**Ф****ЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информационная безопасность»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине

«Объектно-ориентированное программирование систем защиты информации»

на тему

«Компиляция файлов исходного кода и компоновка полученных объектных файлов в исполняемый модуль. Линейные алгоритмы»

Вариант 27

Выполнила:

студентка группы БПЗ1801

Яковлева К.А.

Проверил:

старший преподаватель кафедры ИБ

Барков В.В

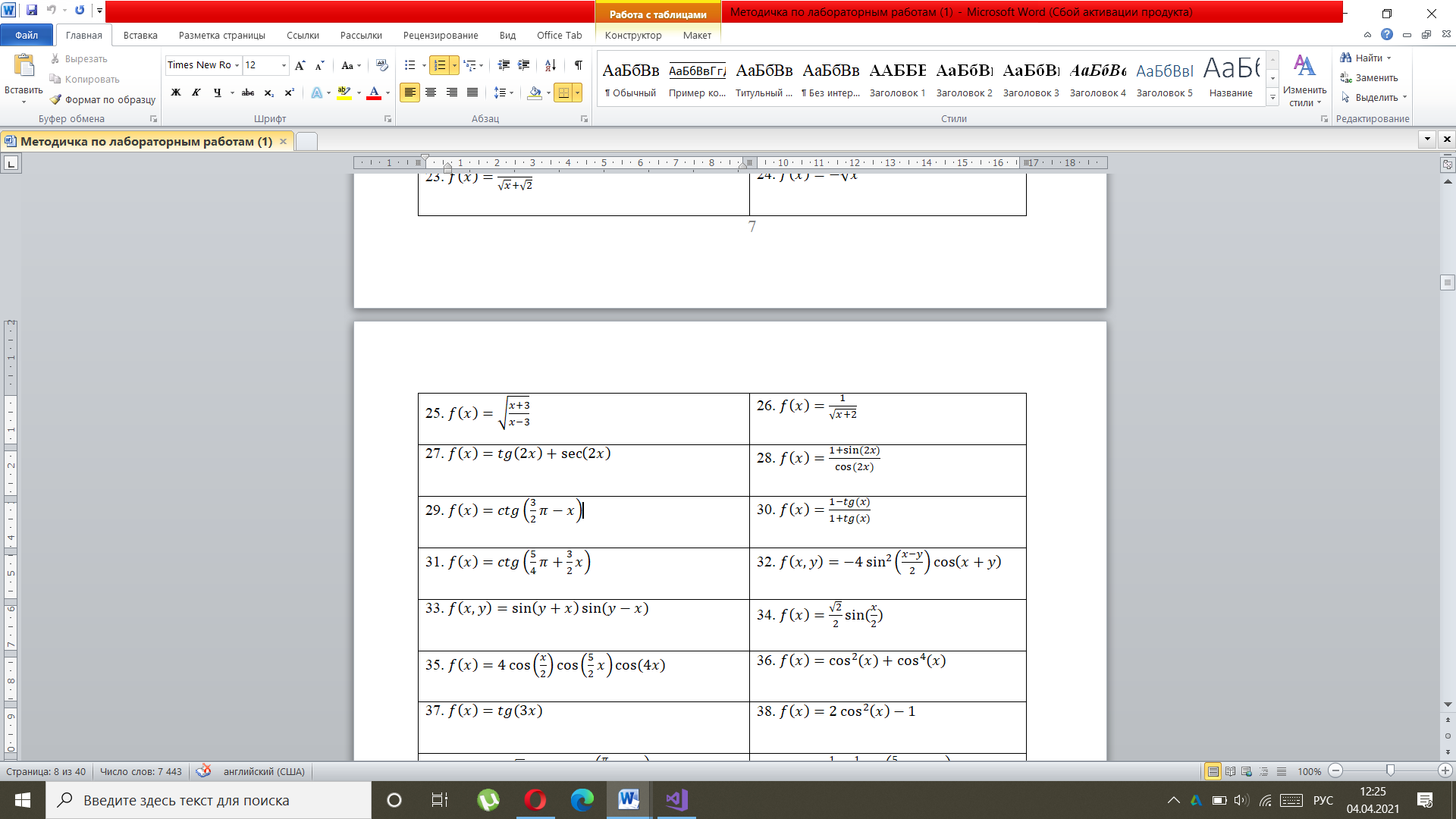
Москва 2021

**Цель работы**

Целью работы является получение навыка создания линейных программ на языке C с применением арифметических операций.

**Задание**

Индивидуальное:



Общее:

#### Задание 1

Составить программу, имеющую линейный алгоритм и состоящую из одной функции void main(). Программу записать в файл с именем task1.c. Скомпилировать, скомпоновать и выполнить.

В функции main организовать вычисление задачи вашего варианта дважды:

* для исходных данных, значения которых задать в виде констант в тексте функции main;
* для исходных произвольных данных, значения которых пользователь Вашей программы должен ввести с клавиатуры в процессе выполнения программы.

Проанализировать результаты работы и сделать выводы. Перенести полученные результаты в отчет.

Вывод действительных чисел осуществлять с точностью до 0.0001

#### Задание 2

Линейный алгоритм функции из задания 1 разделить на две процедуры, выделив в одну вычислительные операции этого алгоритма, а в другую все операции ввода-вывода. Каждую процедуру оформить как функцию. Вычислительную часть алгоритма оформить как функцию с параметрами, передаваемыми по значению, и возвращаемым значением. Прототип функции:

double f(double x) // Если функция имеет один параметр

double f(double x, double y) // Если функция имеет два параметра

Другую часть алгоритма оформить как функцию void main(), вызывающую первую функцию нужное количество раз. Записать тексты функций файл с именем task2.с в следующем порядке: функция с параметрами, функция main. Cкомпилировать, скомпоновать и выполнить.

#### Задание 3

Создать файл task3.с, в котором изменить порядок записи текстов функций, созданных в задании 2. Функции записать в следующем порядке: функция main, функция с параметрами, организующая вычисления (double f(double x) или double f(double x, double y)). Внести требуемые дополнения, добиться успешной компиляции, скомпоновать и выполнить.

#### Задание 4

По-прежнему линейный алгоритм функции из задания 1 разделить на две процедуры, выделив в одну вычислительные операции этого алгоритма, а в другую все операции ввода-вывода.

Вычислительную часть алгоритма оформить как функцию без параметров и без возвращаемого значения. Прототип функции void f(void)

Другую часть алгоритма оформить как функцию void main(), вызывающую вычислительную функцию. Обмен данными между функциями организовать через глобальные объекты (double x, y – аргументы, double result – результат вычисления). Записать тексты функций в файл с именем task4.с в следующем порядке: функция main, затем функция без возвращаемого значения и без параметров. Cкомпилировать, скомпоновать и выполнить.

#### Задание 5

В этом задании необходимо разделить текст файла task3.с на два файла. В первый файл с именем task5\_main.c поместить текст функции main. Скомпилировать только файл task5\_main.c.

Во второй файл с именем task5\_func.с поместить текст функции с параметрами (double f(double x) или double f(double x, double y)). Скомпилировать только файл task5\_func.c. После раздельной компиляции осуществить совместную компоновку. Полученный исполняемый файл выполнить.

Проанализировать результаты работы и сделать выводы.

#### Задание 6

В этом задании необходимо разделить текст файла task4.с на два файла.

В первый файл с именем task6\_main.c поместить текст функции main. Скомпилировать только файл task6\_main.c.

Во второй файл с именем task6\_func.с поместить определения глобальных объектов (double x, y – аргументы, double result – результат вычисления) и текст функции без параметров (void f(void)). Скомпилировать только файл task6\_func.c.

После раздельной компиляции осуществить совместную компоновку. Разобраться в проблемах, возникающих при совместной компиляции и компоновки. Полученный исполняемый файл выполнить.

Проанализировать результаты работы и сделать выводы.

#### Задание 7

В этом задании необходимо модифицировать тексты файлов из задания 6.

Описание функции и внешних переменных выделить в отдельный заголовочный файл func.h, включить его содержимое в файлы task7\_main.c и task7\_func.c. Определение внешних переменных произвести в файле task7\_func.c.

Скомпилировать по отдельности файлы task7\_main.c и task7\_func.c. Произвести компоновку, выполнить полученный исполняемый файл.

#### Задание 8

Скомпилировать по отдельности файлы из задания 7 task7\_main.c и task7\_func.c. Создать статическую библиотеку task8\_lib.lib, включающую объектный файл task7\_func.obj. Полученную статическую библиотеку скомпоновать с файлом task7\_main.obj. Выполнить полученный исполняемый файл.

**Выполнение**

Задание 1

Для того, чтобы задать точность 0.0001 я использую функцию std::setprecision(), которая находится в заголовочном файле iomanip:

Листинг 1 – код к заданию номер 1

#include "pch.h"

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

double x = 8, f;

f = tan(2 \* x) + 1 / cos(2 \* x);

std::cout << std::setprecision(4);

cout << "x = " << x << "; f = " << f << endl;

cin >> x;

f = tan(2 \* x) + 1 / cos(2 \* x);

cout << "x = " << x << "; f = " << f << endl;

}

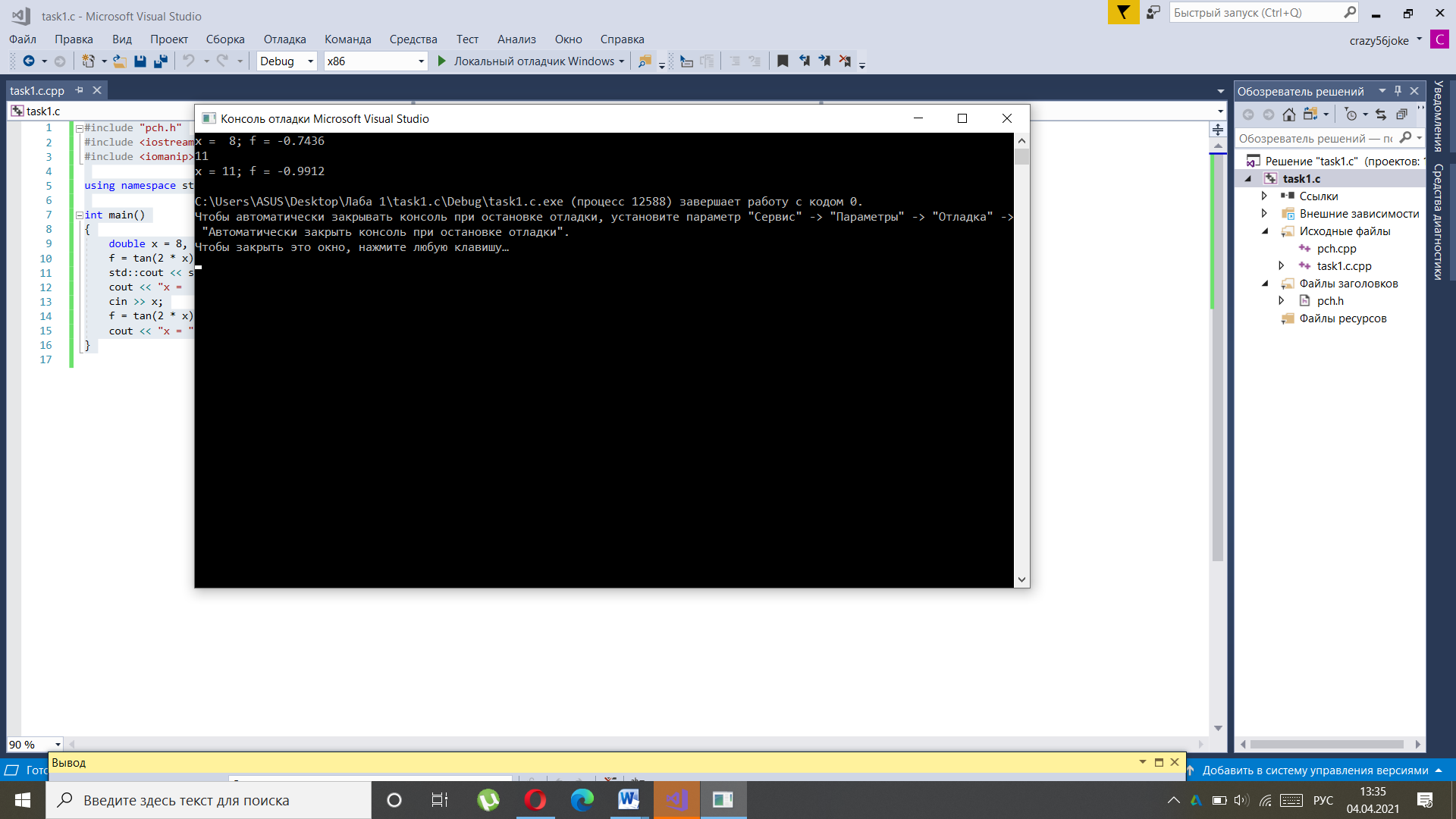


Рисунок 1 – Результат выполнения программы task1.cpp

Задание 2

Листинг 2 – программный код к заданию номер 2

#include "pch.h"

#include <iostream>

using namespace std;

double f(double x)

{

return tan(2 \* x) + 1 / cos(2 \* x);

}

void main()

{

double x = 8, fun;

fun = f(x);

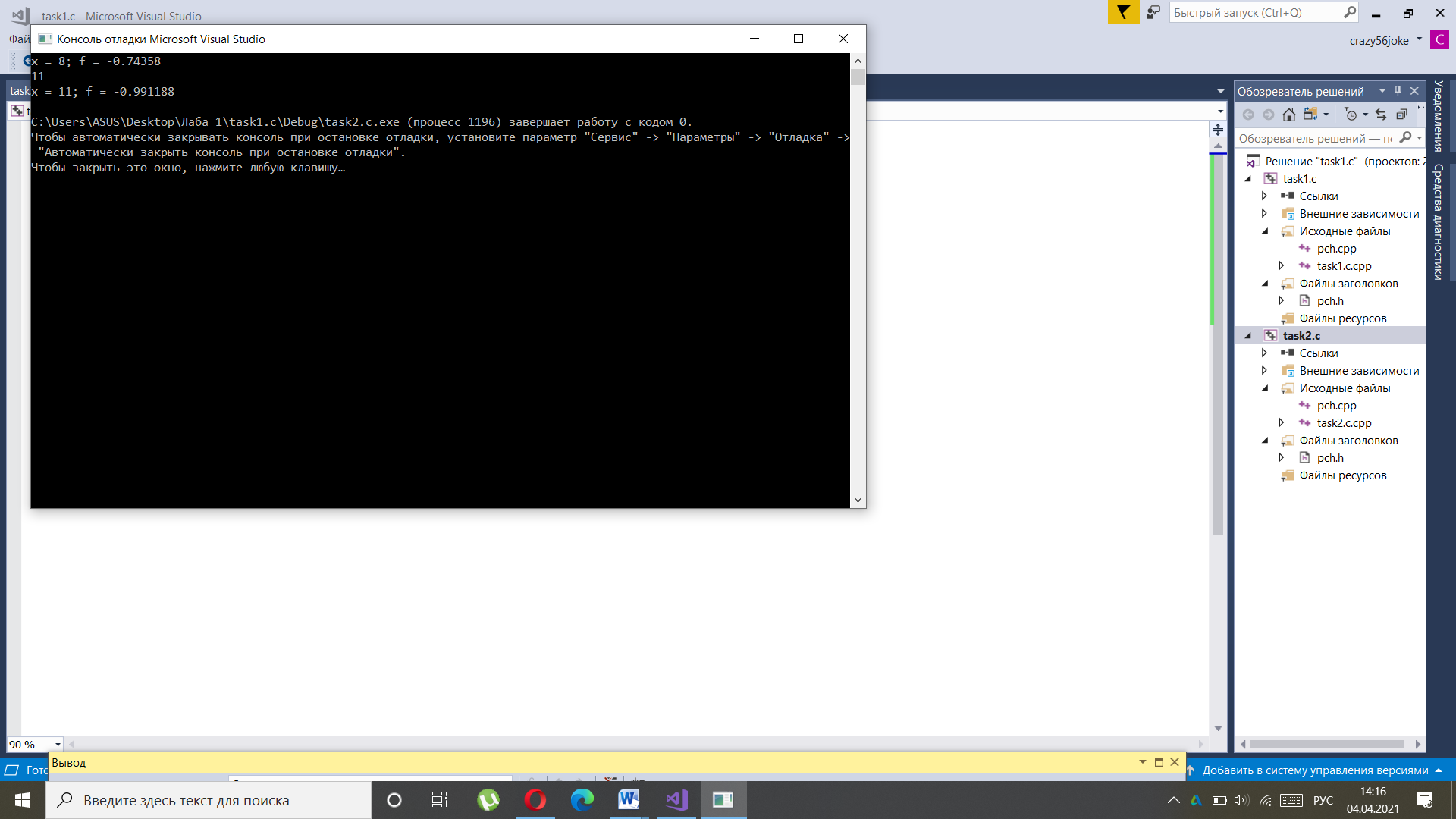
cout << "x = " << x << "; f = " << fun << endl;

cin >> x;

fun = f(x);

cout << "x = " << x << "; f = " << fun << endl;

}

Рисунок 2 –результат работы программы task2.cpp

Задание 3.

Листинг 3 – программный код к работе номер 3

#include "pch.h"

#include <iostream>

using namespace std;

double f(double x);

void main()

{

double x = 8, fun;

fun = f(x);

cout << "x = " << x << "; f = " << fun << endl;

cin >> x;

fun = f(x);

cout << "x = " << x << "; f = " << fun << endl;

}

double f(double x)

{

return tan(2 \* x) + 1 / cos(2 \* x);

}

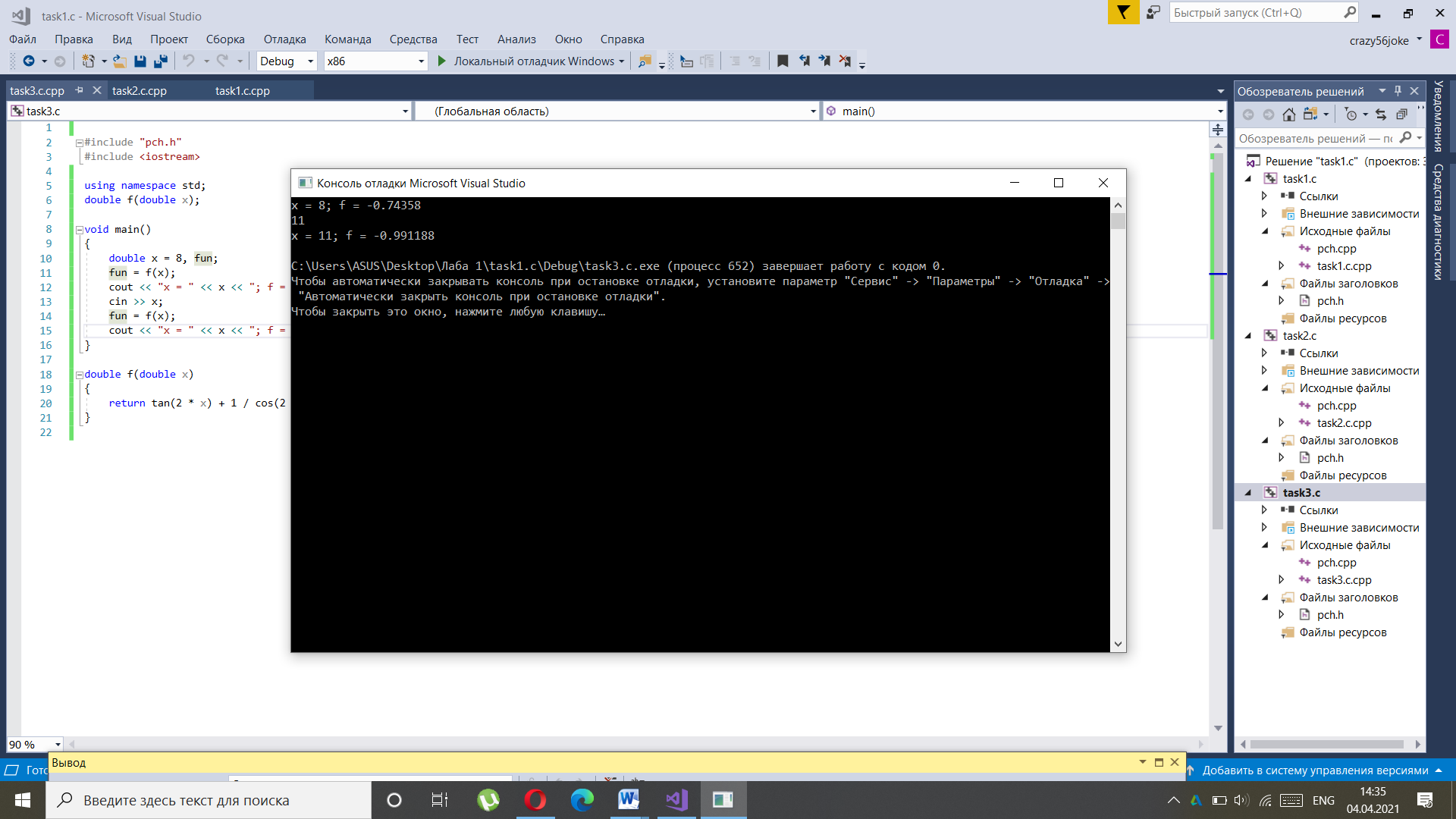


Рисунок 3 – результат работы программного кода task3.cpp

Задание 4.

Листинг 4 – программный код к заданию 4

#include "pch.h"

#include <iostream>

double x = 8, result;

using namespace std;

void f(void)

{

result = tan(2 \* x) + 1 / cos(2 \* x);

}

void main()

{

f();

cout << "x = " << x << "; f = " << result << endl;

cin >> x;

f();

cout << "x = " << x << "; f = " << result << endl;

}

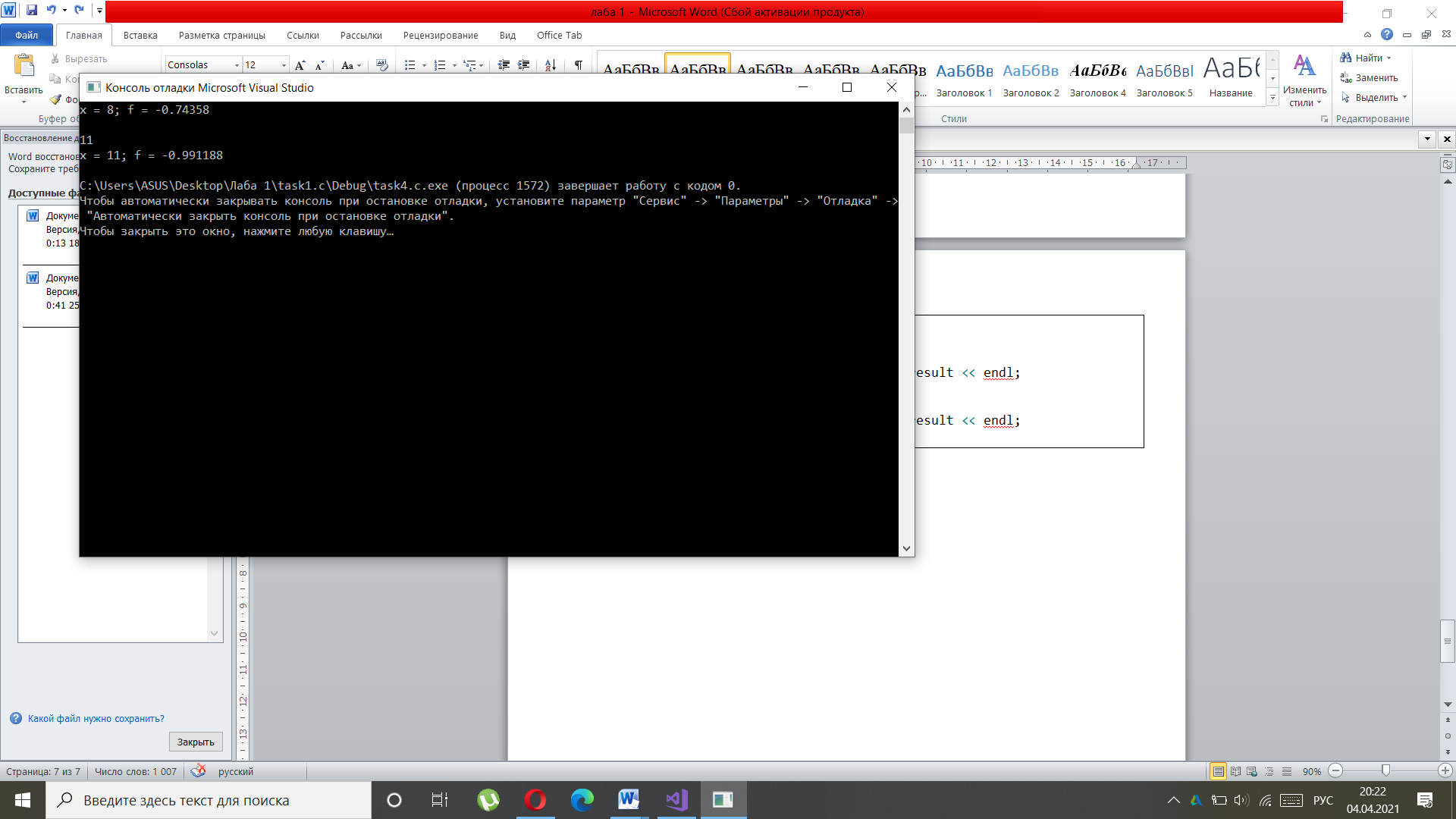


Рисунок 4 – результат работы программы task4.cpp

Задание 5.

Листинг 5 – программный код task5\_main.cpp

#include "pch.h"

#include <iostream>

using namespace std;

double f(double x);

void main()

{

double x = 8, fun;

fun = f(x);

cout << "x = " << x << "; f = " << fun << endl;

cin >> x;

fun = f(x);

cout << "x = " << x << "; f = " << fun << endl;

}

Листинг 6 – программный код task5\_func.cpp

#include "pch.h"

#include <iostream>

double f(double x)

{

return tan(2 \* x) + 1 / cos(2 \* x);

}

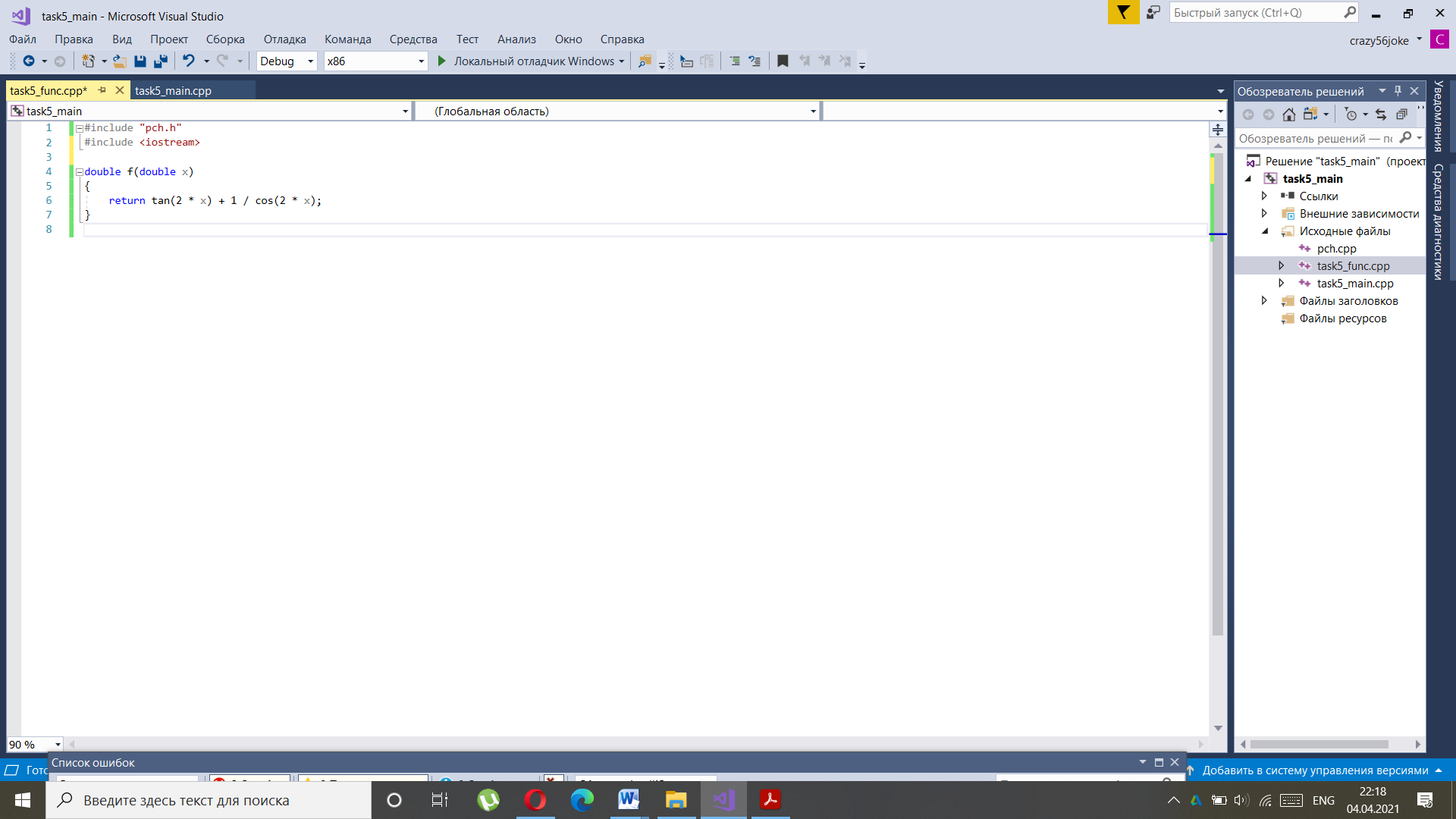


Рисунок 5 – директорий к заданию 5

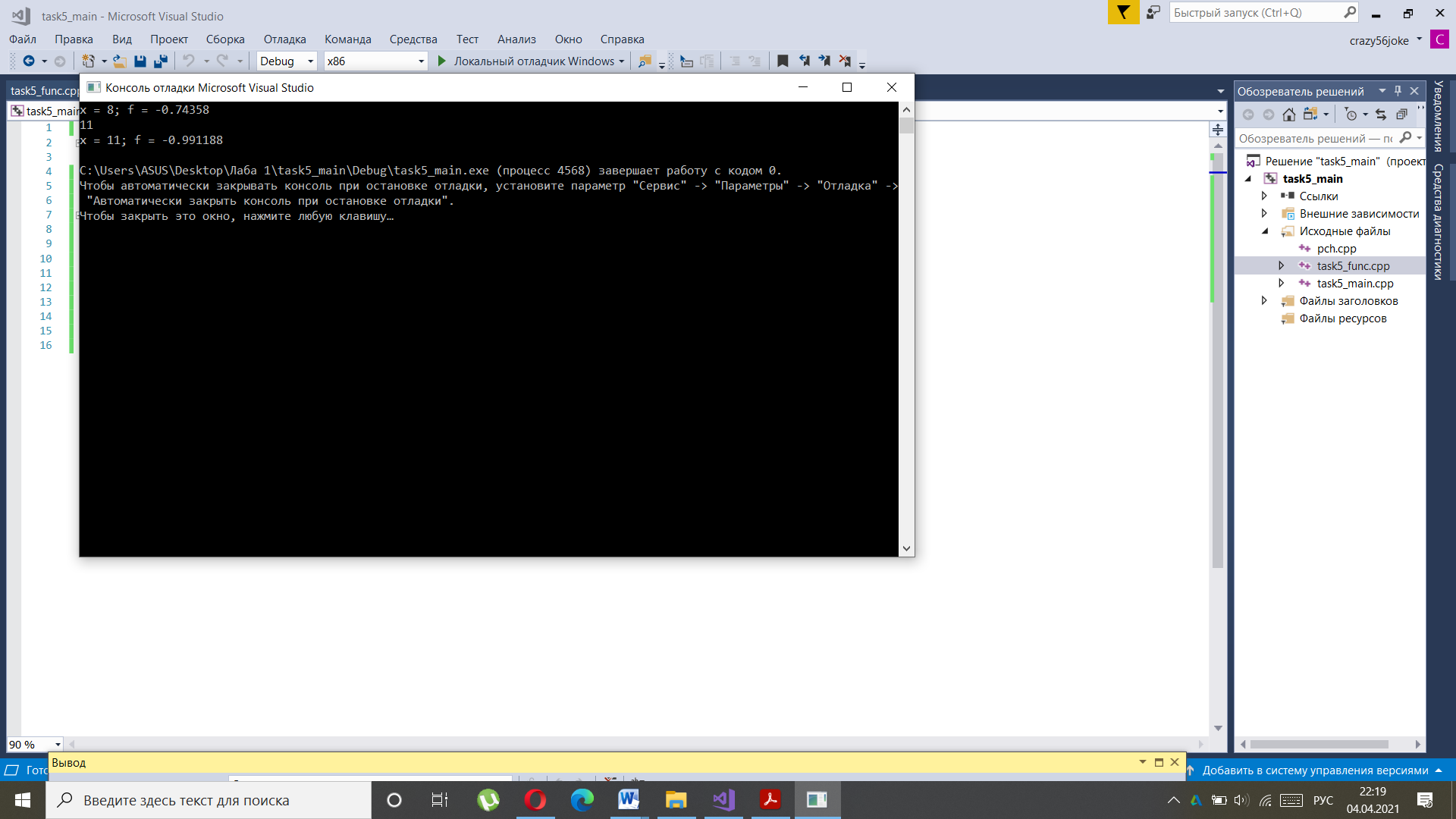


Рисунок 6 – результат работы программы task5\_main.cpp

Задание 6.

Листинг 7 – программный код task6\_main.cpp

#include "pch.h"

#include <iostream>

using namespace std;

extern double x, result;

void f();

void main()

{

f();

cout << "x = " << x << "; f = " << result << endl;

cin >> x;

f();

cout << "x = " << x << "; f = " << result << endl;

}

Листинг 8 – программный код task6\_func.cpp

#include "pch.h"

#include <iostream>

double x=8, result;

void f(void)

{

result = tan(2 \* x) + 1 / cos(2 \* x);

}

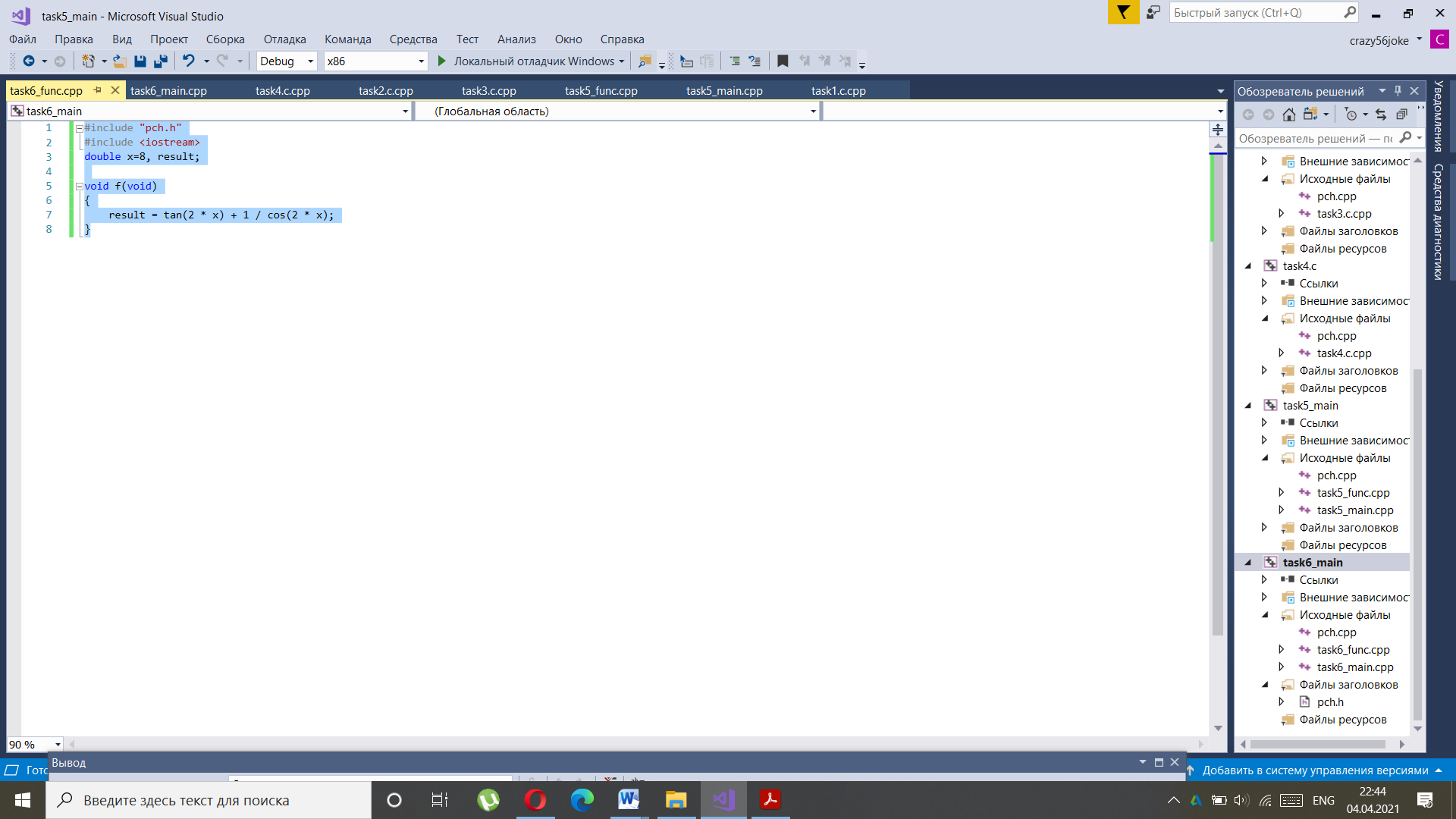


Рисунок 7 – директорий к заданию 6

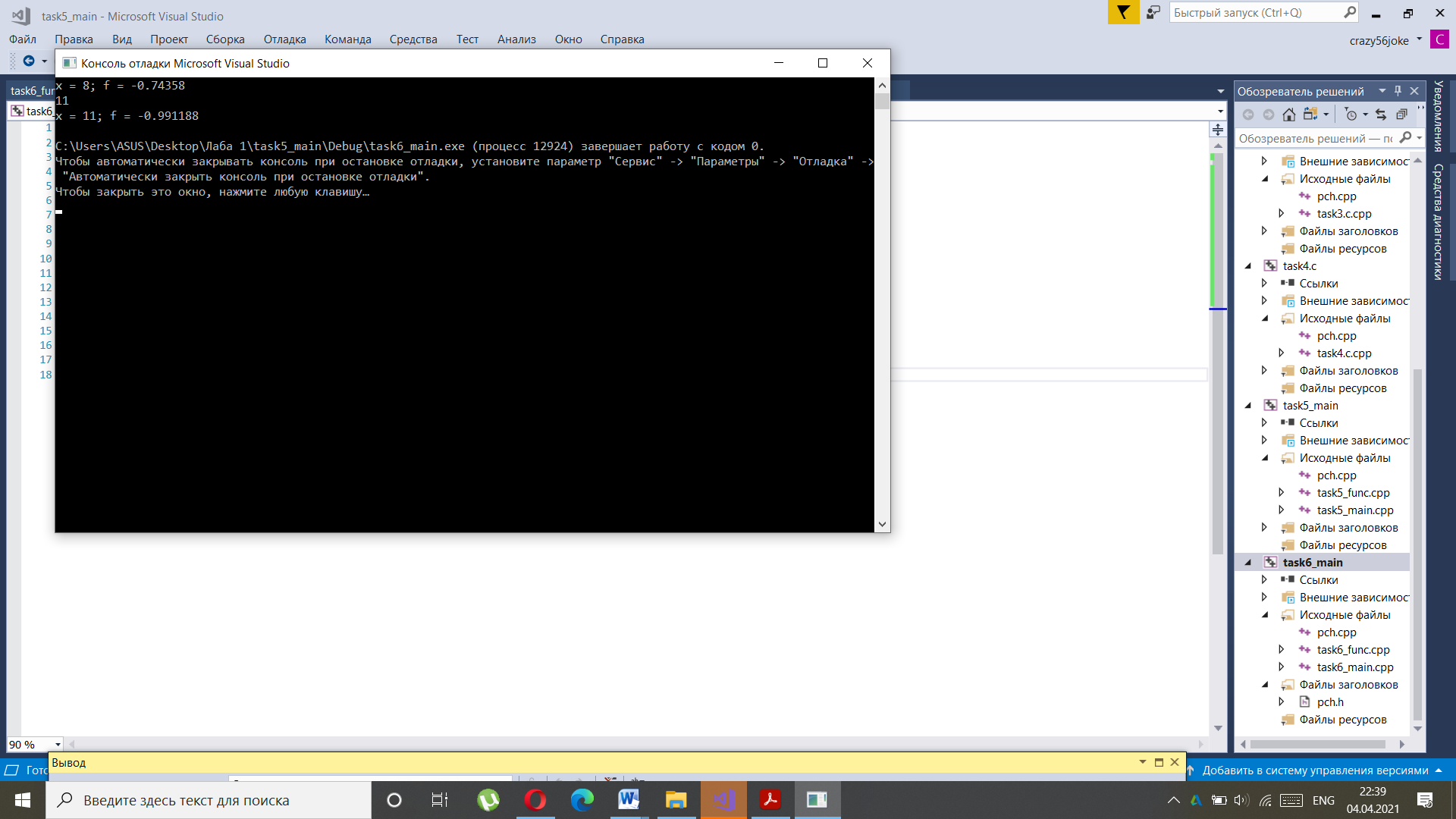


Рисунок 8 – результат программы task6\_main.cpp

Листинг 9 – программный код task7\_main.cpp

#include "pch.h"

#include <iostream>

#include "func.h"

using namespace std;

void main()

{

f();

cout << "x = " << x << "; f = " << result << endl;

cin >> x;

f();

cout << "x = " << x << "; f = " << result << endl;

}

Листинг 10 –программный код task7\_func.cpp

#include "pch.h"

#include "func.h"

#include "math.h"

double x = 8, result;

void f(void)

{

result = tan(2 \* x) + 1 / cos(2 \* x);

}

Листинг 11 – программный код к func.h

#include "pch.h"

extern double x, result;

void f();

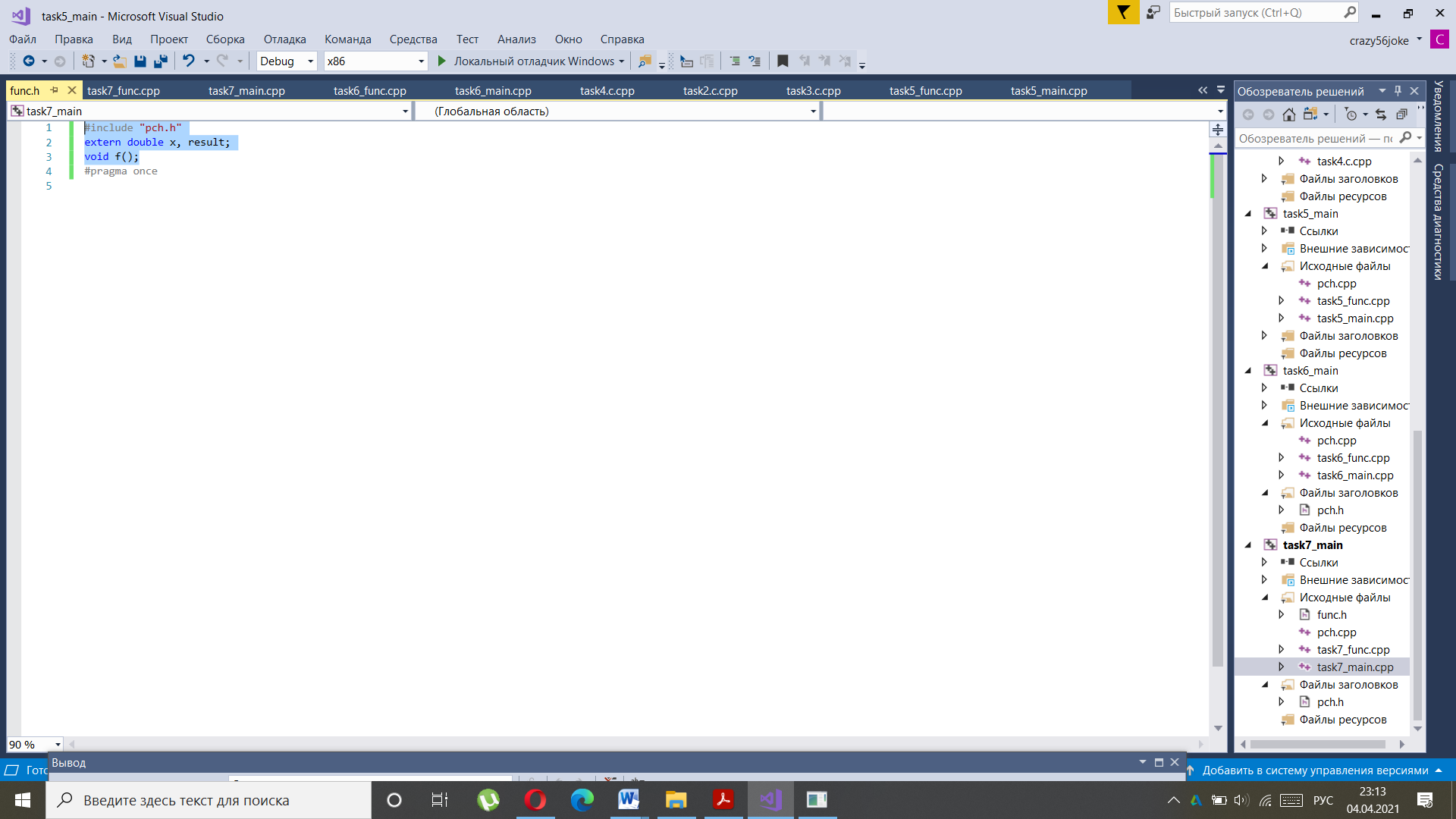


Рисунок 9 – директорий к заданию 7

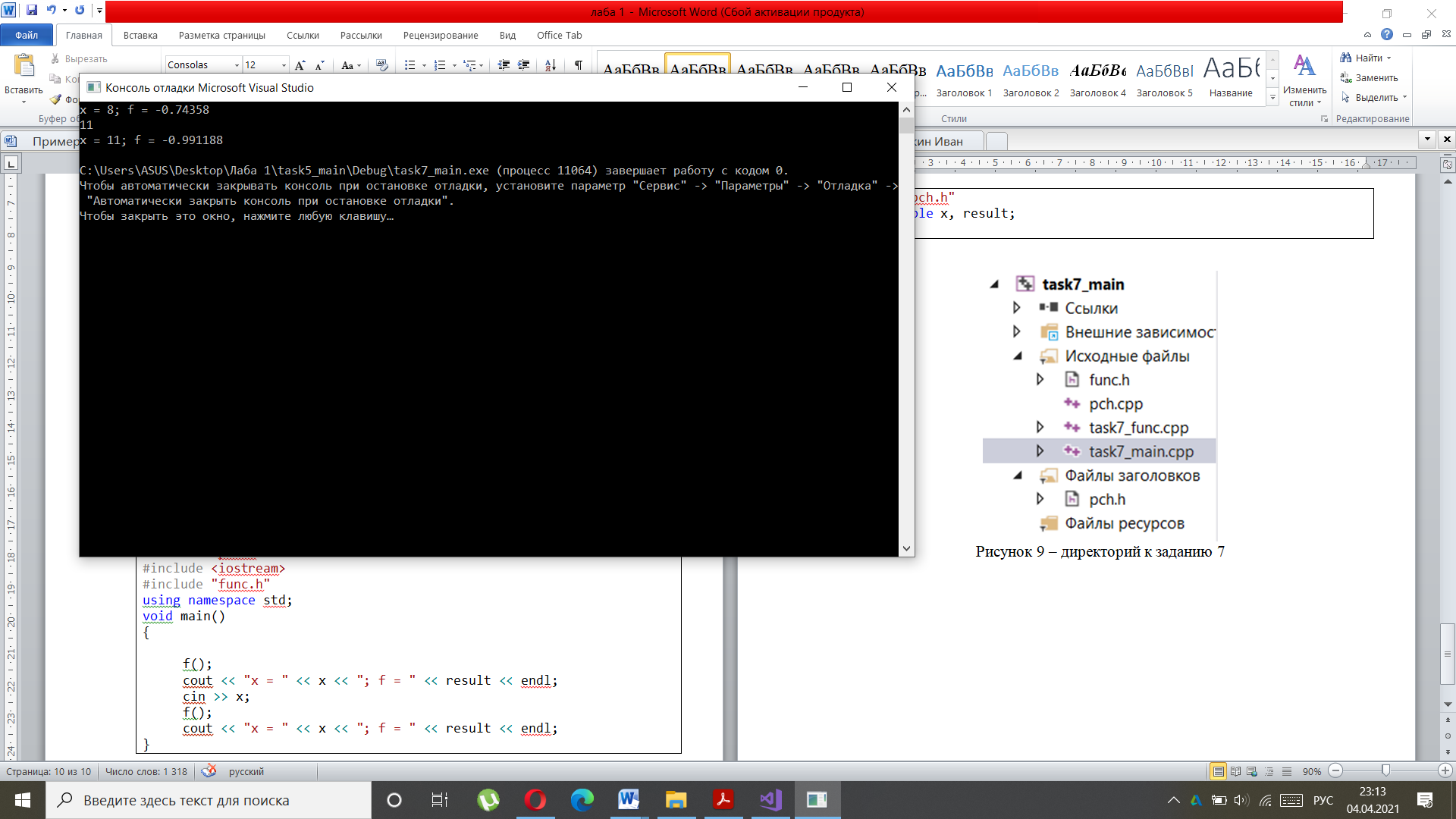


Рисунок 10 – результат работы программного кода task7\_main.cpp

Задание 8.

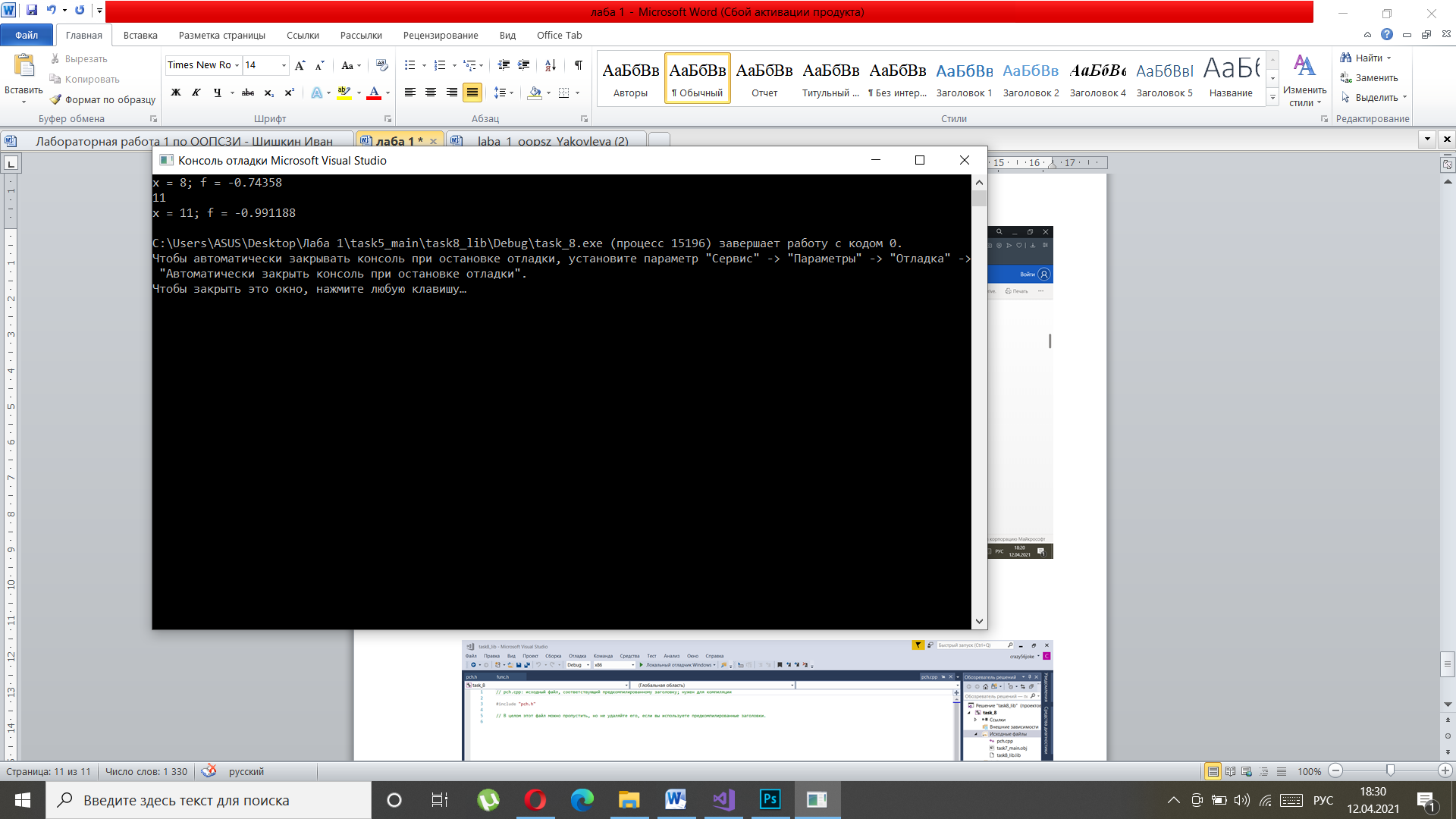


Рисунок 11 – результат работы скомпоновки task7\_main.obj и статистической библиотеки task8\_lib.lib

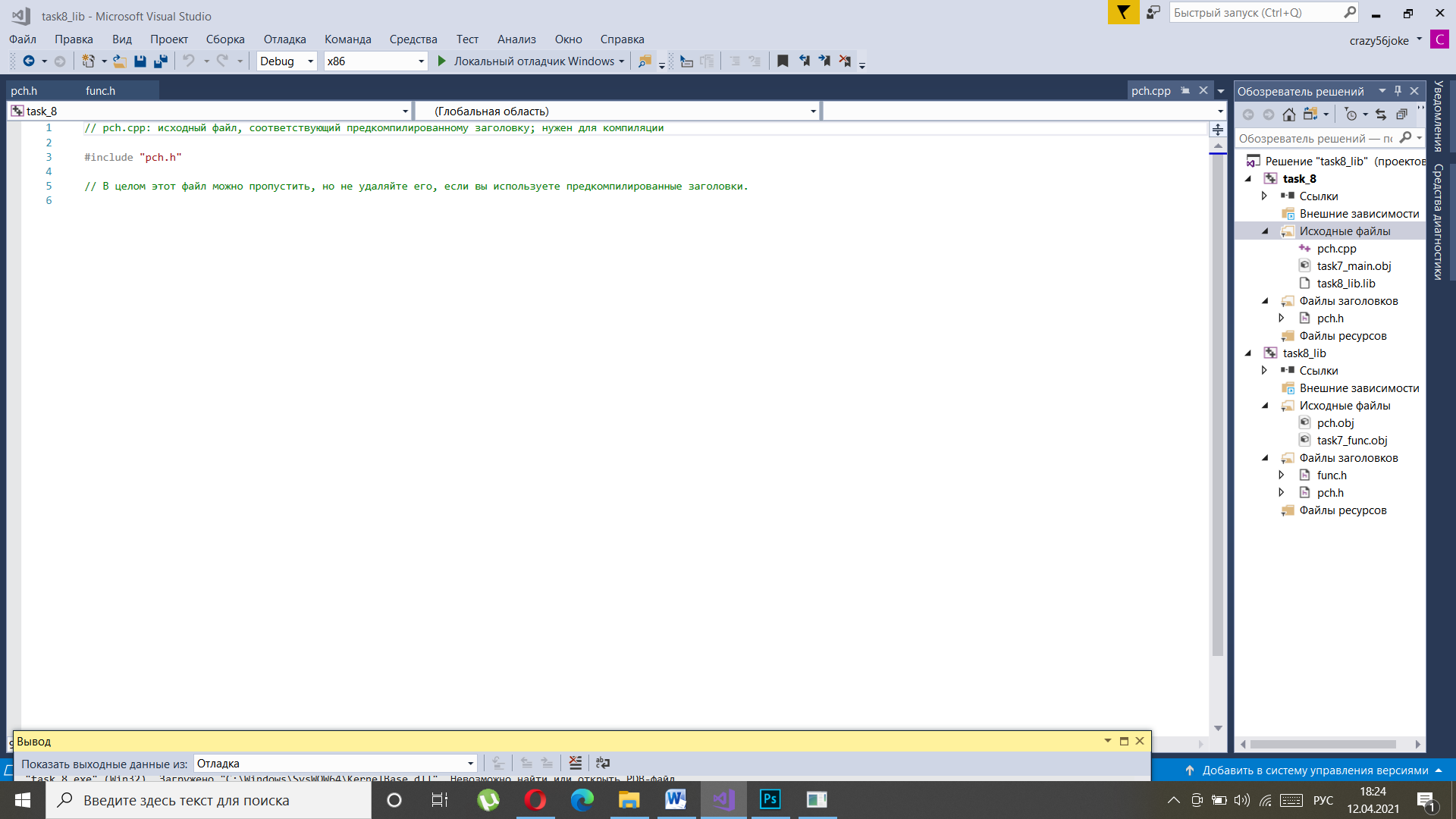


Рисунок 12 – директорий к заданию 8

**Вывод**

В данной лабораторной работе мною были изучены: метод создания линейных программ с использованием дополнительных библиотек и функций, различные типы функций, особенности глобальных и локальных переменных и области их применения, метод создания статической библиотеки.