

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №1**

**з дисципліни «Проектний практикум»**

**на тему:**

**«Дослідження предметної області. Виявлення функціональних вимог»**

Виконав:

студент гр. ПЗ2011

Кулик С. В.

Прийняв:

Іванов О.П.

Дніпро, 2023

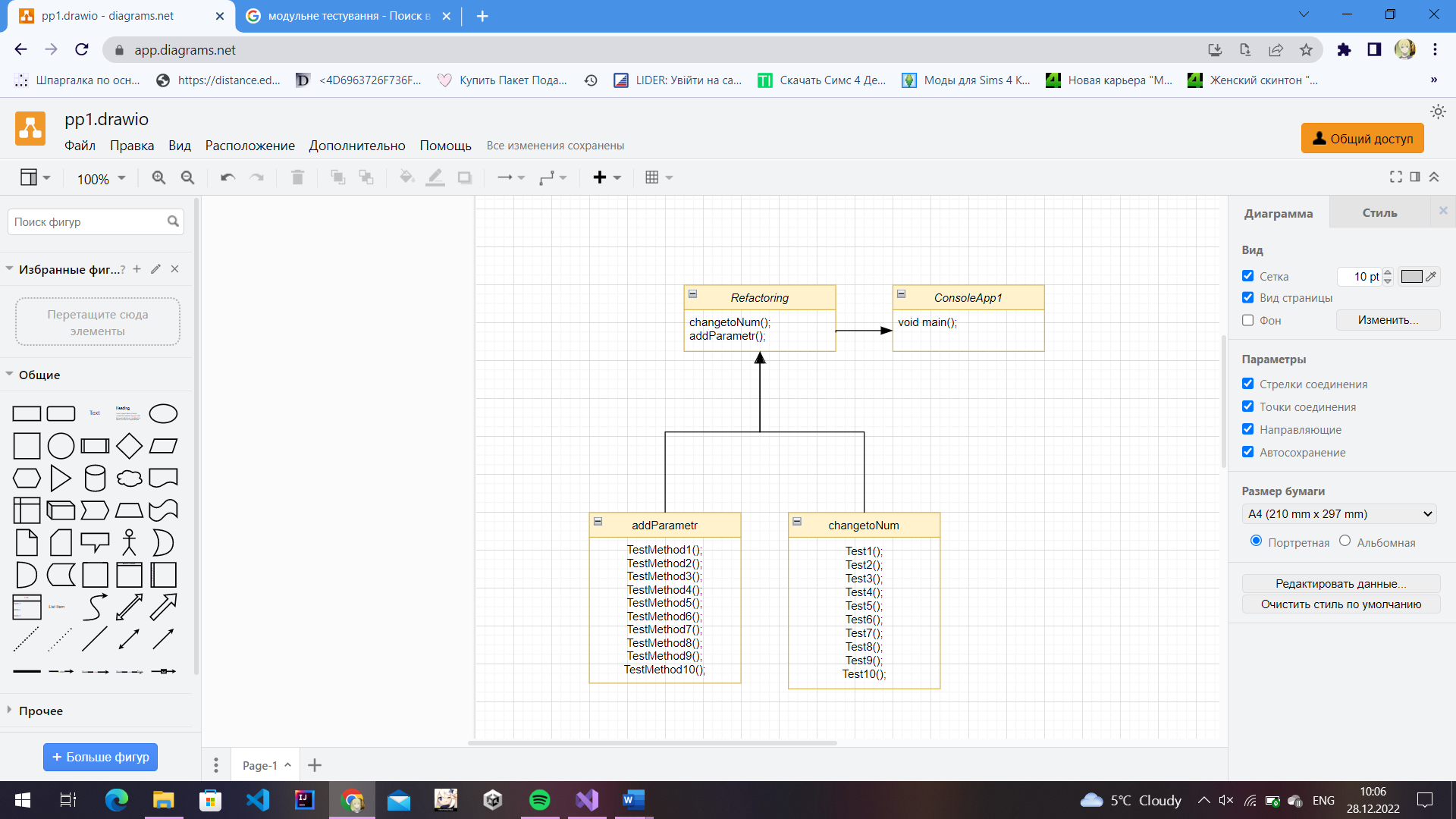
Тема: Дослідження предметної області. Виявлення функціональних вимог

Завдання:

Створення текстового редактору для виконання рефакторингу вихідних текстів програм для мови С++.

* Заміна магічного числа символічної константи
* Додавання параметра

1. Концептуальна модель системи (узагальнена діаграма класів)



1. Код програми

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using ConsoleApp1;

namespace UnitTestProject1

{

[TestClass]

public class changetoNum

{

[TestMethod]

public void Test1()

{

string name = "double c = mass\*height\*9.81;";

Refact test = new Refact();

Assert.AreEqual(

"const double GRAVITATIONAL\_CONSTANT = 9.81;\n string name = double c = mass\*height\*GRAVITATIONAL\_CONSTANT;",

test.changetoNum(name, "9.81", "GRAVITATIONAL\_CONSTANT"));

}

[TestMethod]

public void Test2()

{

string name = "double c = 3.14 \* pow(R,2)";

Refact test = new Refact();

Assert.AreEqual(

"const double pi = 3.14;\n string name = double c = pi \* pow(R,2);",

test.changetoNum(name, "3.14", "pi"));

}

[TestMethod]

public void Test3()

{

string name = "double c = 2.9979 / n";

Refact test = new Refact();

Assert.AreEqual(

"const double speed\_vacuum = 2.9979; \n string name = double c = speed\_vacuum ",

test.changetoNum(name, "2.9979", "speed\_vacuum"));

}

[TestMethod]

public void Test4()

{

string name = "double c = 30 \* 24 \* 3600 \* 1000; ";

Refact test = new Refact();

Assert.AreEqual

("const double milliseconds\_in\_second = 1000;\n const double hours\_in\_day = 24; \n const double secons\_in\_hour = 3600; \n const double days = 30;\n string name = double c = days\*milliseconds\_in\_second\*hours\_in\_day\*secons\_in\_hour ",

test.changetoNum(name, "30", "days"),

test.changetoNum(name, "24", "hours\_in\_day"),

test.changetoNum(name, "3600", "secons\_in\_hour"),

test.changetoNum(name, "1000", "milliseconds\_in\_second"));

}

[TestMethod]

public void Test5()

{

string name = " double c = c + (c \* 0.07);if(c < 50.00){c += 20;}";

Refact test = new Refact();

Assert.AreEqual(

"const double tax =0.7;\n const double min\_price = 50.00; \n const double ship = 20;\n string name = double c = c + (c \* tax);if(c < min\_price){c += ship;} ",

test.changetoNum(name, "0.7", "tax"),

test.changetoNum(name, "50.00", "min\_price"),

test.changetoNum(name, "20", "ship"));

}

[TestMethod]

public void Test6()

{

string name = "const double c = 20000 \* workedhours // c - salary";

Refact test = new Refact();

Assert.AreEqual(

"const double hour\_salary = 20000; \n string name = double c = hour\_salary \* workedhours ",

test.changetoNum(name, "20000", "hour\_salary"));

}

[TestMethod]

public void Test7()

{

string name = " double c = (int i=0; i<9; i++);";

Refact test = new Refact();

Assert.AreEqual(

"const double first\_cell = 0; \n const double last\_cell = 8; \n string name = double c = (int i = first\_cell; i<last\_cell; i++) ",

test.changetoNum(name, "0", "first\_cell"), test.changetoNum(name, "8", "last\_cell"));

}

[TestMethod]

public void Test8()

{

string name = "double c = 3,14 \* 6;";

Refact test = new Refact();

Assert.AreEqual(

"const double pi = 3,14; \n const double d = 6; \n string name = double c = pi \* d ;",

test.changetoNum(name, "3.14", "pi"),

test.changetoNum(name, "6", "d"));

}

[TestMethod]

public void TestMethod9()

{

string text = "double c = mass \* 9,78 // c - force of gravity;";

Refact r = new Refact();

Assert.AreEqual(

"const double g = 9.78; \n string name = double c = mass \* g ;",

r.changetoNum(text, "9.78", "g"));

}

[TestMethod]

public void TestMethod10()

{

string text = "double c = (int k=1; k<=9; k++);";

Refact r = new Refact();

Assert.AreEqual(

"const double first\_elem = 1; \n const double last\_elem = 9; \n string name = double c = (int i = first\_elem; i<last\_elem; i++) ",

r.changetoNum(text, "1", "first\_elem"),

r.changetoNum(text, "9", "last\_elem"));

}

}

[TestClass]

public class addParametr

{

[TestMethod]

public void TestMethod1()

{

string text = "void func(int CustomerID)";

Refact r = new Refact();

Assert.AreEqual(

"int date = 5; \n void function(int CustomerID, int date)",

r.changetoNum(text, "func", "function"));

}

[TestMethod]

public void TestMethod2()

{

string text = "void func(8)";

Refact r = new Refact();

Assert.AreEqual(

"int num = 8; \n void function ( int num ) ",

r.changetoNum(text, "func", "function"));

}

[TestMethod]

public void TestMethod3()

{

string text = "void func()";

Refact r = new Refact();

Assert.AreEqual(

"string Line = hello world; \n void function(string Line )",

r.changetoNum(text, "func", "function"));

}

[TestMethod]

public void TestMethod4()

{

string text = "void func(), void test();// func - test function";

Refact r = new Refact();

Assert.AreEqual(

"int num = 8; \n void function(int num), void testing(int num) ;",

r.changetoNum(text, "test", "function"));

}

[TestMethod]

public void TestMethod5()

{

string text = "void func();// func - test function";

Refact r = new Refact();

Assert.AreEqual(

"int tmp; \n void function(int tmp);// Func - test function",

r.changetoNum(text, "func", "function", false, true));

}

[TestMethod]

public void TestMethod6()

{

string text = "void func(){int num = 9};";

Refact r = new Refact();

Assert.AreEqual(

"void function(int num){return num};",

r.changetoNum(text, "func", "function"));

}

[TestMethod]

public void TestMethod7()

{

string text = "void func(){string message = \"line\";}";

Refact r = new Refact();

Assert.AreEqual(

"void function(string message){return message;}",

r.changetoNum(text, "func", "function"));

}

[TestMethod]

public void TestMethod8()

{

string text = "void func(){string line = \"call function\"; \nConsole.WriteLine(line);}";

Refact r = new Refact();

Assert.AreEqual(

"void function(string line)",

r.changetoNum(text, "func", "function"));

}

[TestMethod]

public void TestMethod9()

{

string text = "void func(){string line = \"call function\"; \nConsole.WriteLine(line);}";

Refact r = new Refact();

Assert.AreEqual(

"void function(string line)",

r.changetoNum(text, "func", "function", true));

}

[TestMethod]

public void TestMethod10()

{

string text = "// func is test function; \n void func(){\n string message = \"call function\"; \nConsole.WriteLine(message);}";

Refact r = new Refact();

Assert.AreEqual(

"// function is test function; \nvoid function(string message)",

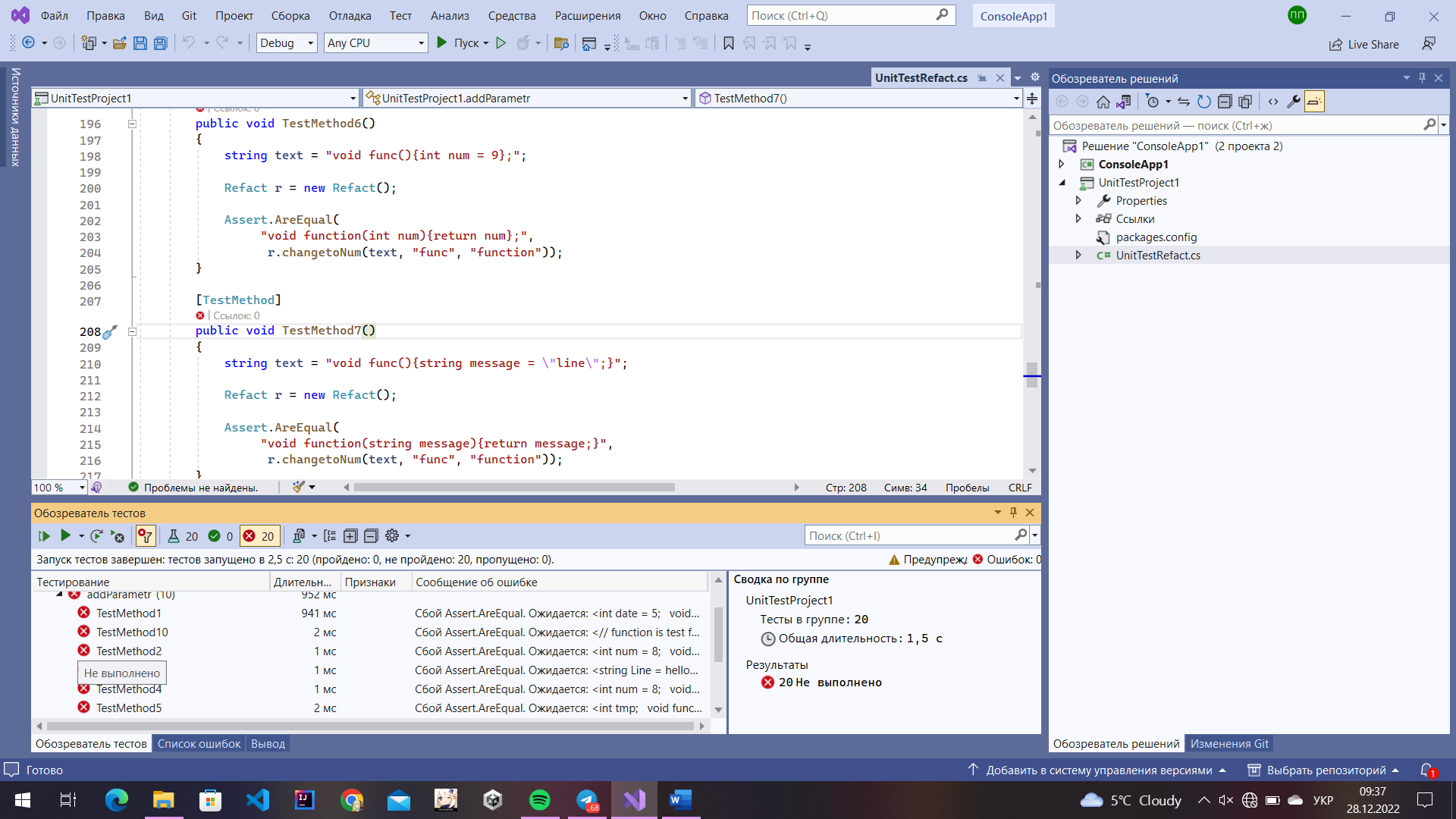
r.changetoNum(text, "func", "function", true, true));

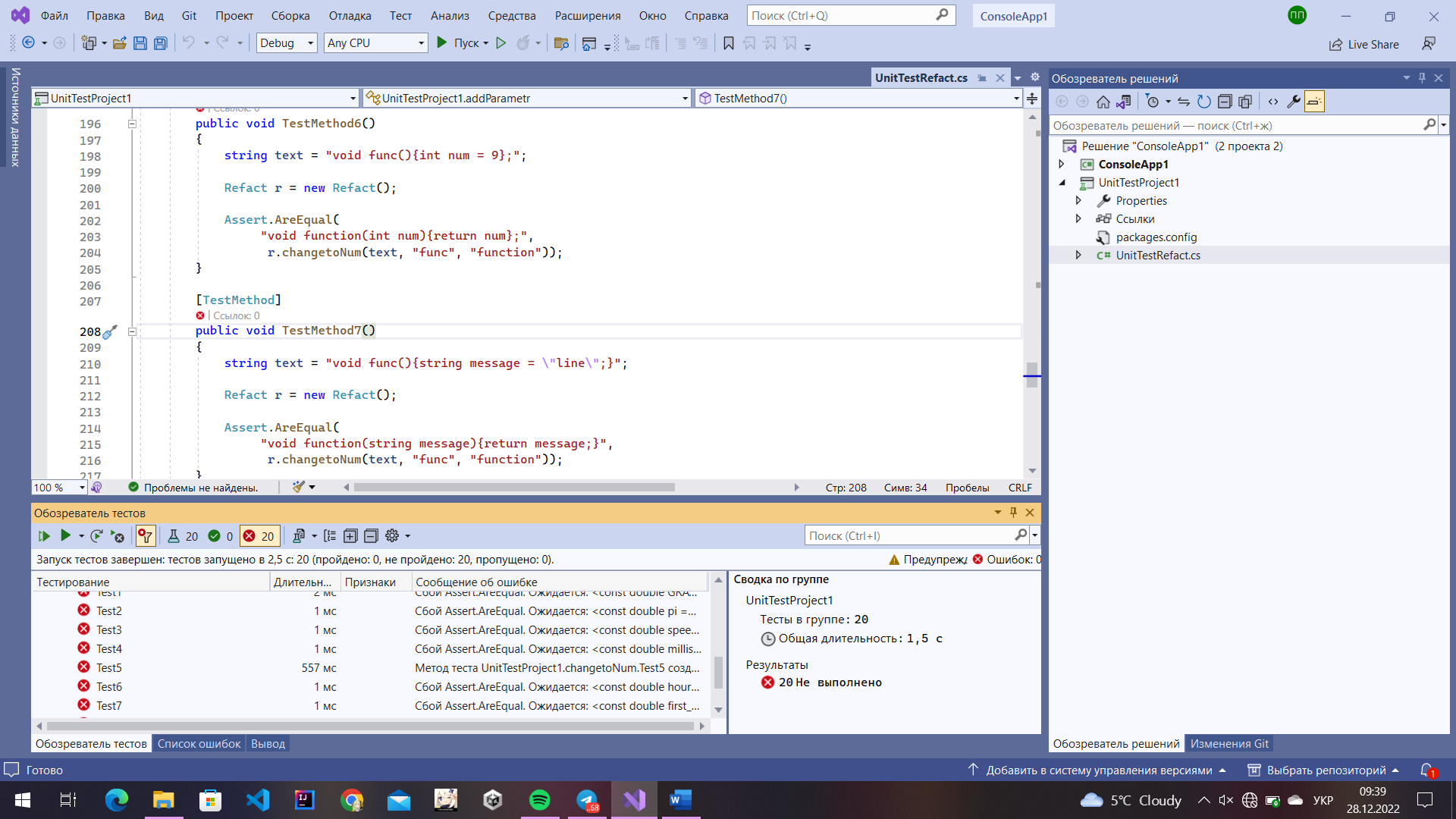
}

}

}

1. Результати роботи програми





1. Під час виконання лабораторної роботи були отримані навички модульного тестування. Модульне тестування - це метод, який передбачає окреме тестування кожної частини програмного коду, відомої як модуль. Основна мета модульного тестування полягає в тому, щоб ізолювати кожну частину програми і переконатися, що вона працює правильно. Модульні тести встановлюють жорсткий "контракт", за яким має функціонувати перевіряємий код. Це надає деякі переваги. Завдяки модульному тестуванню помилки можна виявити на ранніх етапах розробки програмного забезпечення, що збільшує ефективність і швидкість розробки. Модульне тестування дозволяє розробникам бути впевненими у коректній роботі модулів при зміні коду (рефакторингу). Оскільки модульне тестування вимагає написання тестів для всіх функцій і методів програми, помилки можуть бути швидко виявлені та виправлені.