Пермский национальный исследовательский политехнический университет

**Лабораторная работа №1**

**Дисциплина: Информатика**

**Вариант № 9**

**"Классы и объекты. Инкапсуляция."**

Выполнил: Зайченко Никита Геннадьевич

Проверила: Доцент кафедры ИТАС Полякова О. А.

Пермь 2022

**Содержание отчета**

1) Постановка задачи (общая и конкретного варианта).

2) Описание класса.

3) Определение компонентных функций.

4) Определение функции make().

5) Объяснение результатов работы программы.

6) Ответы на контрольные вопросы

**Постановка задачи (общая и конкретного варианта)**

1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.

2) Использование классов и объектов в ОО программе.

3) Поле first – дробное число х, координата точки, поле second – дробное число y, координата точки. Реализовать метод distance(double x1, doudle y1) – вычисление расстояния от точки с координатами (first, second) до точки с координатами (x1,y1).

**Описание класса.**

struct fraction

{

double first;

double second;

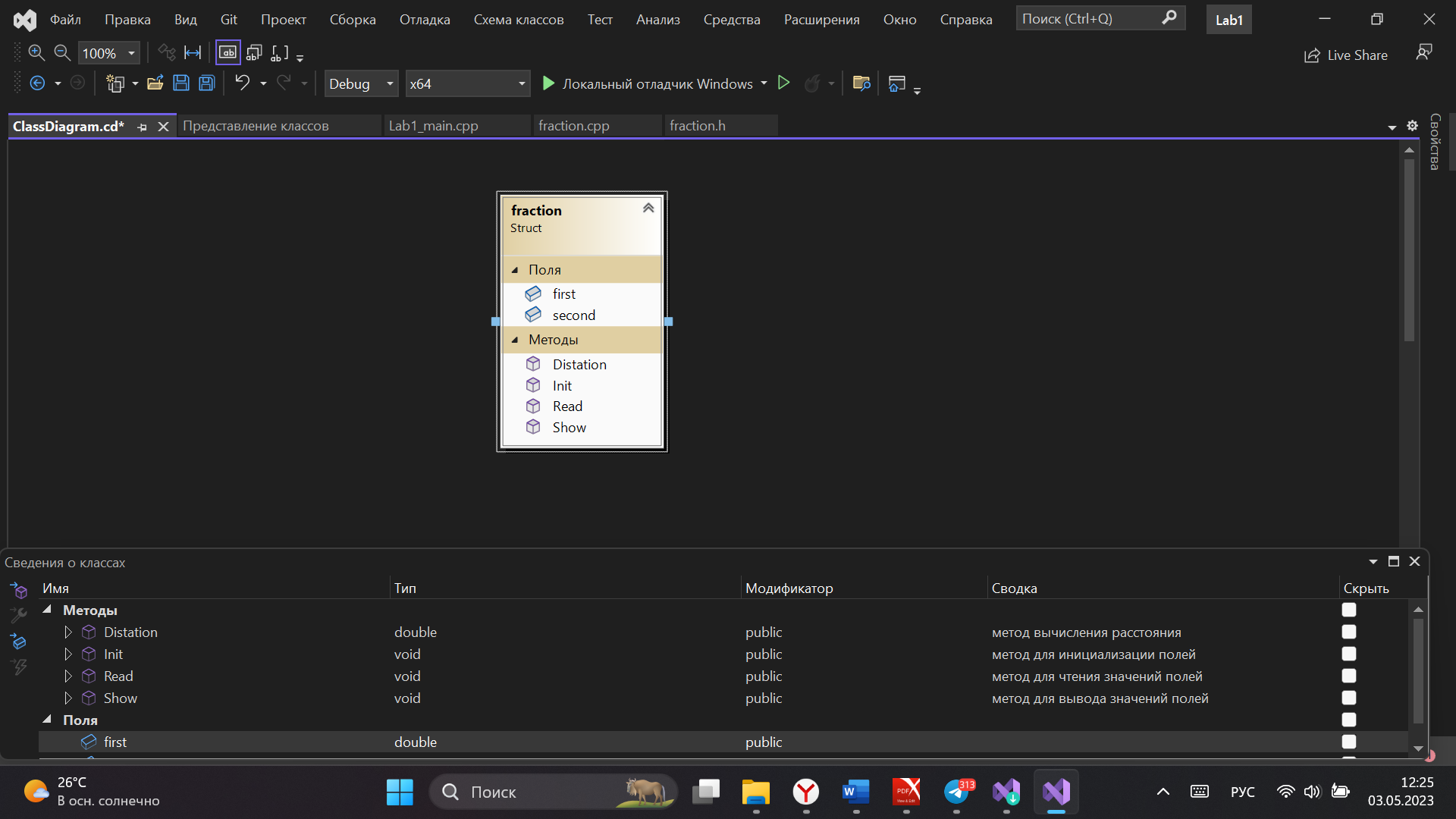
void Init(double, double);//метод для инициализации полей

void Read();//метод для чтения значений полей

void Show();//метод для вывода значений полей

double Distation(fraction, fraction);//метод вычисления расстояния

};



**Определение компонентных функций.**

void fraction::Init(double F, double S)

{

first = F; second = S;

}

//реализация метода для чтения значений полей структуры

void fraction::Read()

{

cout << endl << "first? "; cin >> first;

cout << "second? "; cin >> second;

}

//реализация метода для вывода значений полей структуры

void fraction::Show()

{

cout << endl << "first = " << first;

cout << endl << "second = " << second;

cout << endl;

}

//метод вычисления расстояния

double fraction::Distation(fraction A, fraction B)

{

return (pow((B.first - A.first), 2)) + (pow((B.second - A.second),2));

}

**Определение функции make().**

double make\_double(double F, double S)

{

if (F < 0 || S < 0)

{

cout << "Error" << endl;

exit(1);

}

fraction t;//создали временные переменные

fraction s;

t.Init(F, S);//инициализировали поля переменной t и s с помощью параметров функции

s.Init(3.0, 2.1);

t.Show();

s.Show();

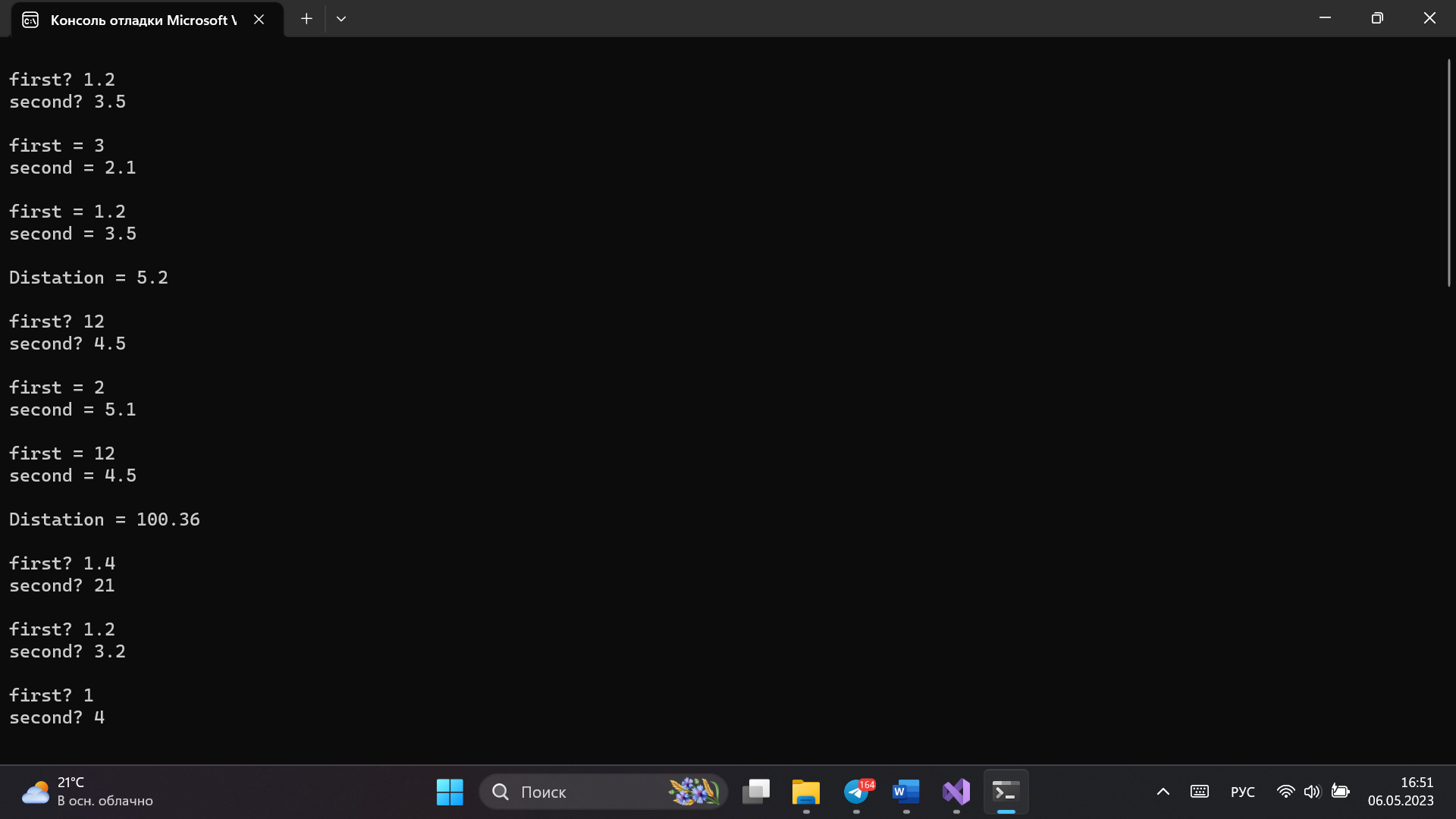
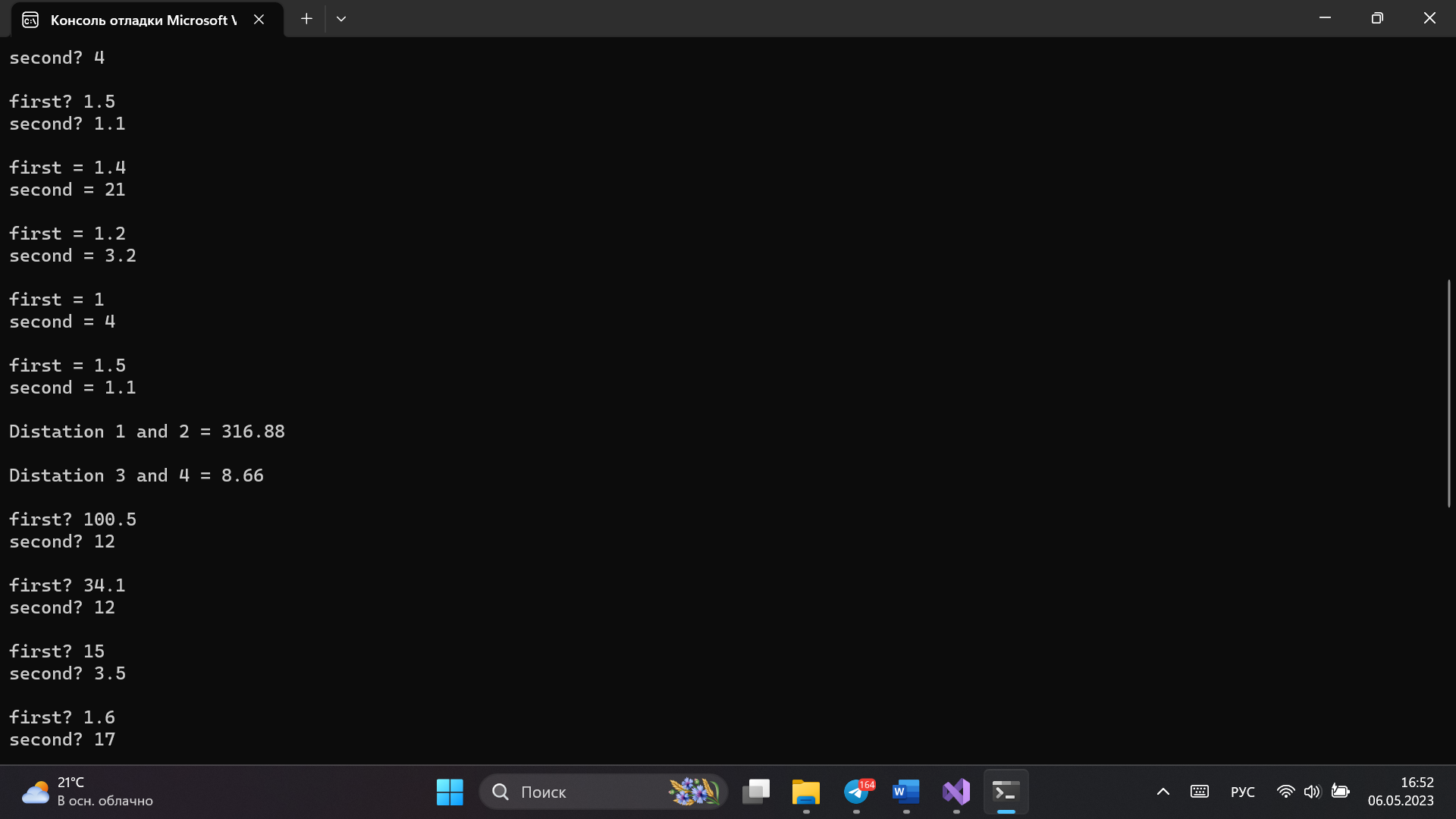
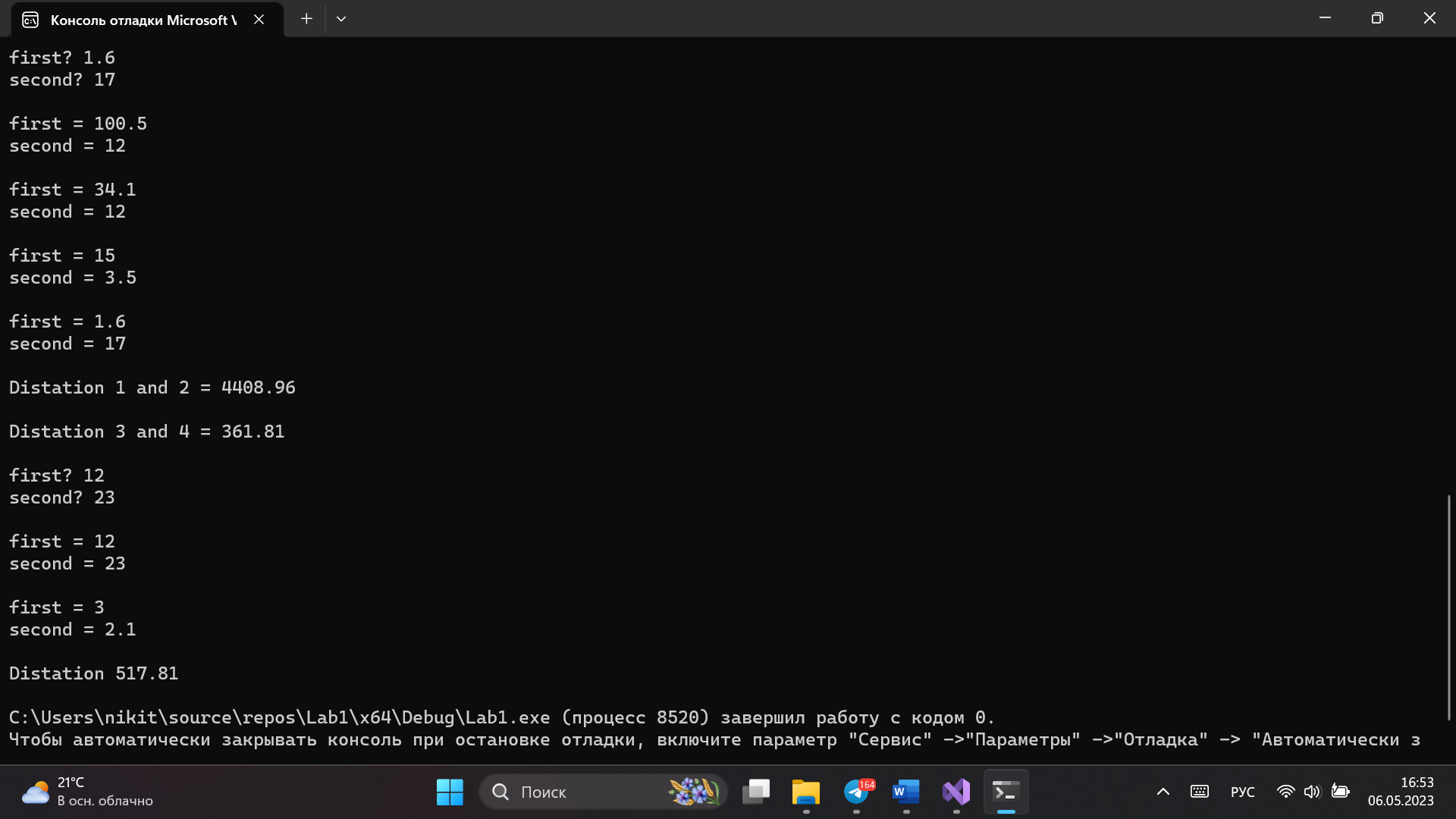
double a = t.Distation(t, s);

return a;//вернули значение переменной t

}

**Объяснение результатов работы программы.**

Результатом работы программы является вычисление дистанции между введёнными пользователем точками координат, с помощью метода double fraction::Distation(fraction A, fraction B).

**Ответы на контрольные вопросы**

**1. Что такое класс?**

Класс является абстрактным типом данных, определяемым пользователем, и

представляет собой модель реального объекта в виде данных и функций для работы с ними.

class <имя>

{

[ private: ]

<описание скрытых элементов>

public:

<описание доступных элементов>

}; // Описание заканчивается точкой с запятой

**2. Что такое объект (экземпляр) класса?**

Объектом класса (экземпляром) называют переменную класса, а переменных такого класса (объектов) может быть сколько угодно.

Person p;

Person \*pp;//указатель на переменную

Person Arr\_p[10];//массив переменных

**3. Как называются поля класса?**

Поля класса - это переменные, объявленные внутри определения класса, которые хранят данные, связанные с объектом этого класса. Также называемые переменными-членами класса или атрибутами.

class Person {

public:

string name; // поле класса, хранящее имя

string surname; // поле класса, хранящее фамилию

int age; // поле класса, хранящее возраст

};

**4. Как называются функции класса?**

Функции класса называются методами.

**5. Для чего используются спецификаторы доступа?**

Спецификаторы доступа в объектно-ориентированном программировании используются для ограничения доступа к полям и методам класса извне класса. Они определяют, какие поля и методы будут видны и доступны для других частей программы. Существуют три спецификатора доступа: public, private и protected.

**6. Для чего используется спецификатор public?**

public делает поля и методы доступными для всех частей программы, включая другие классы.

**7. Для чего используется спецификатор private?**

private делает поля и методы доступными только внутри класса, в котором они объявлены.

**8. Если описание класса начинается со спецификатора class, то какой спецификатор доступа будет использоваться по умолчанию?** Private.

**9. Если описание класса начинается со спецификатора struct, то какой спецификатор доступа будет использоваться по умолчанию?** Public.

**10. Какой спецификатор доступа должен использоваться при описании интерфейса класса? Почему?**

При описании интерфейса класса, все методы должны быть объявлены как public, потому что интерфейс представляет набор методов, которые доступны для использования всеми частями программы, включая те, которые не являются частью исходного класса.

**11. Каким образом можно изменить значения атрибутов экземпляра класса?**

Значения атрибутов экземпляра класса можно изменить, вызывая соответствующие методы класса, которые изменяют значения атрибутов, либо обращаясь к этим атрибутам напрямую из других функций программы, если эти атрибуты являются открытыми (public).

**12. Каким образом можно получить значения атрибутов экземпляра класса?**

Получить значения атрибутов экземпляра класса можно вызывая соответствующие методы класса или обращаясь к ним напрямую в других функциях программы, если эти атрибуты являются открытыми.

**13. Класс описан следующим образом**

**struct Student**

**{**

**string name;**

**int group; ………**

**};**

**Объект класса определен следующим образом**

**Student \*s=new Student;**

**Как можно обратиться к полю name объекта s?**

s->name

**14. Класс описан следующим образом**

**struct Student**

**{**

**string name;**

**int group; …….. };**

**Объект класса определен следующим образом**

**Student s;**

**Как можно обратиться к полю name объекта s?**

s.name

**15. Класс описан следующим образом**

**class Student**

**{**

**string name;**

**int group; ……. };**

**Объект класса определен следующим образом**

**Student \*s=new Student;**

**Как можно обратиться к полю name объекта s?**

s->name

**16. Класс описан следующим образом**

**class Student**

**{**

**string name;**

**int group;**

**public:**

**….. };**

**Объект класса определен следующим образом**

**Student s;**

**Как можно обратиться к полю name объекта s?**

s.name

**17. Класс описан следующим образом**

**class Student**

**{**

**public:**

**char\* name;**

**int group; ………**

**};**

**Объект класса определен следующим образом**

**Student \*s=new Student;**

**Как можно обратиться к полю name объекта s?**s->name