cin >> num;

## while (!std::cin.fail()) {

## temp = new int[n + 1];

## for (int i = 0; i != n; i++) {

## temp[i] = arr[i];

## }

## temp[n] = num;

## delete[] arr;

## arr = temp;

## n++;

## std::cin >> num;

## }

## cout << "Вы ввели: ";

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

## {

## cout << arr[i] << " ";

## }

## cout << "\n";

## std::cin.clear();

## std::cin.ignore(32767, '\n');

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "Время на создание массива: " << time.count() << '\n';

## break;

## }

## case 3:

## {

## start = std::chrono::system\_clock::now();

## string arr[9];

## ifstream file("massiv.txt");

## if (file.is\_open())

## {

## for (int i = 0; i < 9; ++i)

## {

## file >> arr[i];

## }

## }

## for (int i = 0; i < 9; i++)

## {

## cout << arr[i] << " ";

## }

## cout << "\n";

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "Время на создание массива: " << time.count() << '\n';

## break;

## }

## case 4:

## break;

## }

## } while (nom != 4);

## }

## case 3:

## {

## int k, l;

## do

## {

## cout << "\nВыберите задание :\n";

## cout << "1.Вставка элемента массива + определение времени\n";

## cout << "2.Удаление элемента массива + определение времени\n";

## cout << "3.Получение элемента массива + определение времени\n";

## cout << "4.Выход\n";

## cout << "\n";

## cout << "Вы выбрали ====> ";

## cin >> nam;

## system("cls");

## switch (nam)

## {

## case 1:

## {

## int kol;

## cout << "Введите размер массива ====>";

## cin >> kol;

## if (kol == 0)

## cout << "В массиве нет элементов";

## int\* mas = new int[kol];

## for (int i = 0; i < kol; i++)

## {

## mas[i] = 0 + rand() % 99;

## cout << mas[i] << " ";

## }

## cout << "\n";

## int vstavka;

## cout << "Введите элемент, который хотите вставить===> ";

## cin >> vstavka;

## start = std::chrono::system\_clock::now();

## cout << "\n";

## int element = kol + 1;

## int\* arr = new int[element];

## for (int i = 0; i < kol; i++)

## {

## arr[i] = mas[i];

## }

## delete[] mas;

## mas = arr;

## arr[kol] = vstavka;

## cout << " Итоговый массив:\n ";

## for (int i = 0; i < element; i++)

## {

## cout << arr[i] << " ";

## }

## cout << endl;

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "Время на вставку элемента: " << time.count() << '\n';

## delete[] arr;

## break;

## }

## case 2:

## {

## do

## {

## cout << "\n1.По индексу \n";

## cout << "2.По значению\n";

## cout << "3.Выход\n";

## cout << "Вы выбрали ====> ";

## cin >> k;

## system("cls");

## switch (k)

## {

## case 1:

## {

## cout << "Введите размерность массива===> ";

## int n;

## cin >> n;

## if (n == 0)

## {

## cout << "В массиве ноль элементов";

## }

## else

## {

## int\* arr = new int[n];

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

## {

## arr[i] = 0 + rand() % 99;

## }

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

## {

## cout << arr[i] << " ";

## }

## cout << endl;

## int\* massiv = new int[n - 1];

## int index = 0;

## cout << "Напишите позицию элемента для удаления===> ";

## cin >> index;

## start = std::chrono::system\_clock::now();

## if ((index > n) || (index < 0))

## {

## cout << "!!!Неверный индекс!!!\n ";

## delete[] arr;

## delete[] massiv;

## }

## else

## if ((index <= n) && (index > 0))

## {

## index--;

## for (int i = 0; i < index; i++)

## {

## massiv[i] = arr[i];

## }

## for (int i = index + 1; i < n; i++)

## {

## massiv[i - 1] = arr[i];

## }

## delete[] arr;

## arr = massiv;

## for (int i = 0; i < n - 1; i++)

## {

## cout << massiv[i] << " ";

## }

## cout << endl;

## n--;

## if (n == 0)

## {

## cout << endl;

## cout << "Нет элементов для удаления" << endl;

## cout << endl;

## delete[] massiv;

## break;

## }

## cout << endl;

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "Время на удаление элемента: " << time.count() << '\n';

## }

## }

## break;

## }

## case 2:

## {

## cout << "Введите размерность массива===> ";

## int n;

## cin >> n;

## if (n == 0)

## {

## cout << "В массиве ноль элементов";

## }

## int\* arr = new int[n];

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

## arr[i] = 0 + rand() % 99;

## }

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

## cout << arr[i] << " ";

## }

## cout << endl;

## int\* massiv = new int[n - 1];

## int znachenie = 0;

## cout << "Введите значение элемента, который хотите удалить===> ";

## cin >> znachenie;

## start = std::chrono::system\_clock::now();

## int rast = distance(arr, find(arr, arr + n, znachenie));

## //std::distance - это функция объединения интерфейсов, цель которой - позволить

## //вычислить расстояния между различными типами итераторов: произвольный, двунаправленный,

## //прямой и т. Д. Эта функция предназначена для сокрытия неэффективности прямого вычисления

## //расстояния для итераторы с неслучайным доступом в ситуациях, когда вы действительно знаете,

## //что делаете, и действительно хотите принять эту неэффективность.

## if (rast != n)

## {

## int\* mass = new int[n - 1];

## for (int i = 0; i < rast; i++)

## {

## mass[i] = arr[i];

## }

## for (int i = rast + 1; i < n; i++)

## {

## mass[i - 1] = arr[i];

## }

## delete[] arr;

## arr = mass;

## cout << "Массив после удаления элемента\n";

## for (int i = 0; i < n - 1; i++)

## {

## cout << mass[i] << " ";

## }

## cout << endl;

## n--;

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя на удаление элемента: " << time.count() << '\n';

## }

## break;

## }

## case 3:

## break;

## }

## } while (k != 3);

## break;

## }

## case 3:

## {

## do

## {

## cout << "\n1.По индексу + время\n";

## cout << "2.По значению + время\n";

## cout << "3.Выход\n";

## cout << "Вы выбрали ====> ";

## cin >> l;

## system("cls");

## switch (l)

## {

## case 1:

## {

## cout << "Введите размерность массива===> ";

## int n;

## cin >> n;

## if (n == 0)

## {

## cout << "В массиве ноль элементов";

## }

## int\* arr = new int[n];

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

## arr[i] = 0 + rand() % 99;

## }

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

## cout << arr[i] << " ";

## }

## cout << "\nВведите индекс для получения элемента===> ";

## int ind;

## cin >> ind;

## start = std::chrono::system\_clock::now();

## if ((ind > 0) && (ind <= n))

## {

## cout << "\nВаш элемент ===> " << arr[ind - 1] << endl;

## delete[] arr;

## }

## else

## {

## cout << "\nПРОВЕРЬТЕ ВВОД\n";

## delete[] arr;

## break;

## }

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя на поиск элемента: " << time.count() << '\n';

## break;

## }

## case 2:

## {

## cout << "Введите размерность массива===> ";

## int n;

## bool l = 0;

## cin >> n;

## if (n == 0)

## {

## cout << "В массиве ноль элементов";

## }

## int\* arr = new int[n];

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

## arr[i] = 0 + rand() % 99;

## }

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

## cout << arr[i] << " ";

## }

## cout << "\nВведите значение элемента===> ";

## int znach;

## cin >> znach;

## start = std::chrono::system\_clock::now();

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

## {

## if (arr[i] == znach)

## {

## cout << "Индекс этого элемента равен " << i + 1 << "\n";

## l = 1;

## }

## }

## if (l == 0)

## {

## cout << "\n!!!Такого элемента нет!!!\n ";

## break;

## }

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя на поиск элемента: " << time.count() << '\n';

## delete[] arr;

## break;

## }

## case 3:

## break;

## }

## } while (l != 3);

## break;

## }

## case 4:

## break;

## }

## } while (nam != 4);

## }

## case 4:

## {

## int k;

## do

## {

## cout << "\nВыберите задание :\n";

## cout << "1.Рандомный массив размерности N + время создания \n";

## cout << "2.Массив пользователя + определение времени\n";

## cout << "3.Получение массива из файла + определение времени\n";

## cout << "4.Выход\n";

## cout << "\n";

## cout << "Вы выбрали ====> ";

## cin >> k;

## system("cls");

## switch (k)

## {

## case 1:

## {

## int n;

## cout << "Введите кол-во элементов ===> ";

## start = std::chrono::system\_clock::now();

## cin >> n;

## cout << endl;

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

## {

## insert(0 + rand() % 99);

## }

## display();

## cout << endl;

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя на создание рандомного массива: " << time.count() << '\n';

## deleteAllNodes();

## break;

## }

## case 2:

## {

## int\* arr = new int[0];

## int\* temp;

## int n = 0, num;

## std::cout << "!!!Введите элементы списка через пробел (закончите ввод любой буквой)!!!: " << '\n';

## start = std::chrono::system\_clock::now();

## std::cin >> num;

## while (!std::cin.fail()) {

## temp = new int[n + 1];

## for (int i = 0; i != n; i++) {

## temp[i] = arr[i];

## }

## temp[n] = num;

## delete[] arr;

## arr = temp;

## n++;

## std::cin >> num;

## }

## cout << "Вы ввели: ";

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

## {

## insert(arr[i]);

## }

## if (n == 0)

## cout << "В массиве нет элементов";

## else

## cout << "\nРазмерность введенного массива " << n << "\n";

## display();

## cout << "\n";

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя на создание рандомного массива: " << time.count() << '\n';

## deleteAllNodes();

## std::cin.clear();

## std::cin.ignore(32767, '\n');

## break;

## }

## case 3:

## {

## start = std::chrono::system\_clock::now();

## int znachenie;

## ifstream file("massiv.txt");

## if (file.is\_open())

## {

## for (int i = 0; i < 9; ++i)

## {

## file >> znachenie;

## insert(znachenie);

## }

## }

## cout << "Массив из файла\n";

## display();

## cout << endl;

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя на чтение массива из файла: " << time.count() << '\n';

## deleteAllNodes();

## break;

## }

## case 4:

## break;

## }

## } while (k != 4);

## break;

## }

## case 5:

## {

## int l;

## do

## {

## cout << "\n1.Вставка элемента двусвязного списка + время\n";

## cout << "2.Удаление элемента двусвязного списка + время\n";

## cout << "3.Получение элемента двусвязного списка + время\n";

## cout << "4.Выход\n";

## cout << "\n";

## cout << "Вы выбрали ====> ";

## cin >> l;

## system("cls");

## switch (l)

## {

## case 1:

## {

## int n, vstavka;

## cout << "Введите кол-во элементов ===> ";

## cin >> n;

## cout << endl;

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

## {

## insert(0 + rand() % 99);

## }

## cout << "Массив до вставки\n";

## display();

## cout << endl << "\nВведите элемент, который вы хотите вставить===> ";

## cin >> vstavka;

## start = std::chrono::system\_clock::now();

## addElement(head, vstavka);

## cout << "\nМассив после вставки\n";

## display();

## cout << endl;

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя вставки: " << time.count() << '\n';

## deleteAllNodes();

## break;

## }

## case 2:

## {

## int j;

## do

## {

## cout << "\n1.Удаление по индексу\n";

## cout << "2.Удаление по значению\n";

## cout << "3.Выход\n";

## cout << "Вы выбрали ====> ";

## cin >> j;

## system("cls");

## switch (j)

## {

## case 1:

## {

## int n;

## cout << "Введите кол-во элементов ===> ";

## cin >> n;

## cout << endl;

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

## {

## insert(0 + rand() % 99);

## }

## cout << "\nМассив до удаления элемента: \n";

## display();

## int l;

## start = std::chrono::system\_clock::now();

## cout << "\nВведите индекс элемента для удаления===> ";

## cin >> l;

## if ((l <= n) && (l > 0))

## {

## deleteNode(l);

## cout << "\nМассив после удаления\n";

## display();

## deleteAllNodes();

## }

## else

## {

## cout << "Ошибка : Индекс не подходит условию";

## deleteAllNodes();

## break;

## }

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя удаления элемента по индексу " << time.count() << '\n';

## break;

## }

## case 2:

## {

## int n;

## cout << "Введите кол-во элементов ===> ";

## cin >> n;

## cout << endl;

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

## {

## insert(0 + rand() % 99);

## }

## cout << "\nМассив до удаления элемента: \n";

## display();

## start = std::chrono::system\_clock::now();

## int l;

## cout << "\nВведите значение элемента для удаления===> ";

## cin >> l;

## Delete(l);

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя удаления элемента по значению: " << time.count() << '\n';

## deleteAllNodes();

## break;

## }

## case 3:

## break;

## }

## } while (j != 3);

## break;

## }

## case 3:

## {

## int d;

## do

## {

## cout << "\n1.Получение элемента по индексу\n";

## cout << "2.Получение элемента по значению\n";

## cout << "3.Выход\n";

## cout << "Вы выбрали ====> ";

## cin >> d;

## system("cls");

## switch (d)

## {

## case 1:

## {

## int n, l;

## cout << "Введите кол-во элементов ===> ";

## cin >> n;

## cout << endl;

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

## {

## insert(0 + rand() % 99);

## }

## cout << "\n";

## display();

## start = std::chrono::system\_clock::now();

## cout << "\nВведите индекс ===> ";

## cin >> l;

## if ((l <= n) && (l > 0))

## {

## cout << "\nВаш элемент ===> " << displayEle(head, l - 1) << endl;

## deleteAllNodes();

## }

## else

## {

## cout << "Ошибка : Индекс не подходит условию";

## deleteAllNodes();

## break;

## }

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя получения элемента по индексу: " << time.count() << '\n';

## break;

## }

## case 2:

## {

## int n, x;

## cout << "Введите кол-во элементов ===> ";

## cin >> n;

## cout << endl;

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

## {

## insert(0 + rand() % 99);

## }

## cout << "\n";

## display();

## cout << "\nВведите элемент для получения===> ";

## start = std::chrono::system\_clock::now();

## cin >> x;

## int c = search(head, x);

## if (c == -1)

## {

## cout << "\nЭтого элемента нет в массиве " << endl;

## deleteAllNodes();

## break;

## }

## else

## {

## cout << "\nИндекс этого элемента ===> " << c;

## deleteAllNodes();

## }

## end = std::chrono::system\_clock::now();

## time = end - start;

## std::cout << "\nВремя получения элемента по элементу: " << time.count() << '\n';

## break;

## }

## case 3:

## break;

## }

## } while (d != 3);

## }

## case 4:

## break;

## }

## } while (l != 4);

## }

## case 6:

## break;

## }

## } while (task != 6);

## break;

## }

## case 3:

## {

## while (answer != 11)

## {

## cout << "1. Преобразовать выражение в обратную польскую нотацию" << endl;

## cout << "2. Преобразовать выражение в польскую нотацию" << endl;

## cout << "3. Проверка выражения, записанного в инфиксной нотации" << endl;

## cout << "4. Проверка выражения, записанного в ОПН" << endl;

## cout << "5. Проверка выражения, записанного в ПН" << endl;

## cout << "6. Вычислить значение выражения, записанного в инфиксной нотации" << endl;

## cout << "7. Вычислить значение выражения, записанного в ОПН" << endl;

## cout << "8. Вычислить значение выражения, записанного в ПН" << endl;

## cout << "9. Проверочная работа ПН" << endl;

## cout << "10. Проверочная работа ОПН" << endl;

## cout << "11. ";

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Выход" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> answer;

## system("CLS");

## switch (answer)

## {

## case (1): {

## int key = 0;

## while (key != 3) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Выражение" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "1. С клавы" << endl;

## cout << "2. Из файла" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "3. Выход" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> key;

## system("CLS");

## switch (key)

## {

## case (1): {

## cin.ignore();

## string infix;

## getline(cin, infix);

## cout << infixToPostfix(infix) << endl;;

## break;

## }

## case (2): {

## string infix;

## ifstream file;

## file.open("sample.txt");

## if (file) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл открыт!" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## else {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл не открывается";

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## while (!file.eof())

## {

## file >> infix;

## }

## cout << infix << endl;

## cout << infixToPostfix(infix) << endl;;

## file.close();

## break;

## }

## }

## }

## break;

## }

## case (2): {

## int key = 0;

## while (key != 3) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Выражение" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "1. С клавы" << endl;

## cout << "2. Из файла" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "3. Выход" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> key;

## system("CLS");

## switch (key)

## {

## case (1): {

## cin.ignore();

## string infix;

## getline(cin, infix);

## cout << infixToPrefix(infix) << endl;

## break;

## }

## case (2): {

## string infix;

## ifstream file;

## file.open("sample.txt");

## if (file) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл открыт!" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## else {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл не открывается";

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## while (!file.eof())

## {

## file >> infix;

## }

## cout << infix << endl;

## cout << infixToPrefix(infix) << endl;

## file.close();

## break;

## }

## }

## }

## break;

## }

## case (3): {

## int key = 0;

## while (key != 3) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Выражение" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "1. С клавы" << endl;

## cout << "2. Из файла" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "3. Выход" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> key;

## system("CLS");

## switch (key)

## {

## case (1): {

## cin.ignore();

## string infix;

## getline(cin, infix);

## if ((isPrefix(infix) == 1) && (isPostfix(infix) == 1)) {

## cout << "Выражение корректно" << endl;

## }

## else {

## cout << "Ошибка" << endl;

## }

## break;

## }

## case (2): {

## string infix;

## ifstream file;

## file.open("infix.txt");

## if (file) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл открыт!" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## else {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл не открывается";

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## while (!file.eof())

## {

## file >> infix;

## }

## cout << infix << endl;

## if ((isPrefix(infix) == 1) && (isPostfix(infix) == 1)) {

## cout << "Выражение корректно" << endl;

## }

## else {

## cout << "Ошибка" << endl;

## }

## file.close();

## break;

## }

## }

## }

## break;

## }

## case (4): {

## int key = 0;

## while (key != 3) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Выражение" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "1. С клавы" << endl;

## cout << "2. Из файла" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "3. Выход" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> key;

## system("CLS");

## switch (key)

## {

## case (1): {

## cin.ignore();

## string infix;

## getline(cin, infix);

## if (isPostfix(infix) == 1) {

## cout << "Ошибка" << endl;

## }

## else {

## cout << "Выражение корректно" << endl;

## }

## break;

## }

## case (2): {

## string infix;

## ifstream file;

## file.open("postfix.txt");

## if (file) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл открыт!" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## else {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл не открывается";

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## while (!file.eof())

## {

## file >> infix;

## }

## cout << infix << endl;

## if (isPostfix(infix) == 1) {

## cout << "Ошибка" << endl;

## }

## else {

## cout << "ВЫражение корректно" << endl;

## }

## file.close();

## break;

## }

## }

## }

## break;

## }

## case (5): {

## int key = 0;

## while (key != 3) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Выражение" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "1. С клавы" << endl;

## cout << "2. Из файла" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "3. Выход" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> key;

## system("CLS");

## switch (key)

## {

## case (1): {

## cin.ignore();

## string infix;

## getline(cin, infix);

## if (isPrefix(infix) == 1) {

## cout << "Ошибка" << endl;

## }

## else {

## cout << "Выражение корректно" << endl;

## }

## break;

## }

## case (2): {

## string infix;

## ifstream file;

## file.open("prefix.txt");

## if (file) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл открыт!" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## else {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "файл не открывается";

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## while (!file.eof())

## {

## file >> infix;

## }

## cout << infix << endl;

## if (isPrefix(infix) == 1) {

## cout << "Ошибка" << endl;

## }

## else {

## cout << "Выражение корректно" << endl;

## }

## file.close();

## break;

## }

## }

## }

## break;

## }

## case (6): {

## int key = 0;

## while (key != 3) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Выражение" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "1. С клавы" << endl;

## cout << "2. Из файла" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "3. Выход" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> key;

## system("CLS");

## switch (key)

## {

## case (1): {

## cin.ignore();

## string infix;

## getline(cin, infix);

## infix = infixToPostfix(infix);

## cout << evaluatePostfix(infix) << endl;

## break;

## }

## case (2): {

## string infix;

## ifstream file;

## file.open("infixE.txt");

## if (file) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл открыт!" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## else {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл не открывается";

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## while (!file.eof())

## {

## file >> infix;

## }

## cout << infix << endl;

## infix = infixToPostfix(infix);

## cout << evaluatePostfix(infix) << endl;

## file.close();

## break;

## }

## }

## }

## break;

## }

## case (7): {

## int key = 0;

## while (key != 3) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Выражение" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "1. С клавы" << endl;

## cout << "2. Из файла" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "3. Выход" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> key;

## system("CLS");

## switch (key)

## {

## case (1): {

## cin.ignore();

## string infix;

## getline(cin, infix);

## cout << evaluatePostfix(infix) << endl;

## break;

## }

## case (2): {

## string infix;

## ifstream file;

## file.open("postfixE.txt");

## if (file) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл открыт!" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## else {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл не открывается";

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## while (!file.eof())

## {

## file >> infix;

## }

## cout << infix << endl;

## cout << evaluatePostfix(infix) << endl;

## file.close();

## break;

## }

## }

## }

## break;

## }

## case (8): {

## int key = 0;

## while (key != 3) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "Выражение" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "1. С клавы" << endl;

## cout << "2. Из файла" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cout << "3. Выход" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> key;

## system("CLS");

## switch (key)

## {

## case (1): {

## cin.ignore();

## string infix;

## getline(cin, infix);

## cout << evaluatePrefix(infix) << endl;

## break;

## }

## case (2): {

## string infix;

## ifstream file;

## file.open("prefixE.txt");

## if (file) {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл открыт!" << endl;

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## else {

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Файл не открывается";

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## }

## while (!file.eof())

## {

## file >> infix;

## }

## cout << infix << endl;

## cout << evaluatePrefix(infix) << endl;

## file.close();

## break;

## }

## }

## }

## break;

## }

## case (9): {

## fstream file;

## fstream answers;

## int n;

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Введите кол-во примеров, которое хотите увидеть (1-8): ";

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> n;

## file.open("verifPostfix.txt", ios::in);

## if (file.is\_open()) {

## string tp;

## answers.open("answersPostfix.txt", ios::app);

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

## (getline(file, tp));

## cout << tp << endl;

## cout << evaluatePostfix(tp) << endl;

## cout << endl;

## answers << evaluatePostfixFile(tp) << endl;

## }

## answers.close();

## }

## file.close();

## break;

## }

## case (10): {

## fstream file;

## fstream answers;

## int n;

## SetConsoleTextAttribute(N, FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY);

## cout << "Введите кол-во примеров, которое хотите увидеть (1-8): ";

## SetConsoleTextAttribute(N, FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY);

## cin >> n;

## file.open("verifPrefix.txt", ios::in);

## if (file.is\_open()) {

## string tp;

## answers.open("answersPrefix.txt", ios::app);

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

## (getline(file, tp));

## cout << tp << endl;

## cout << evaluatePrefix(tp) << endl;

## cout << endl;

## answers << evaluatePrefixFile(tp) << endl;

## }

## answers.close();

## }

## file.close();

## break;

## }

## return 0;

## }

## break;

## }

## }

## }

## }

## }