МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І. І. МЕЧНИКОВА
ФАКУЛЬТЕТ МАТЕМАТИКИ, ФІЗИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА МЕХАНІКИ, АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЧИСЕЛЬНІ МЕТОДИ

Модуль 1. Елементарні обчислення

ЕЛЕКТРОННИЙ МЕТОДИЧНИЙ ПОСІБНИК

до лабораторних робіт
студентам факультету математики, фізики та інформаційних технологій
першого (бакалаврського) рівня освіти, спеціальності 122 «Комп’ютерні науки»,
126 «Інформаційні системи і технології»

ОДЕСА
ОНУ
2023

УДК 519.61/.64(075.8)
Ч-67

Укладачі:

О. П. Царенко, старший викладач кафедри механіки, автоматизації та інформаційних технологій;
О. А. Недєва, викладач кафедри механіки, автоматизації та інформаційних технологій.

Рецензенти:

О. Д. Кічмаренко, доктор фіз.-мат. наук, доцент, завідувач кафедри оптимального керування та економічної кібернетики ОНУ імені І.І.Мечникова;
Н. П. Волкова, кандидат технічних наук, доцент, завідувач кафедри прикладної математики та інформаційних технологій Національного університету «Одеська політехніка».

Рекомендовано вченою радою факультету математики, фізики та інформаційних технологій ОНУ імені І.І.Мечникова.
Протокол № 6 від 8 червня 2023 року.

Ч-67 **Чисельні** методи. Модуль 1. Елементарні обчислення [Електронний ресурс] : електрон. метод. посібник до лаб. робіт студентів факультету математики, фізики та інформаційних технологій першого (бакалаврського) рівня освіти, спец. 122 «Комп'ютерні науки», 126 «Інформаційні системи і технології» / уклад.: О. П. Царенко, О. А. Недєва, – Одеса : Одес. нац.. ун-т ім. І. І. Мечникова, 2023. – 259 с. – 8,3 МБ.

Пропонований методичний посібник стане у нагоді при виконанні лабораторних робіт з обов'язкових дисциплін «Чисельні методи» та «Методи обчислень», які викладаються студентам першого (бакалаврського рівня вищої освіти спеціальностей 122 «Комп'ютерні науки», 126 «Інформаційні системи і технології» Одеського національного університету імені І. І. Мечникова.

УДК 519.61/.64(075.8)

Загальні методичні рекомендації щодо організації виконання лабораторних робіт та контрольних завдань з навчальної дисципліни «Чисельні методи»

Обов’язковим етапом всіх без винятку завдань за даною навчальною дисципліною є розробка і подальша реалізація обчислювальної програми, результатом виконання якої є певний набір чисельних даних, який має бути представлений в заданому форматі задля подальшого опрацювання та аналізу.

Під час виконання деяких із завдань буде виникати необхідність у використанні певних алгоритмів, методів, функцій або наборів числових даних, які вже були раніше вами або отримані та оброблені, або розроблені та протестовані.

З цієї причини ми скористуємося можливістю об’єднання всіх алгоритмів, які нами розробляються, в складі власної єдиної динамічної бібліотеки чисельних методів.

В свою чергу, алгоритми, які відносяться до певних розділів чисельних методів, наприклад, до розділу «Інтерполяція функцій», ми будемо об’єднувати в рамках окремих класів, які, в свою чергу, будуть входити до складу зазначеної динамічної бібліотеки класів.

Такий підхід до організації процесу розробки складних застосувань має певні переваги.

По-перше, студент вчиться розробляти певні програмні компоненти та типи даних, які у подальшому можна використовувати в інших незалежних додатках.

По-друге, студент набуває навиків компонування виконуваних додатків, коли їх складові частини попередньо вже відкомпільовані і розміщені в самостійних бібліотеках.

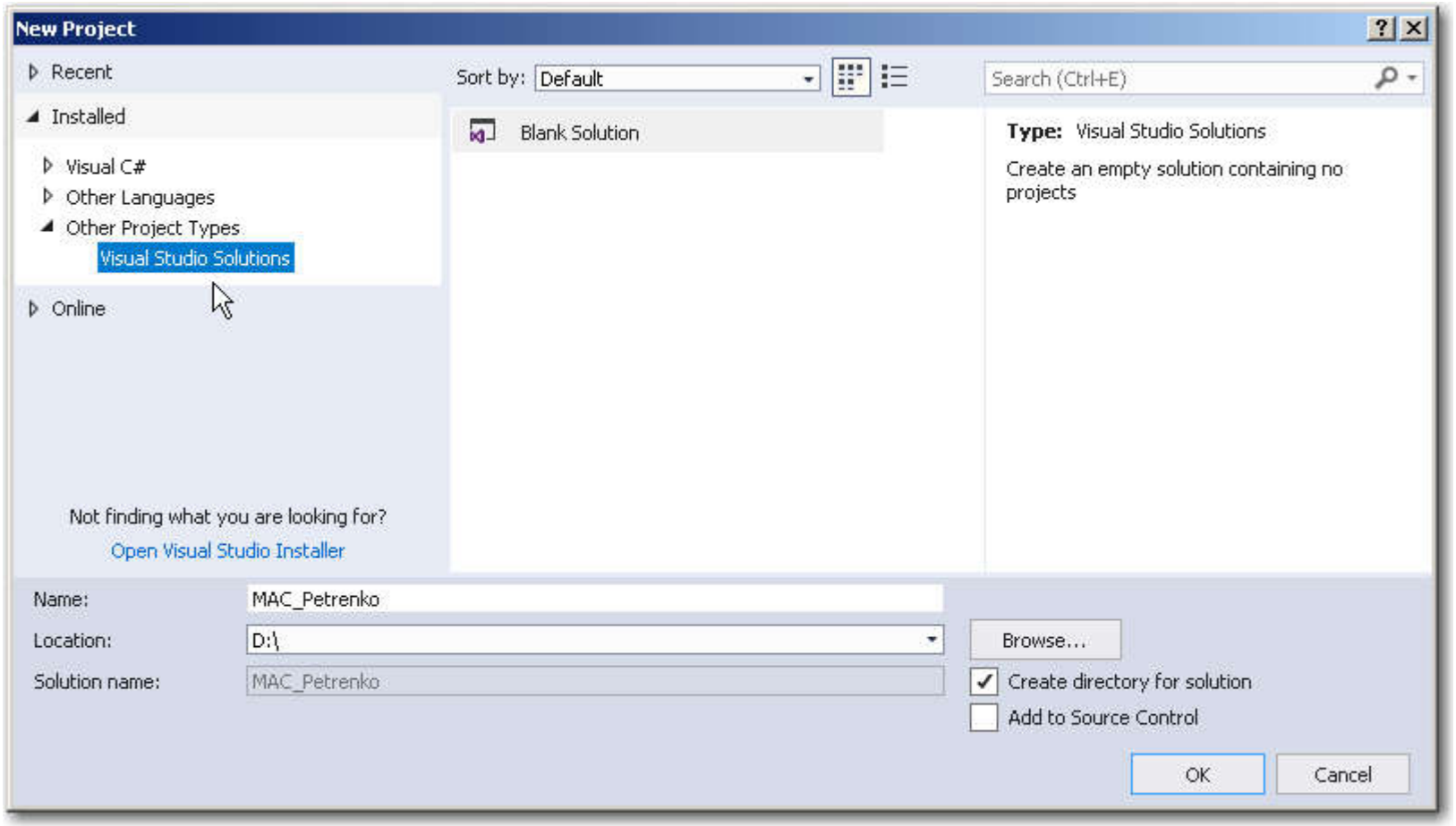
По-третє, студент вчиться розробляти додатки, складові частини якого реалізовані на різних мовах програмування, наприклад, на **`Fortran`**, **`C++`** або **`C#`**.

Крім цього, бібліотеку чисельних методів, яку буде розроблено студентом в рамках даної навчальної дисципліни, можна також використовувати під час виконання курсової роботи за суміжною навчальною дисципліною або при реалізації обчислень в рамках дипломної роботи.

Повністю оформлений посібник для «потенційного користувача» динамічної бібліотеки чисельних методів може бути основою ***курсової роботи*** з навчальної дисципліни «***Чисельні методи***» та оцінюватися самостійною підсумковою оцінкою наприкінці навчального семестру.

Середовище розробки застосувань, яким ми будемо користуватися при виконанні лабораторних і контрольних завдань, це – **MS Visual Studio (MSVS)**, в складі якого є повний інструментарій для сучасної мови програмування **C#**.

Першим етапом роботи буде створення загального робочого простору **MSVS – Visual Studio Solution**, призначеного для упорядкованого (систематизованого) розміщення всіх програм, бібліотек та інших файлів даних, які будуть нами розроблені. Для цього стартуємо середовище розробки **MSVS** та генеруємо в ньому новий проект –



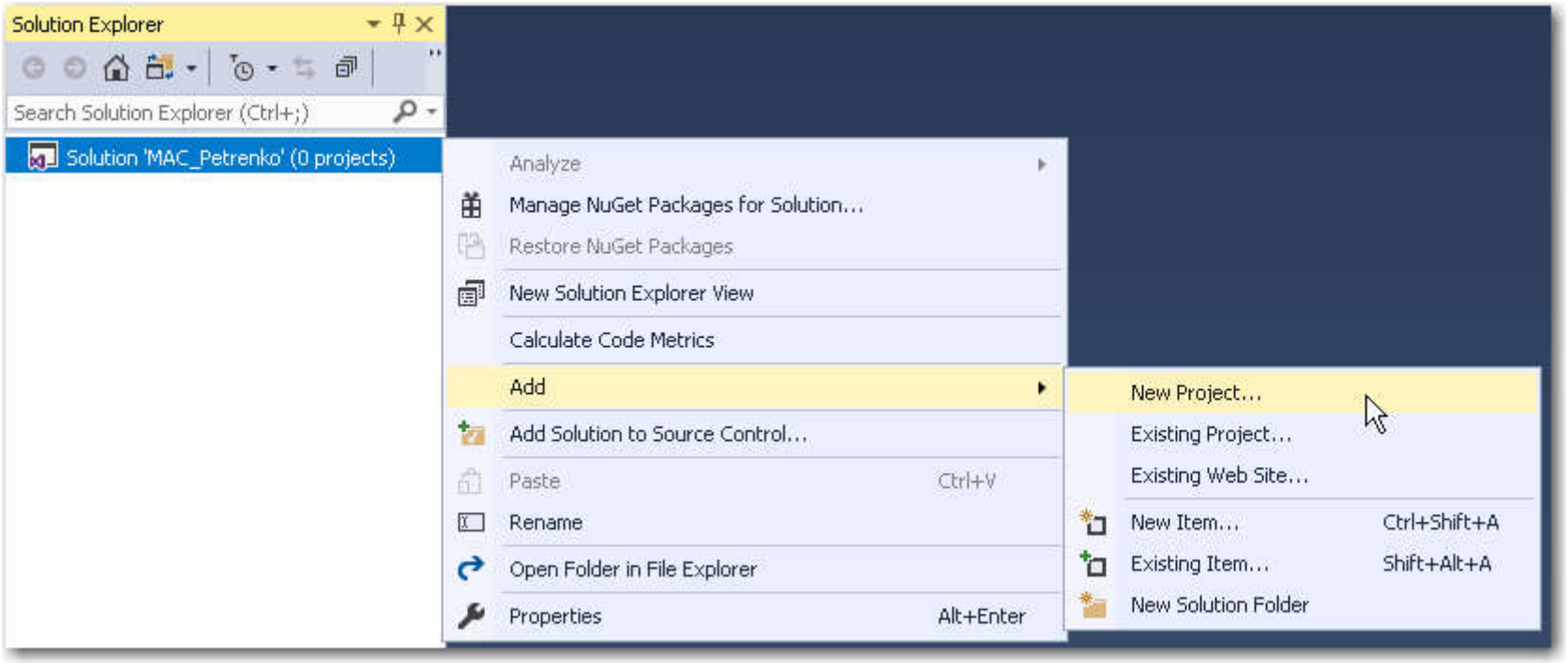
Слід звернути увагу на параметри, які задаються:

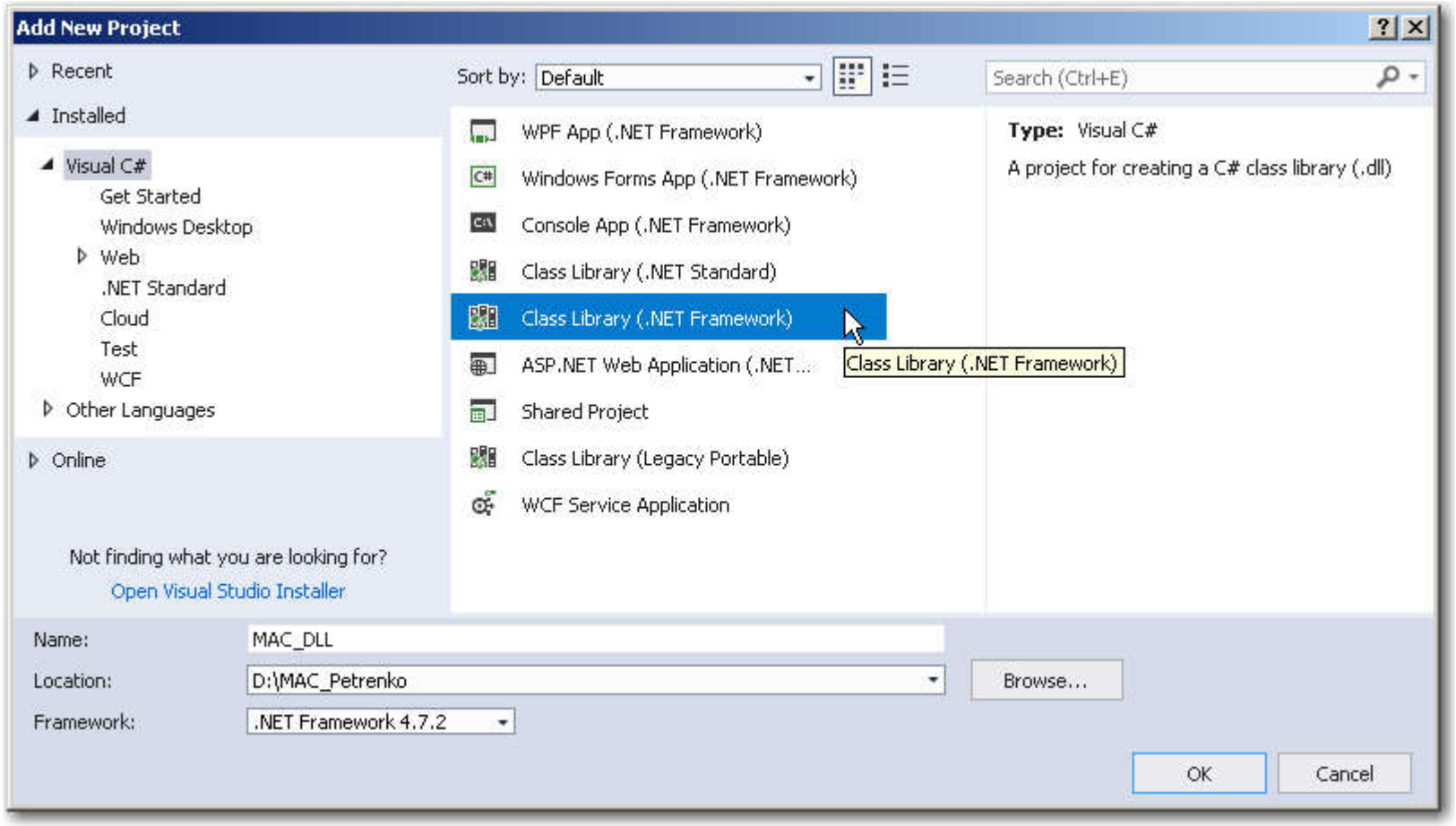
- тип проекту (**`Project types: Other Project Types – Visual Studio Solutions`**),
- ім’я проекту (**`Name: MAC_Petrenko`**).

Пам’ятайте, що ймовірну Вашу помилку, яку було допущено на етапі генерації любого з проектів **`MSVS`**, вкрай важко буде потім виправити (якщо взагалі це буде можливо). Тому, будь ласка, будьте уважними при виконанні наступних інструкцій.

Ім’я даного проекту (тобто робочого простору) формується у відповідності до особистого прізвища та ініціалів реального студента. В нашому прикладі це віртуальний студент – ***Петренко Ігор Миколайович***. Бажано розташовувати робочий простір (позиція **`Location`** в кореневому каталозі одного з логічних розділів вашого жорсткого диска.

Далі, у вже існуючому робочому просторі **`MAC_Petrenko`**, згенеруємо новий проект – проект бібліотеки класів **`MAC_DLL`**:





Перші три символи, які прописані в імені бібліотеки – **MAC** – заголовні літери із назви даної навчальної дисципліни – **Methods of Approximate Calculations**.

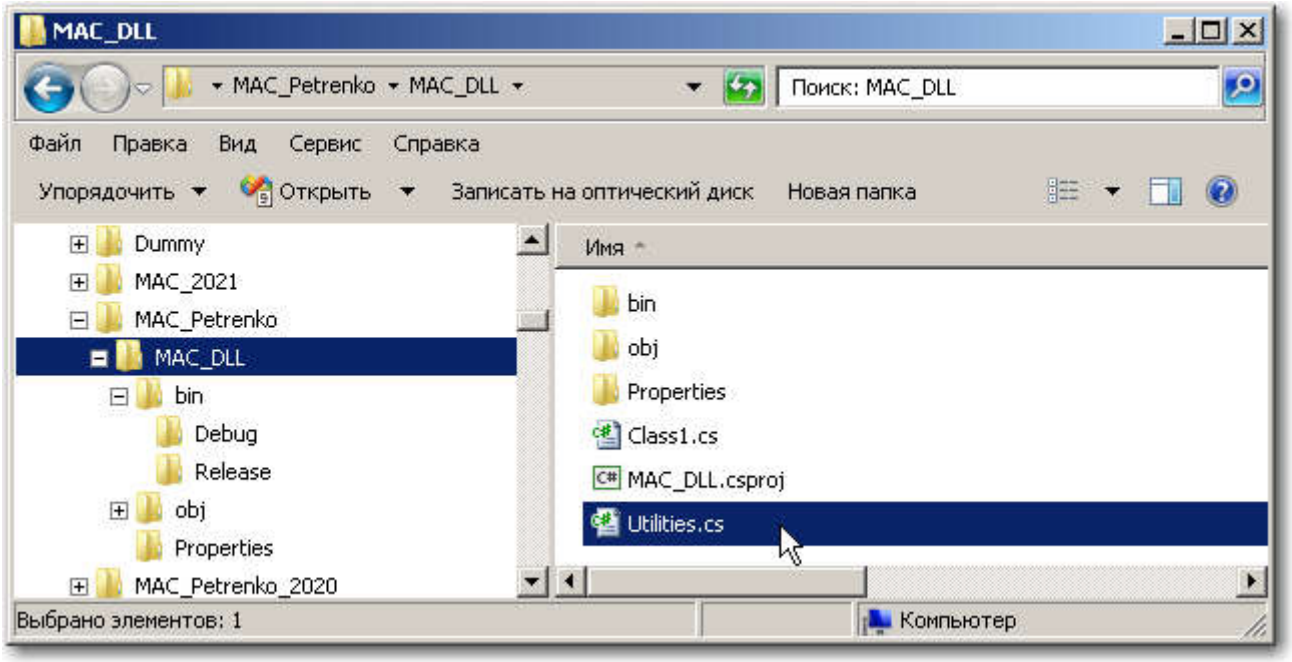
Бібліотеки класів, які плануються до використання для інших навчальних дисциплін, Ви також можете ідентифікувати подібними літерними «масками».

Зробимо перший крок на шляху створення власної бібліотеки класів **MAC_DLL**.

Додамо до складу цієї бібліотеки вже існуючий файл **Utilities.cs**, який містить відповідний клас, в якому вже створено метод, який буде реалізовувати для нас етап формування файлу результатів в форматі ***.txt** в кожній наступній роботі або завданні, які ви будете виконувати. Такий файл буде обов'язковим елементом звіту студента про виконану роботу (завдання).

Файл **Utilities.cs** для подальшого копіювання вам надається безпосередньо викладачем (або він розташований на відповідному ресурсі **GoogleDisk**) . Отже, будемо вважати, що цей файл із кодом є у вашому розпорядженні.

Стандартним шляхом – за допомогою «Провідника» – скопіюємо файл **Utilities.cs** до робочої директорії вже існуючого проекту **MAC_DLL**:

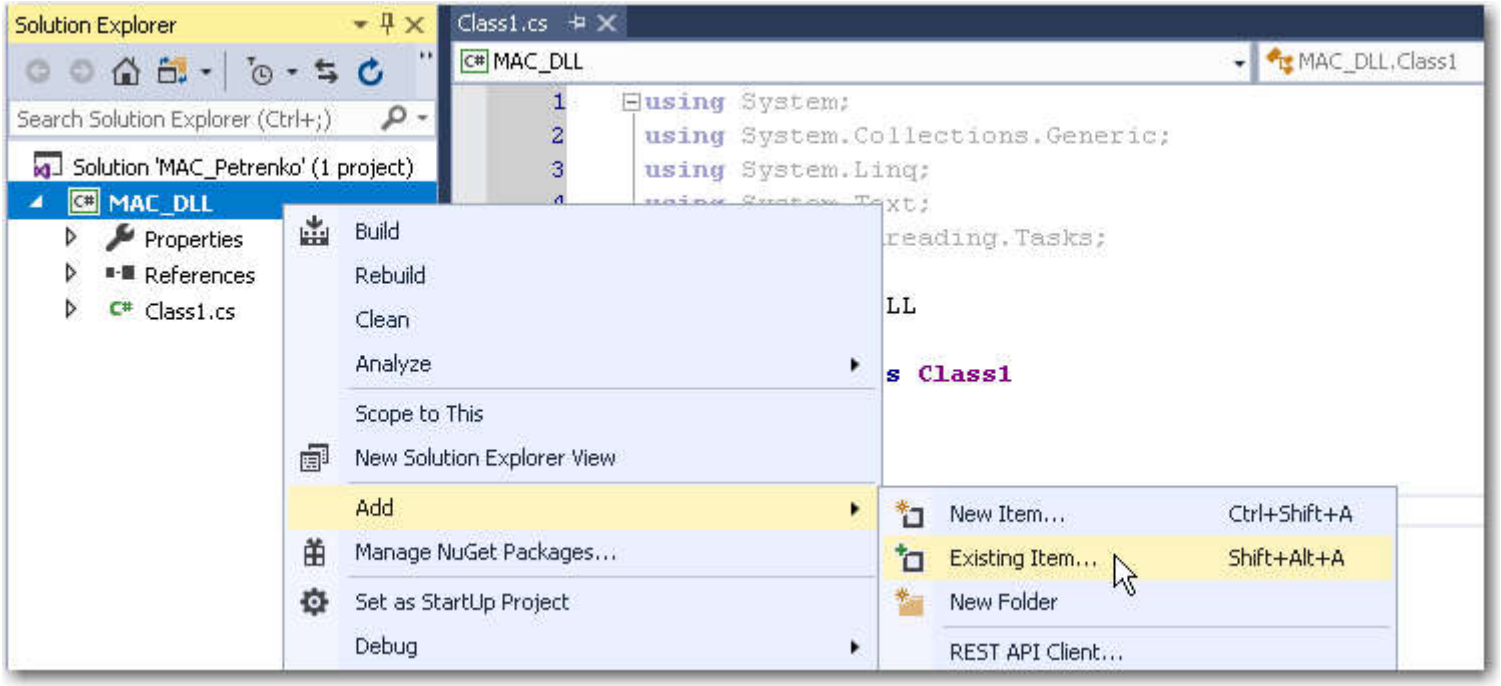


Тепер слід цей файл ввести до складу майбутньої множини файлів нашої бібліотеки класів **MAC_DLL**.

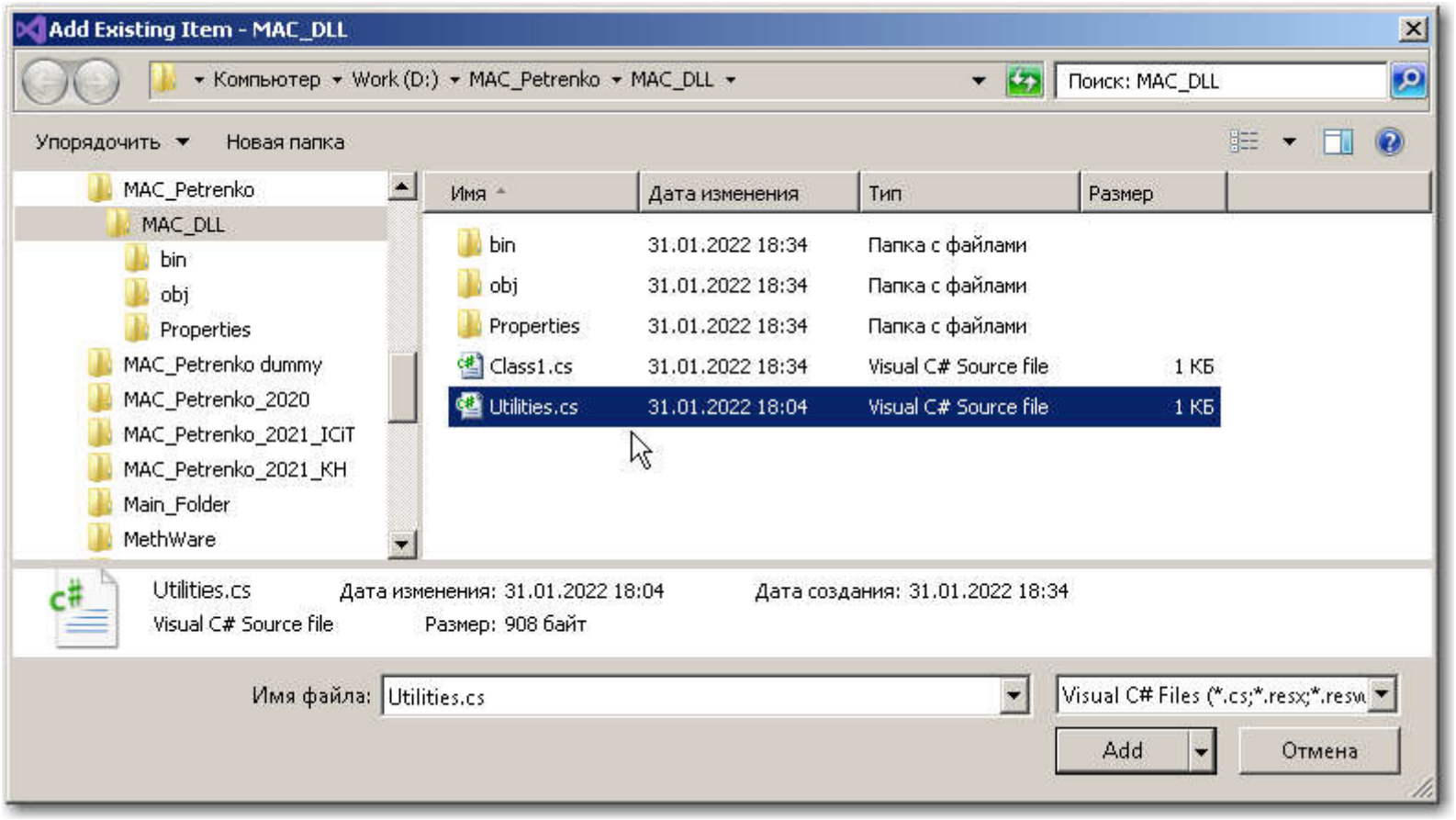
Для цього повернемося до робочого простору **MAC_Petrenko** у середовищі розробки **MS Visual Studio**.

Відкриємо дерево проектів (**Solution Explorer**) та правою кнопкою тицьнемо на імені проекту **MAC_DLL**.

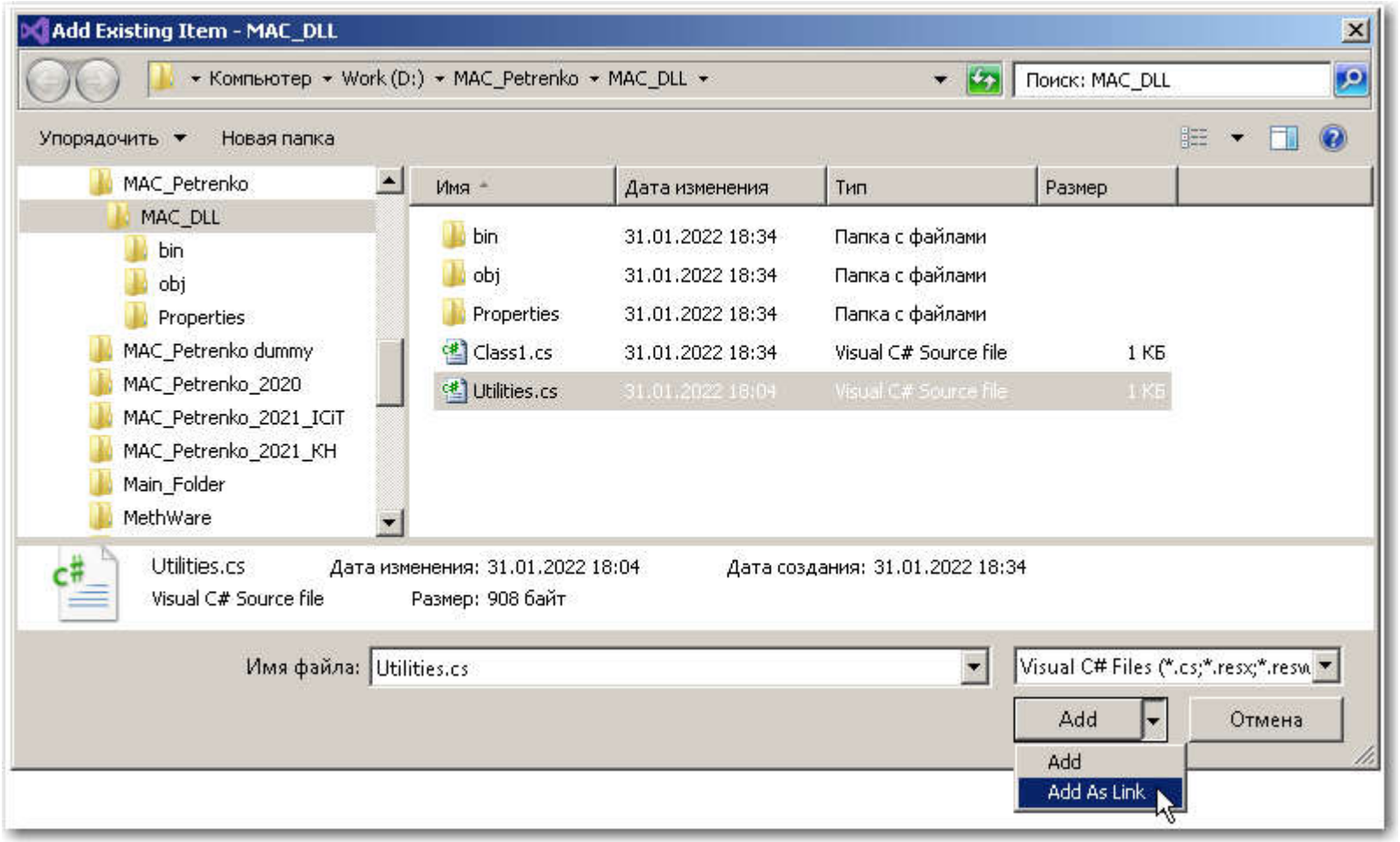
Перед нами з'явиться меню, в якому ми оберемо саме задачу додавання до складу даного проекту вже існуючих (не нових) компонентів (**Existing Items**).



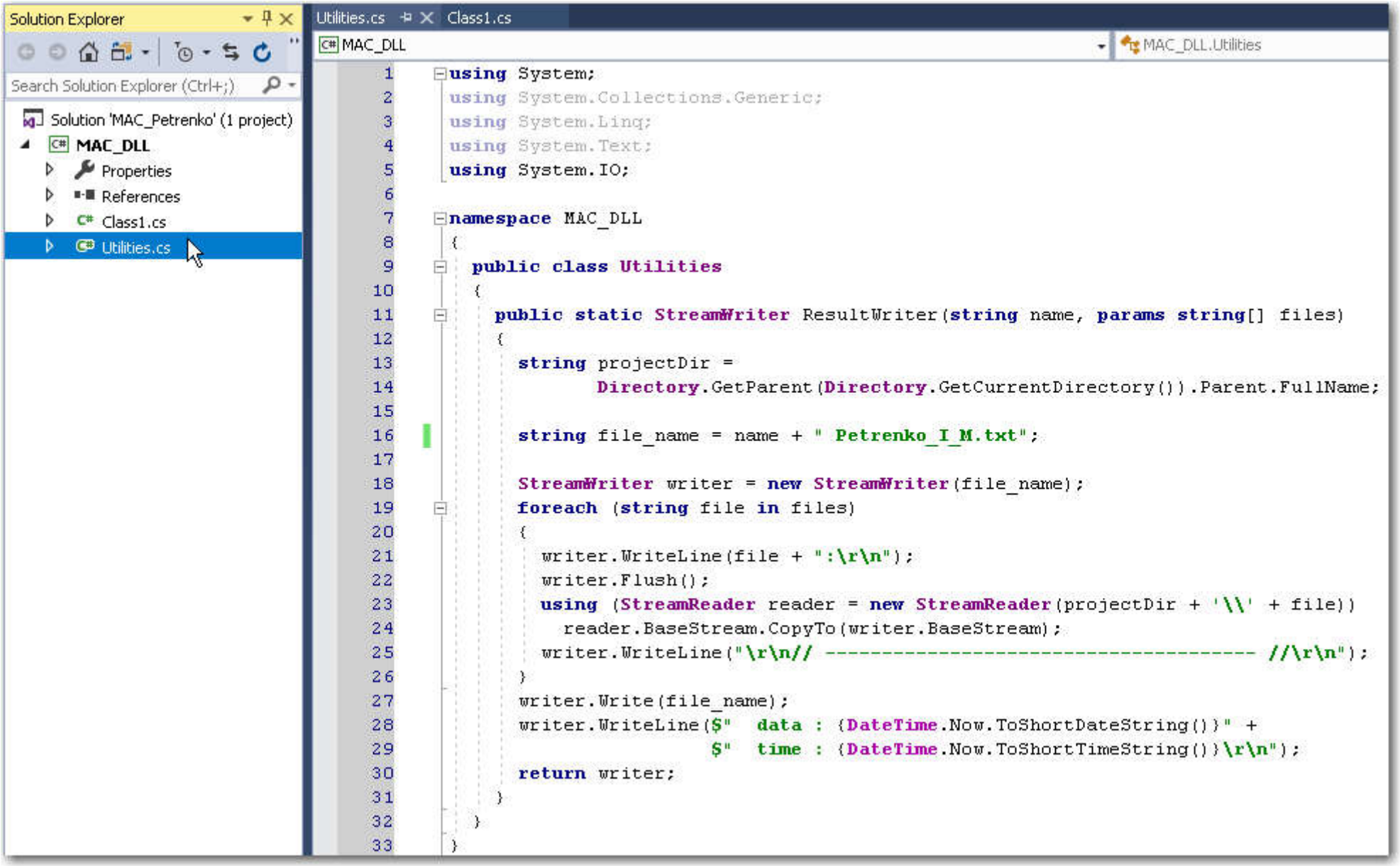
Після того, як ми тицьнемо на цей пункт меню (**`Existing Items`**), перед нами відкриється наступна форма, на якій ми виділимо саме файл **`Utilities.cs`**:



Після цього обираємо метод **`Add As Link`** підключення файлу **`Utilities.cs`** до проекту **`MAC_DLL:`**



Після цього ми будемо спостерігати файл **`Utilities.cs`** в складі даного проекту безпосередньо в робочому просторі середовища розробки застосувань:



Тепер ми можемо редагувати цей файл, компілювати проект бібліотеки класів **`MAC_DLL`**, тощо.

Програмний код цього класу містить поки що один метод.

Ми будемо його використовувати під час роботи з іншими проектами, та тоді й ознайомимося із його суттю.

Єдине, що ви маєте одразу відредагувати в цьому коді – так це суть строкової змінної **`file_name`**.

Замість літералів **`« Petrenko_I_M»`** ви повинні створити запис латинськими літерами ***власного прізвища та ініціалів***.

На цьому підготовчу роботу (перший етап) по організації програмного забезпечення для навчальної дисципліни «***Чисельні методи***» можна вважати завершеною.

Другий етап – **«Створення програмного компонента для візуалізації результатів чисельного експерименту»**

Постановка задачі.

Нехай результати чисельного експерименту є упорядкованою таблицею даних, яка складається з пар дійсних значень «дійсна змінна – значення функції», або мають зображення у вигляді функції, яку задано аналітичним виразом.

Нам необхідно буде здійснювати візуальний аналіз таких даних у вигляді діаграми, безперервного графіка або деякої дискретної точкової залежності.

Оскільки всі програми, які ми будемо розробляти, будуть консольними додатками, нам потрібно створити такий окремий програмний компонент, який буде виконувати зазначені дії та використовувати при цьому наявні засоби візуалізації безпосередньо із середовища розробки **`MS Visual Studio`**.

Вочевидь, що нам доведеться додати до наявного робочого простору **`MAC_Petrenko`** новий клас **`Form_with_Graphics`** та наповнити його відповідним функціоналом.

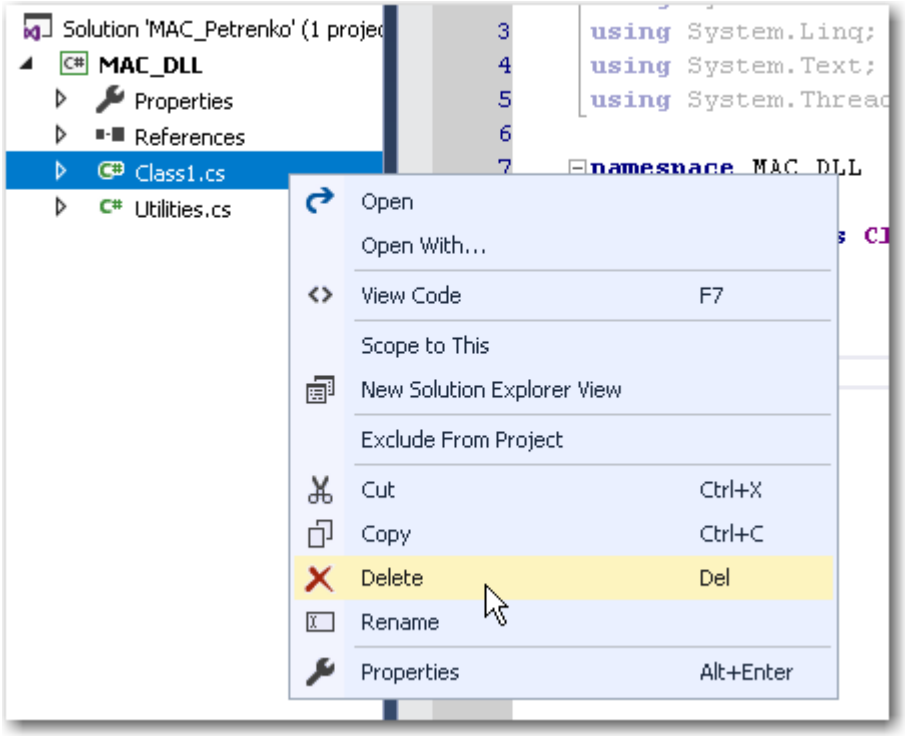
Крім того, в рамках даної лабораторної роботи ми повинні протестувати цей функціонал на певному наборі числових даних, з яким в найближчі часи ми будемо мати справу при вивченні цієї дисципліни.

Для виконання поставленого завдання ми будемо використовувати існуючий компонент **`Chart`** середовища розробки **`MS Visual Studio`**.

Повертаємося до робочого простору **`MAC_Petrenko`**.

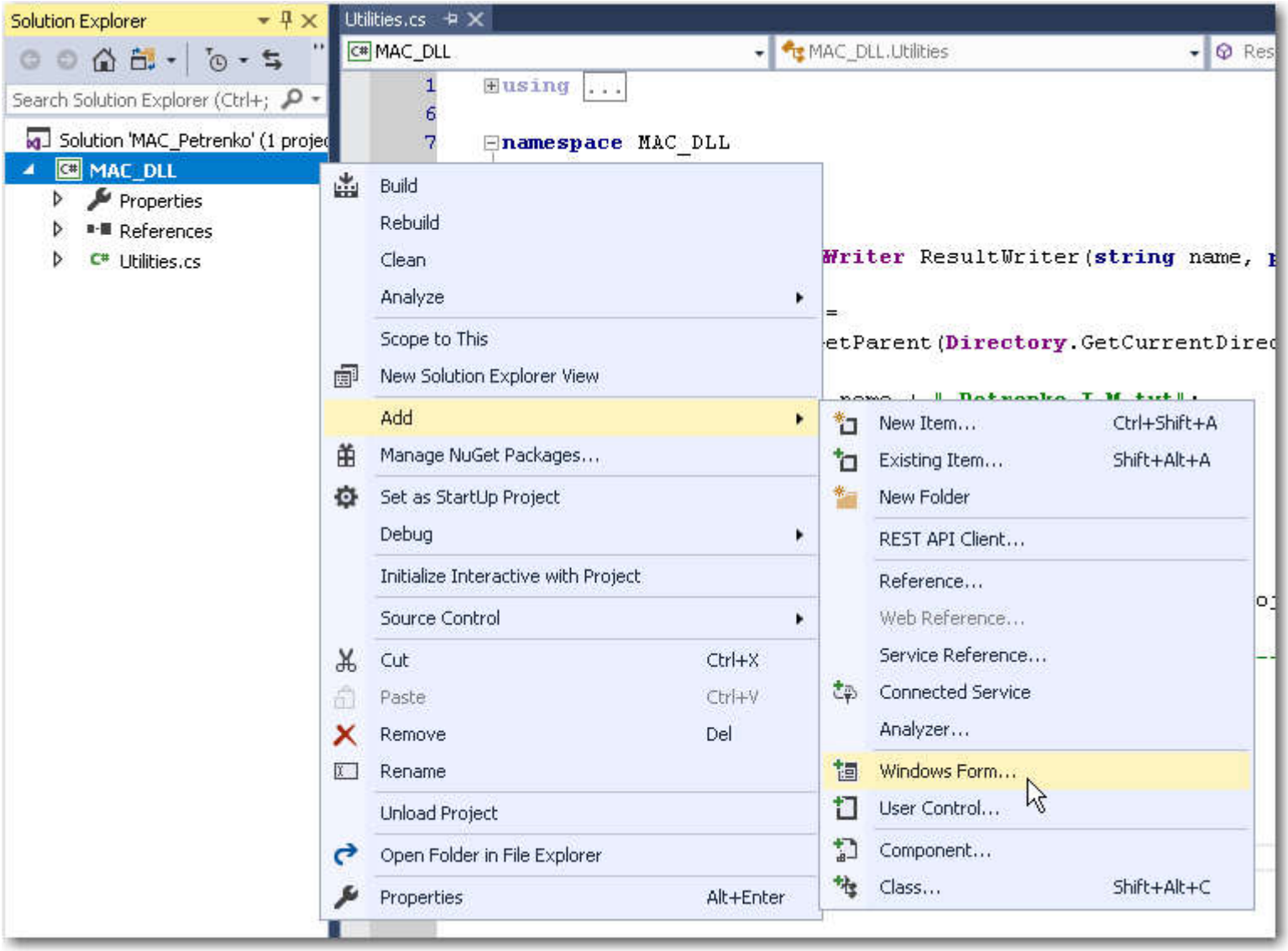
Видаляємо існуючий клас **`Class1`**.

1)

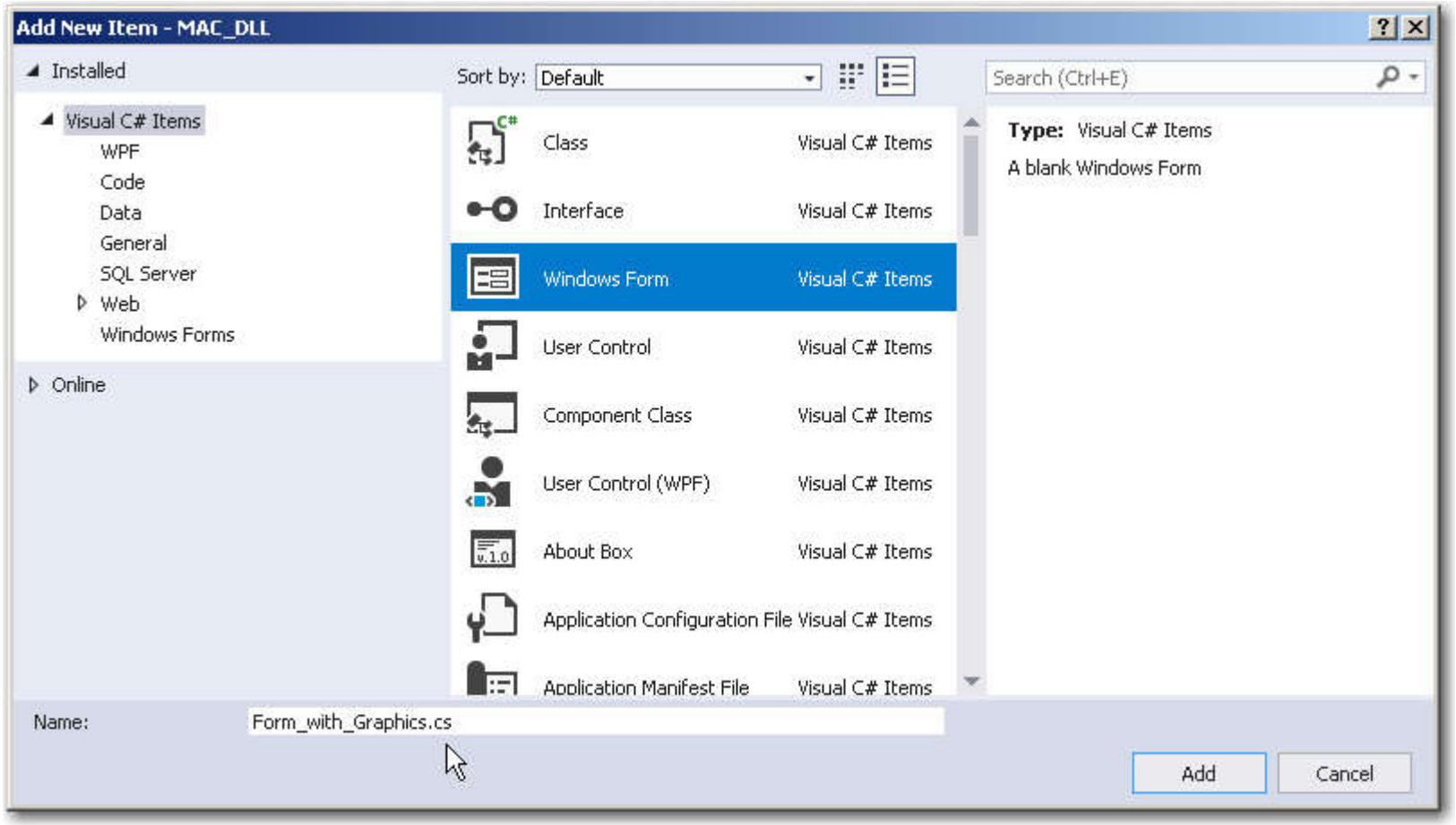


2)

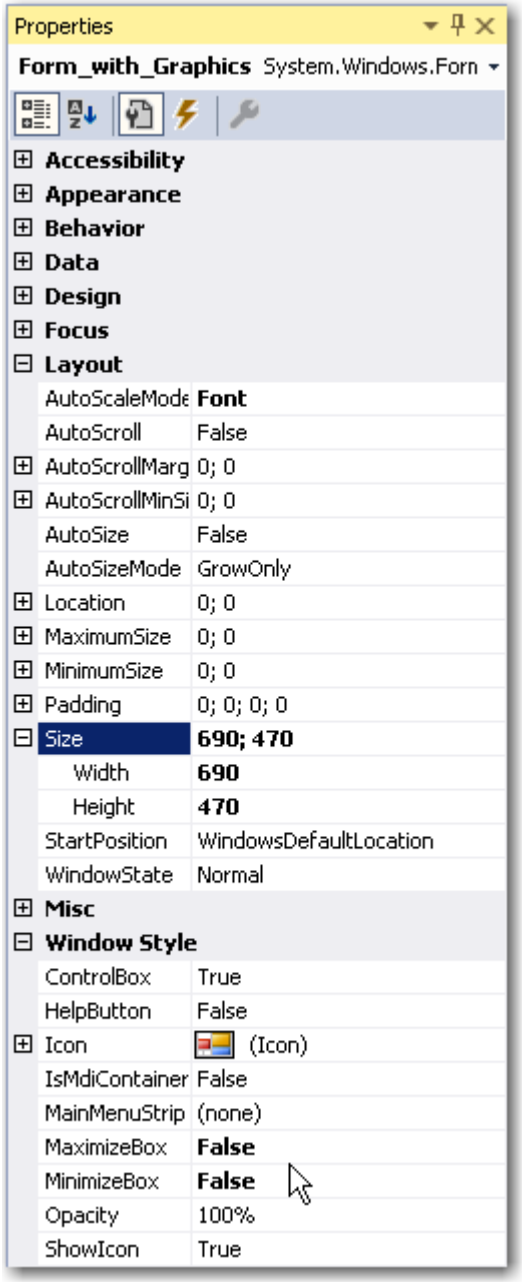
Додаємо до складу проекту **MAC_DLL** **компонент форми – Windows Form**:



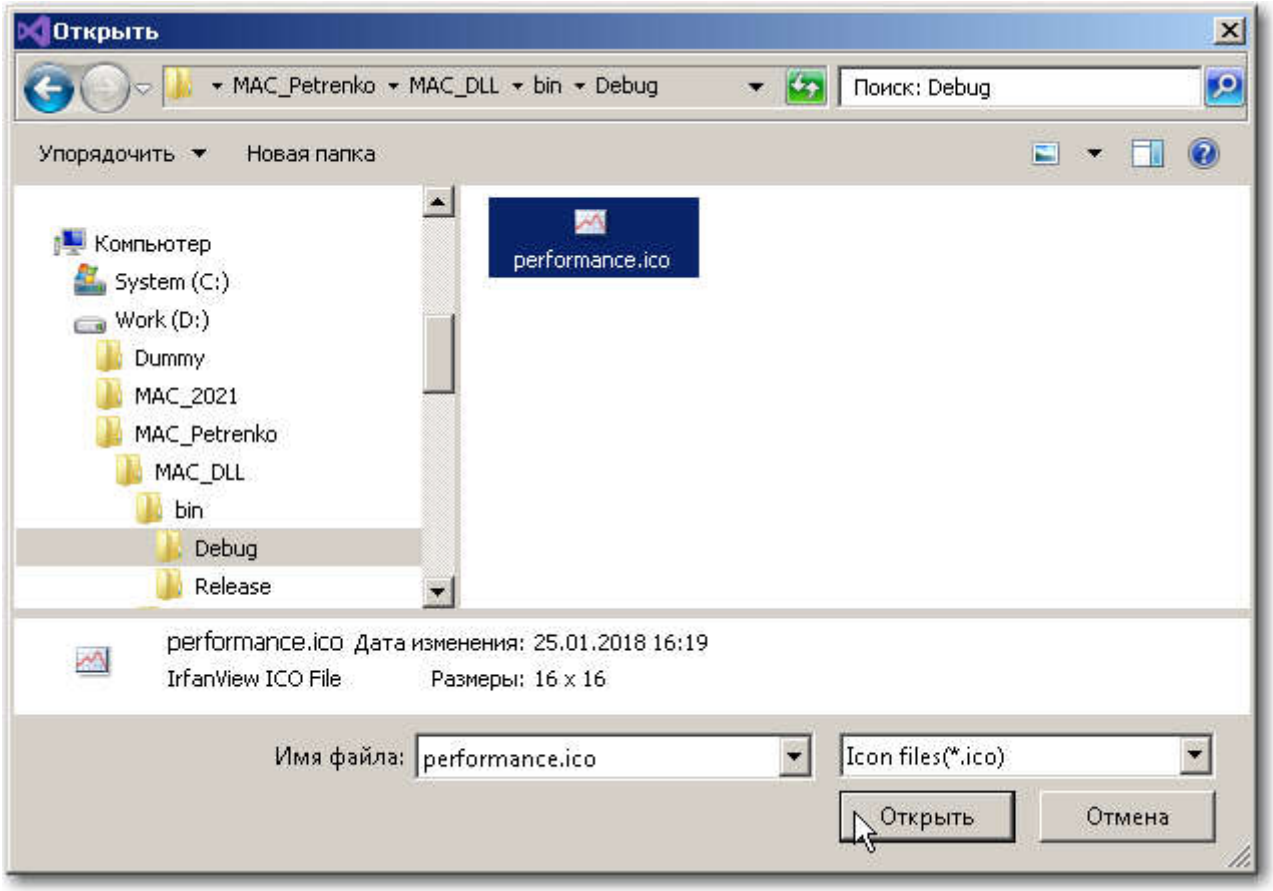
Задаємо м.’я цьому компоненту – **`Form_with_Graphics`**:



Виділяємо мишею нову форму, яку було щойно відкрито для подальшого редагування, і в меню властивостей (**`Properties`**) цієї форми змінюємо її розміри (**`Size`**) та деякі інші параметри, а саме – **`MaximizeBox`** і **`MinimizeBox`**:



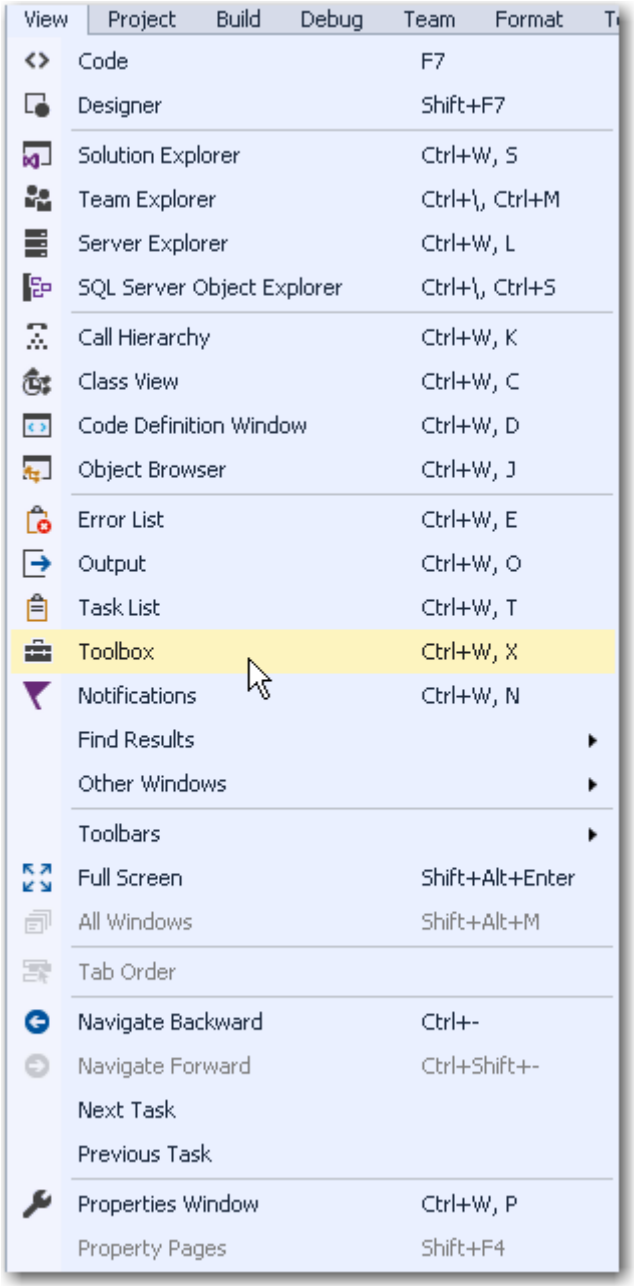
Робимо копію файлу **`performance.ico`** в директорії **`Debug`** даного проекту (із використанням Провідника):



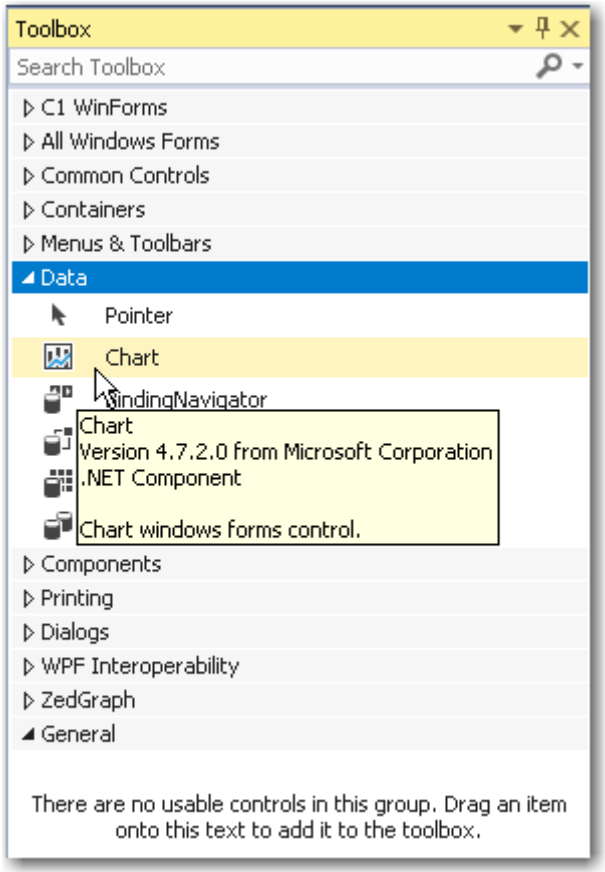
та ініціалізуємо цим компонентом властивість **`Icon`**:

Переносимо на форму компонент **`Chart`**:

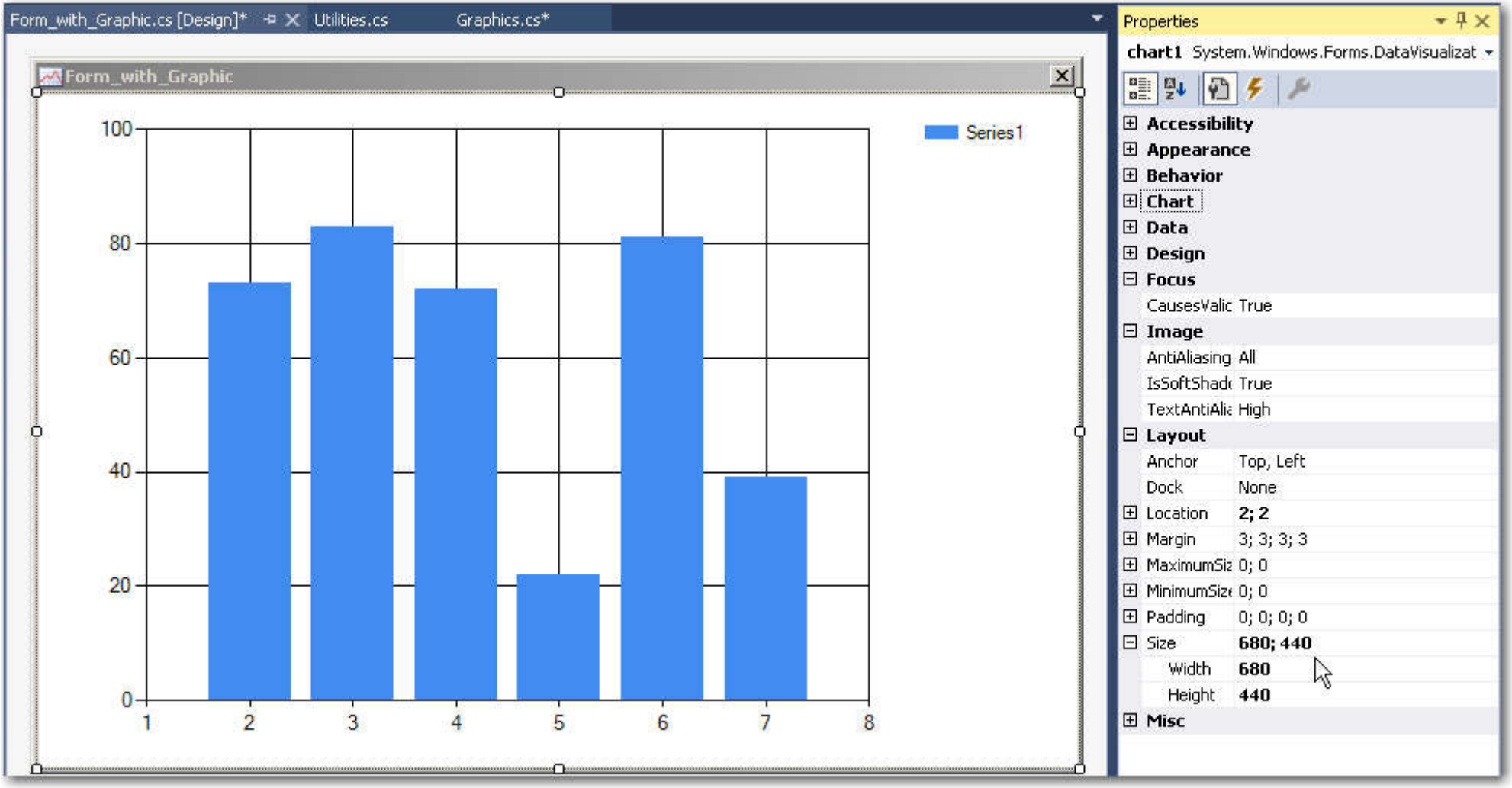
1)



2)

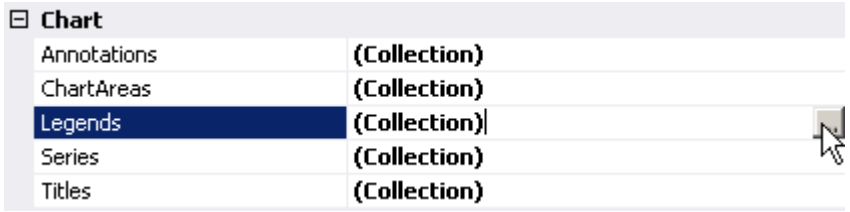


«Розтягуємо» компонент **`Chart`** вздовж внутрішніх границь форми так, аби розміри компонента **`Chart`** стали 680×440:

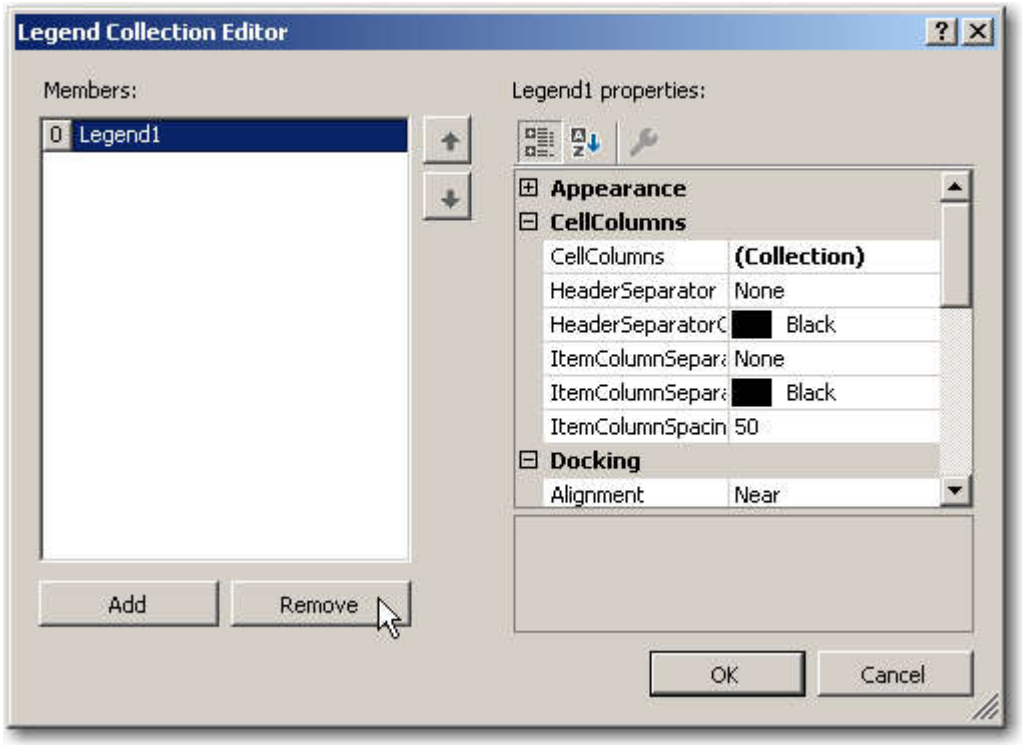


Змінюємо ідентифікатор компонента – нове м.’я – **`Chart_with_Graphics`**:

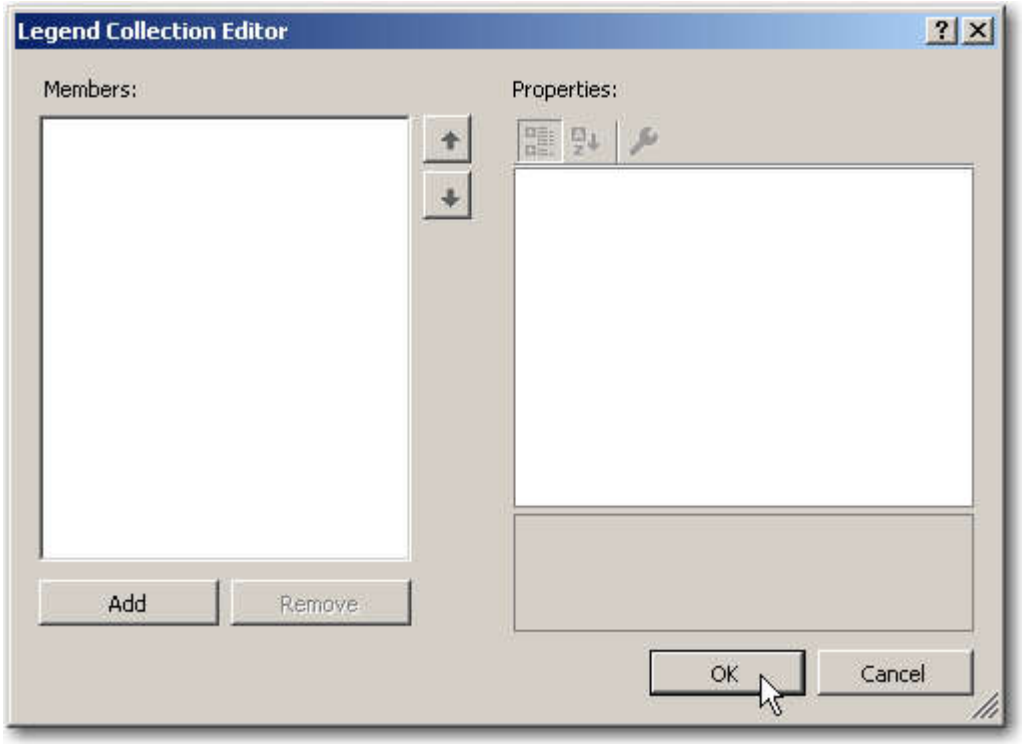
Відкриваємо редактор властивостей **`Chart`**:



Обираємо пункт **`Legends`** – маємо результат:



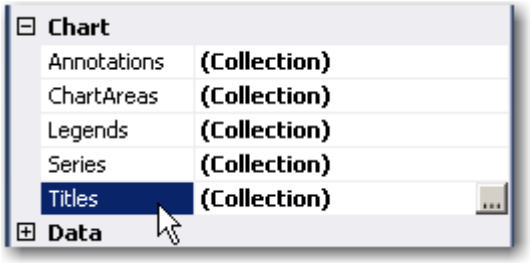
Тицяємо на кнопку **`Remove`** – маємо результат:



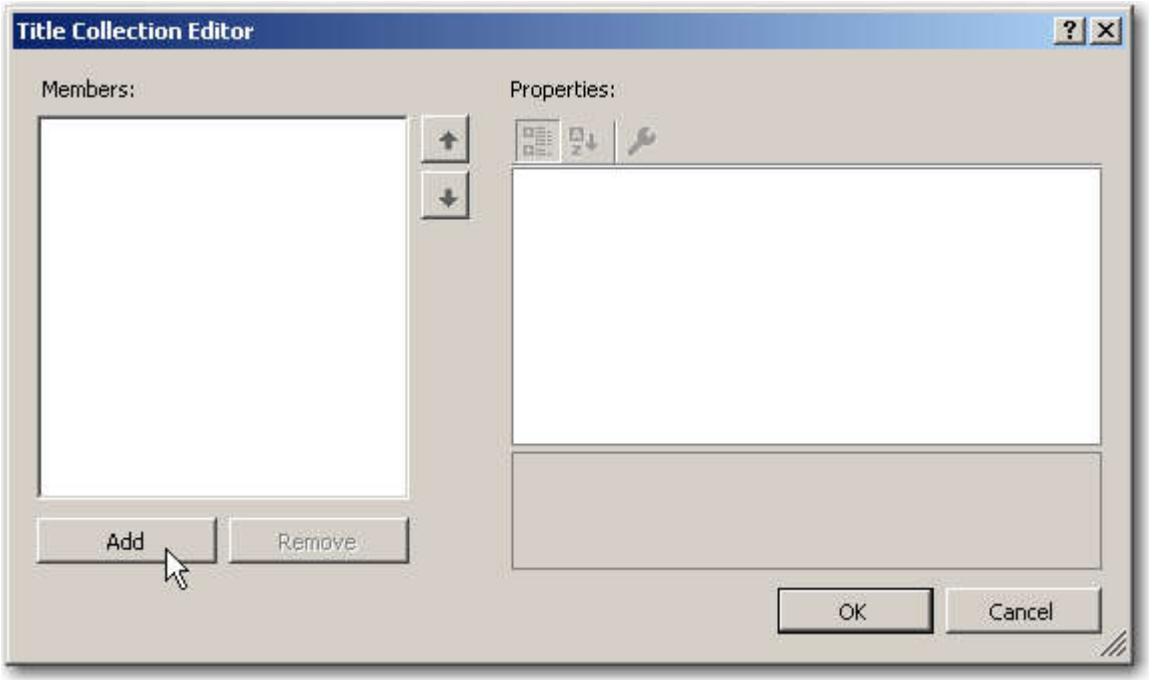
А потім на кнопку **`Ok`** – так ми звільняємо площу компонента **`Chart`** від «запланованого» заголовка для конкретної графічної залежності, який для нас поки що не має ніякого практичного сенсу.

Нас буде цікавити який-небудь загальний заголовок на формі компонента **`Chart`**, який ми зараз і створимо.

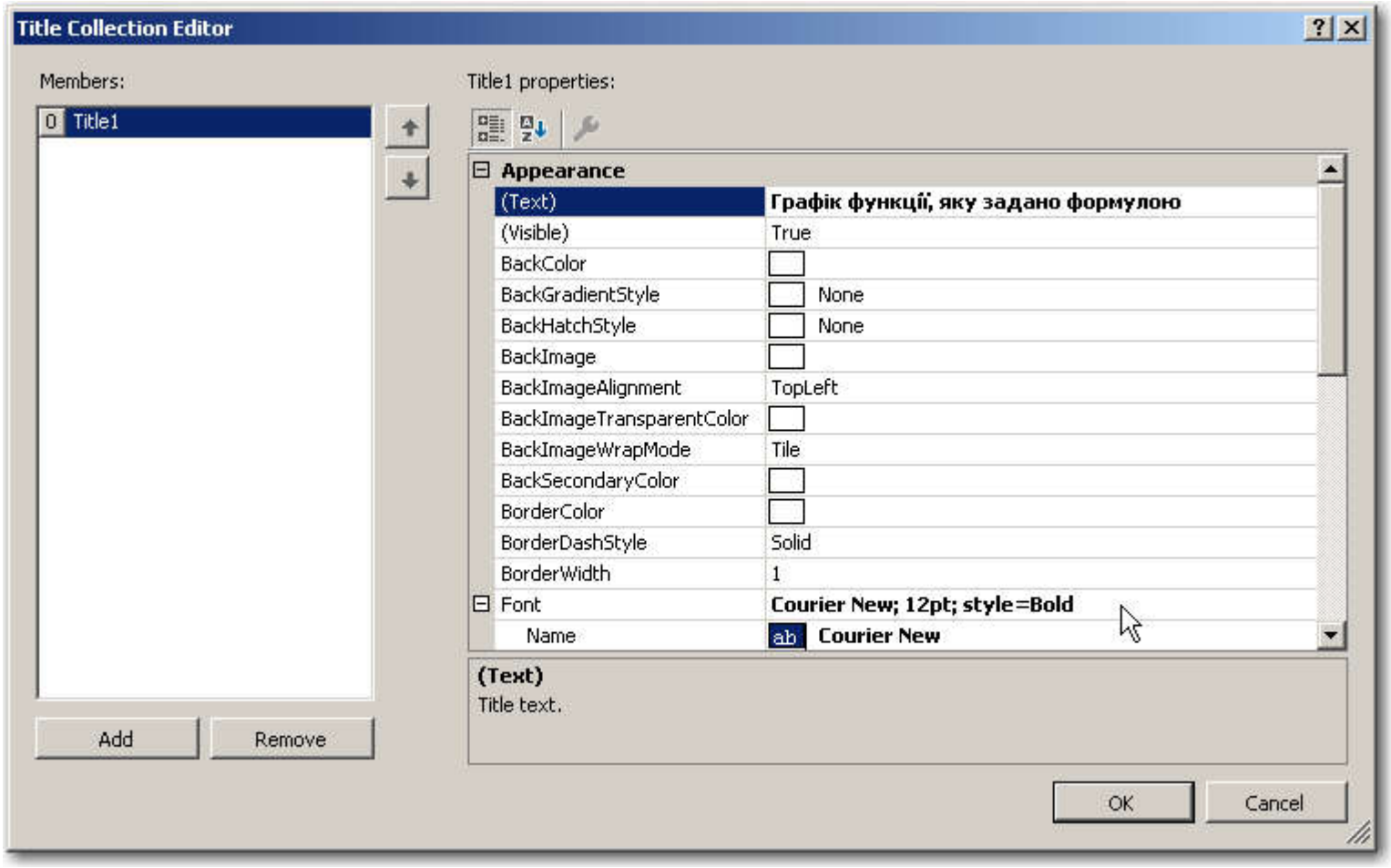
Обираємо властивість **`Titles`**:



Додаємо власне новий заголовок на компоненті:

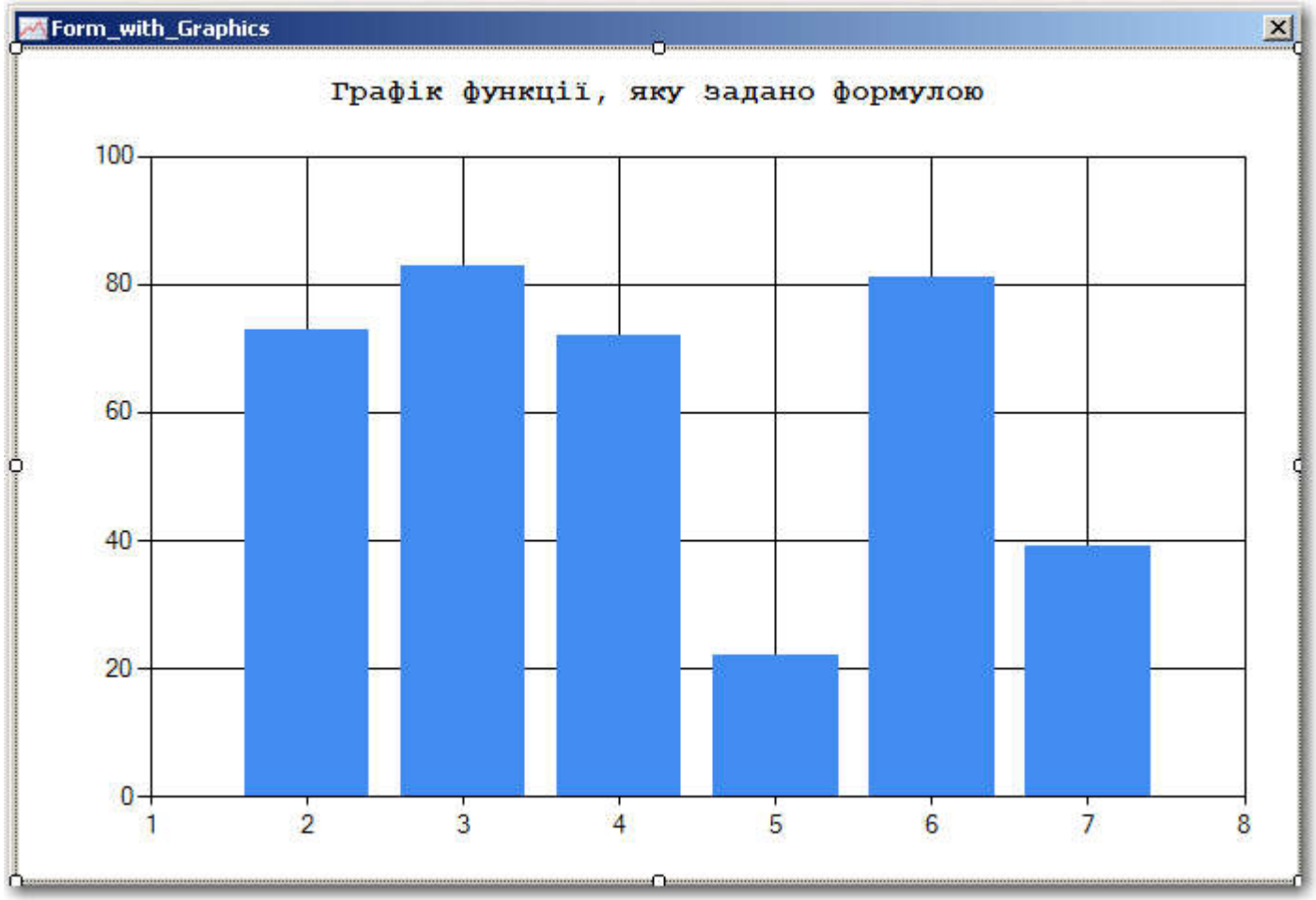


Вносимо зміни до відповідних позицій в таблиці властивостей заголовка – вони виділені жирним шрифтом, та формуємо власне сам заголовок – «**`Графік функції, яку задано формулою`**»:



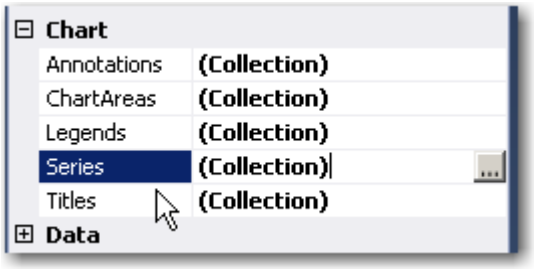
Одночасно змінюємо розмір, тип та стиль шрифту (**`Font`**).

Маємо тепер дещо оновлений вид компонента **`Chart`**:

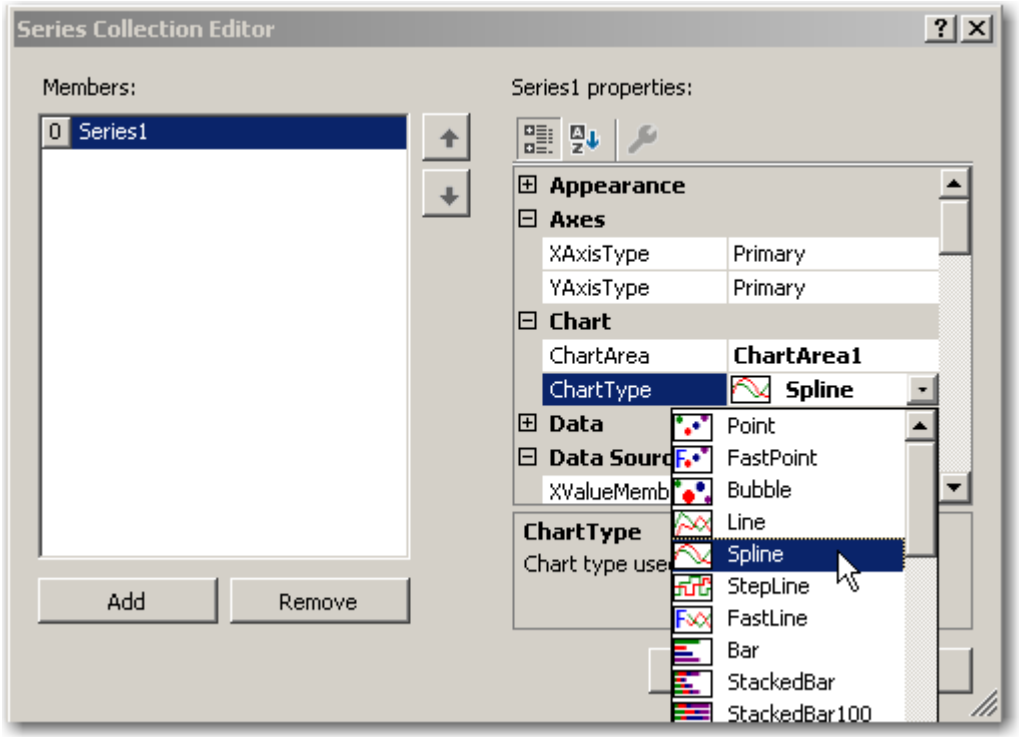


Продовжимо перетворювати зміст цього компонента.

Відкриваємо редактор властивостей **`Series`**:



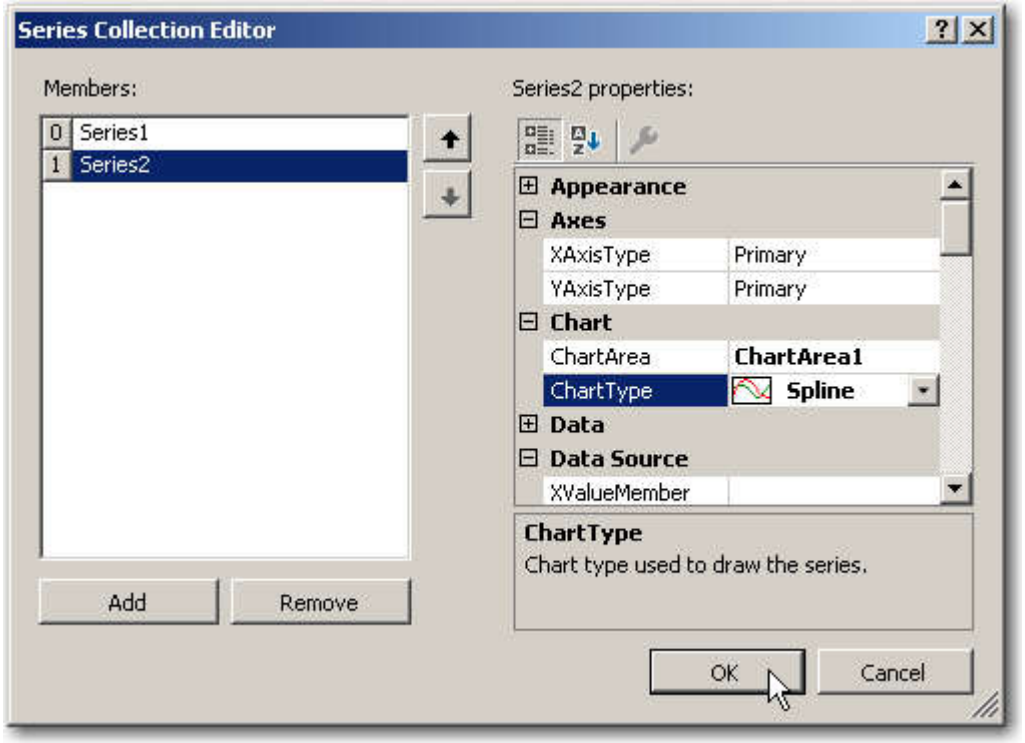
Тут вже існує перший набір даних (серія), тому ми внесемо зміни в його характеристики:



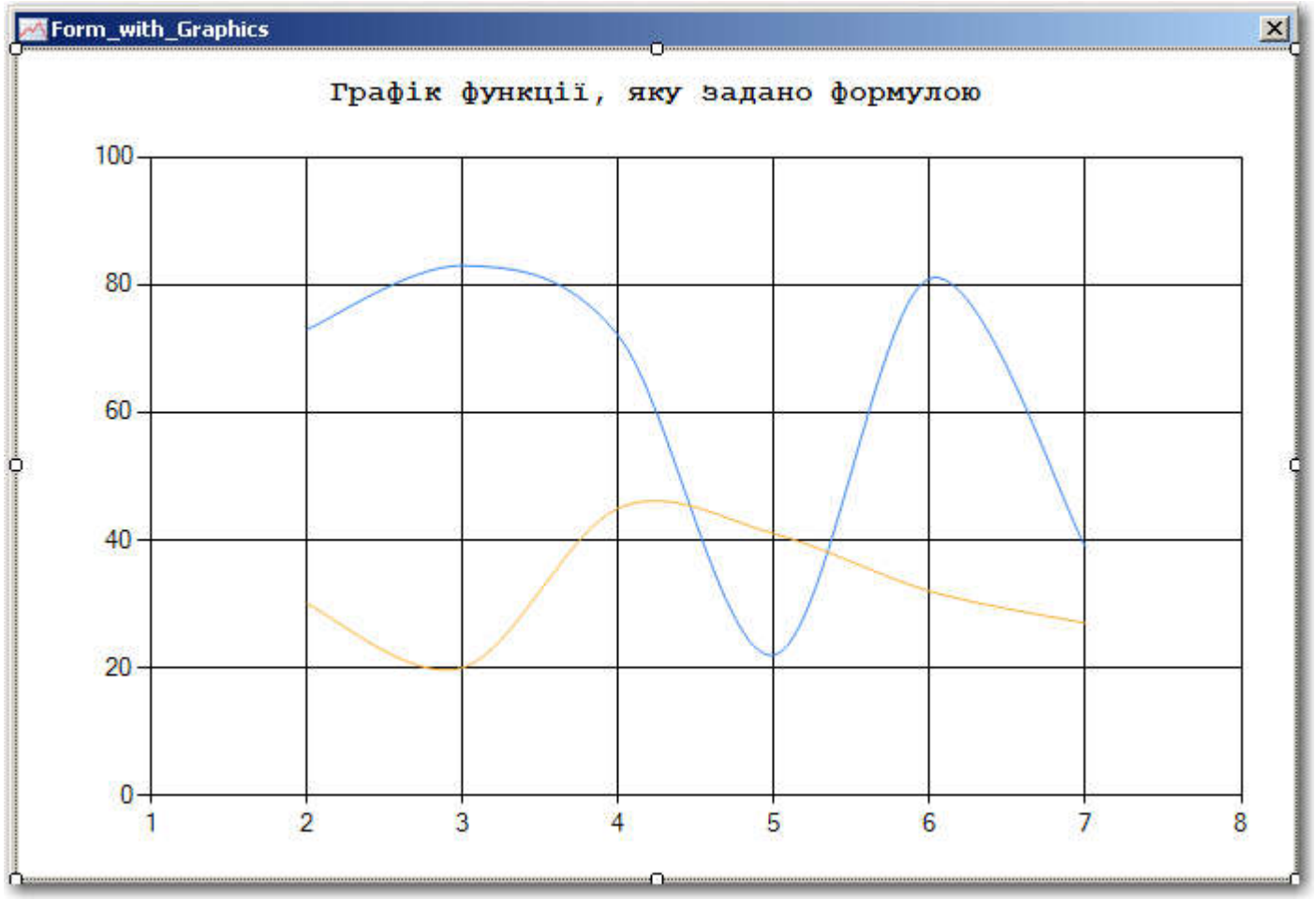
Оскільки ми спочатку будемо відображати лише аналітичні залежності (тобто функції, які задано формулою) – нам потрібний буде гладкий графік, тому задаємо **`ChartType -> Spline`**.

Одразу реалізуємо можливість відображення двох різних залежностей (функцій) на одному компоненті **`Chart`**.

Для цього додамо ще одну послідовність даних (клацнемо на кнопку **`Add`**) та повторимо попередні дії щодо зміні типу числової інформації, яка буде відображатися (**`ChartType -> Spline`**):



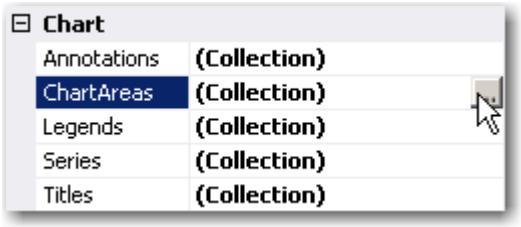
Маємо оновлений вид основної форми **`Chart`**:



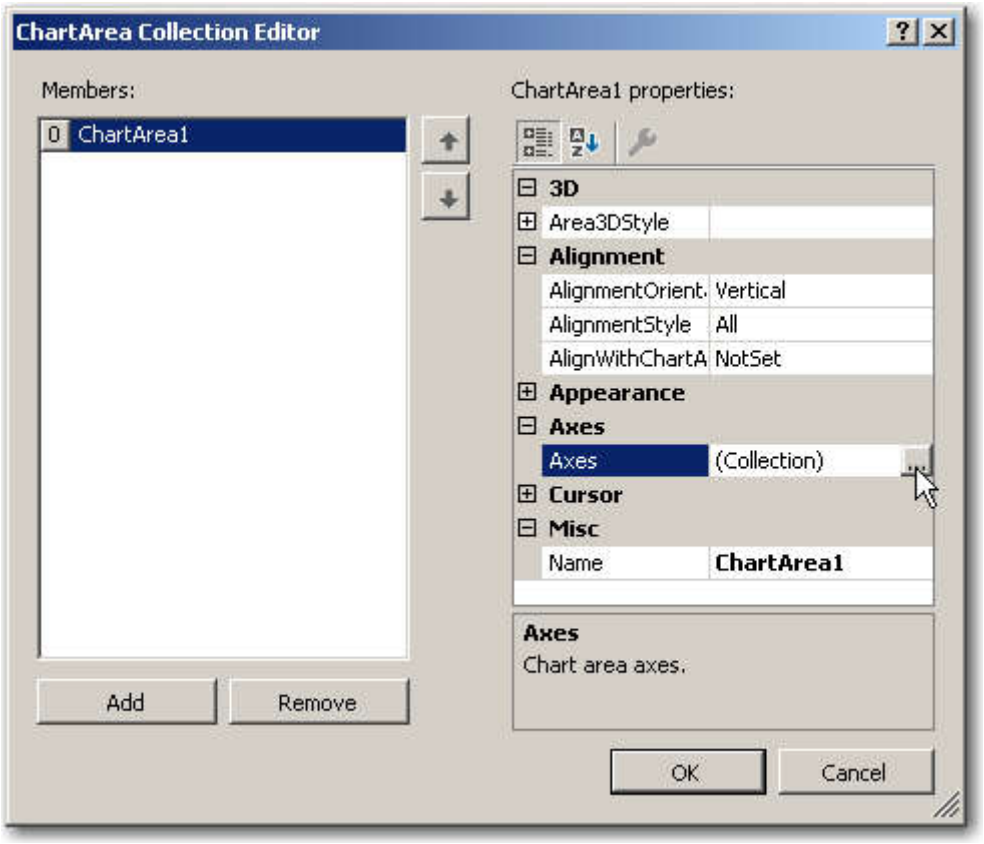
На формі ми спостерігаємо дві віртуальні функціональні залежності (заготовки графіків).

Перейдемо до налаштування осей абсцис та ординат.

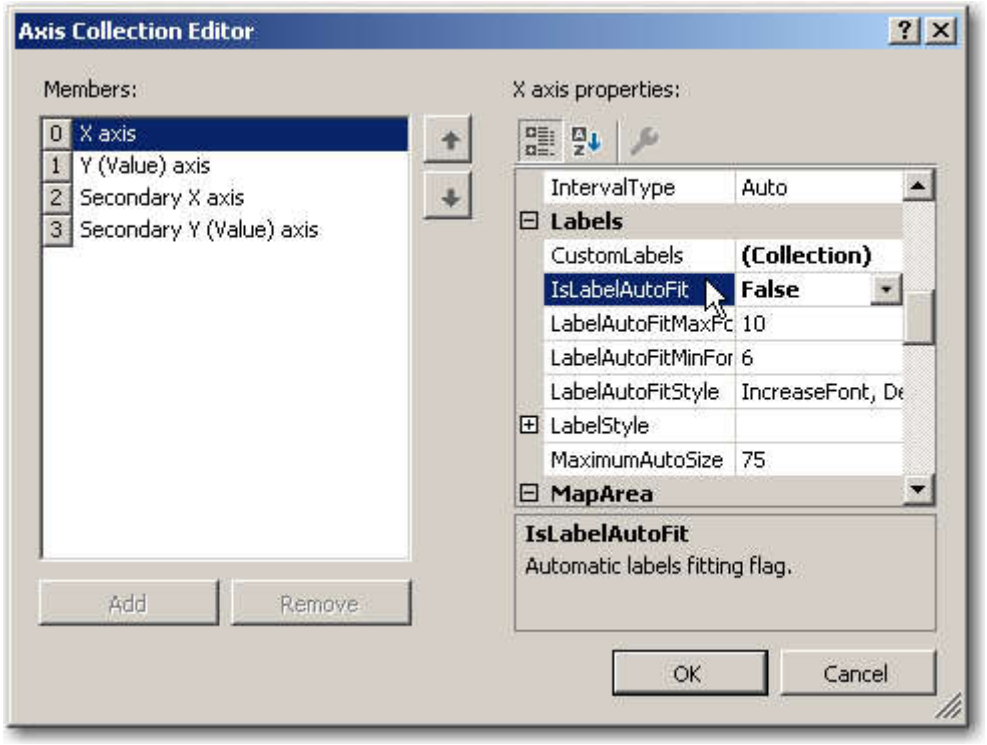
Обираємо наступний пункт редактора властивостей:



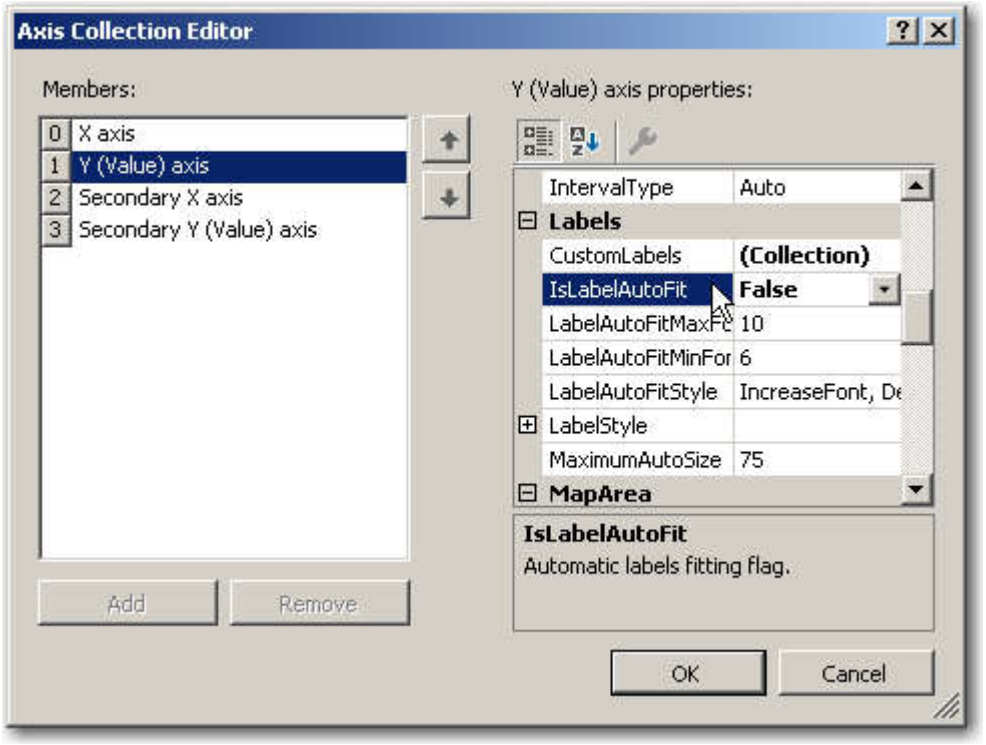
Потім переходимо до його розділу **`Axes`**:



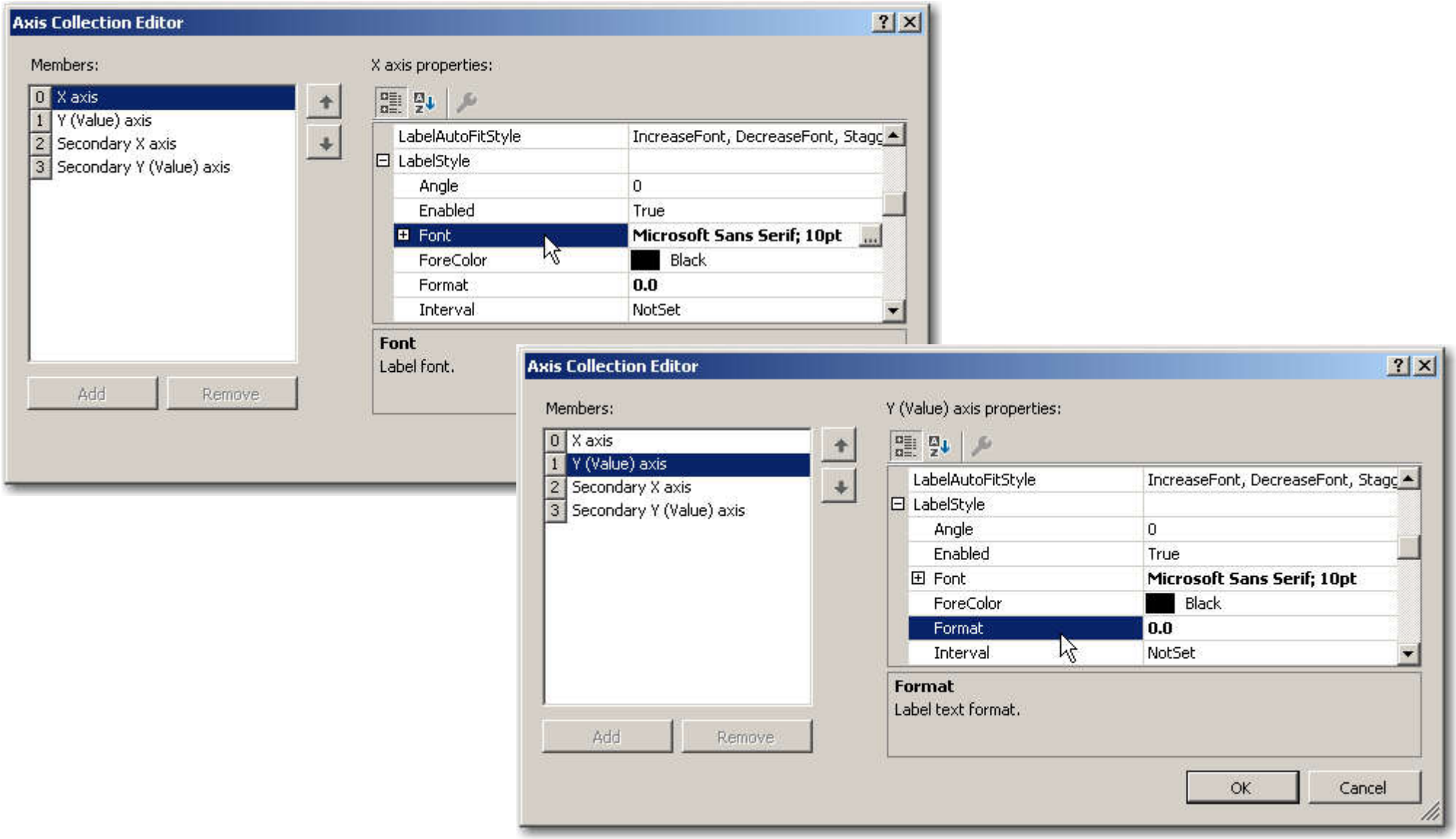
Маємо відкритий новий редактор властивостей **`Axis Collection Editor`**, у якому змінюємо значення властивості **`IsLabelAutoFit`** на **`false`** для основної осі абсцис – **`X axes`**, та основних осей ординат – **`Y (Value) axis`**:



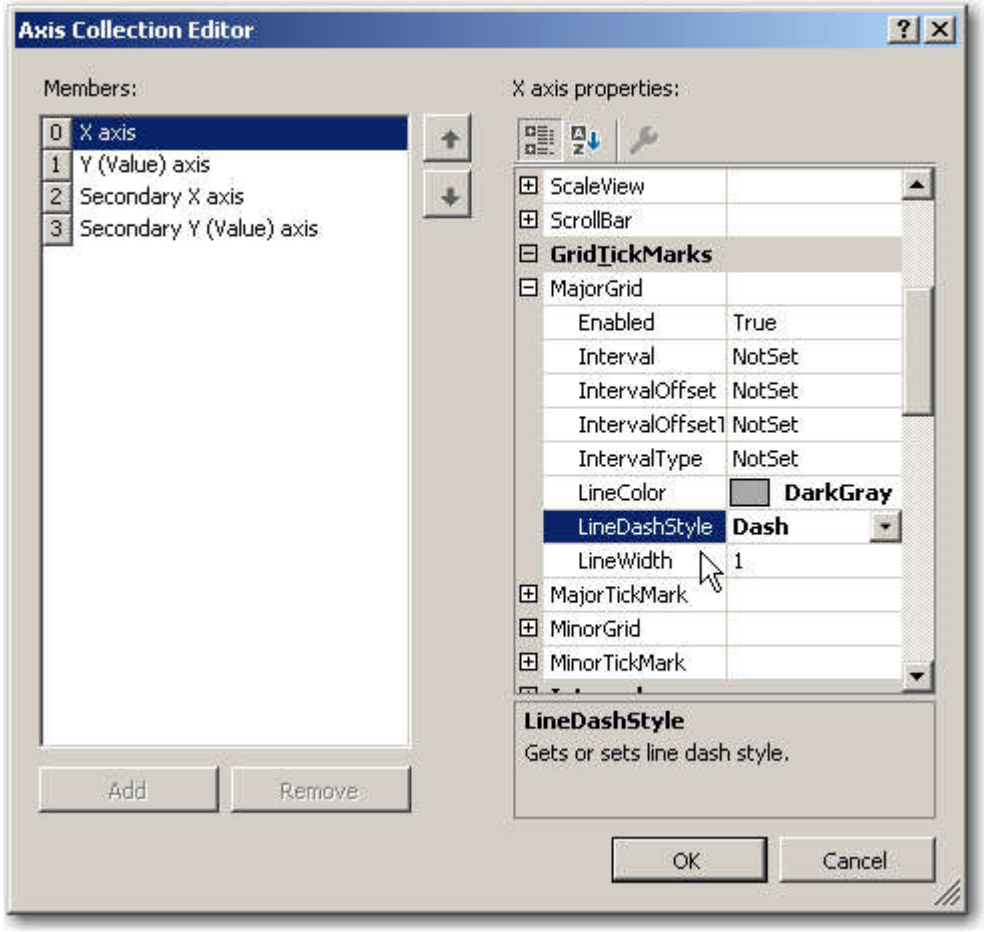
і

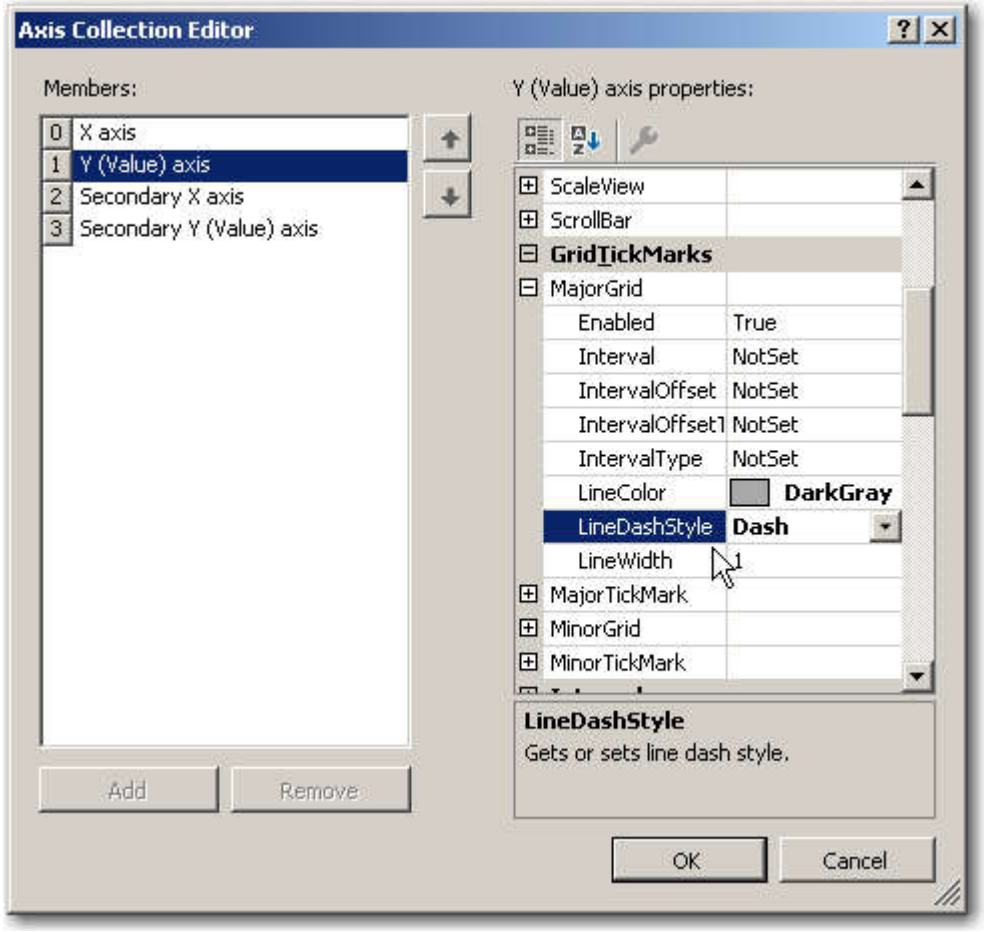


Змінюємо в **`LabelStyle`** формат **`Format`** (і розмір шрифту **`Font`**) відображення шкали на осях абсцис і ординат:

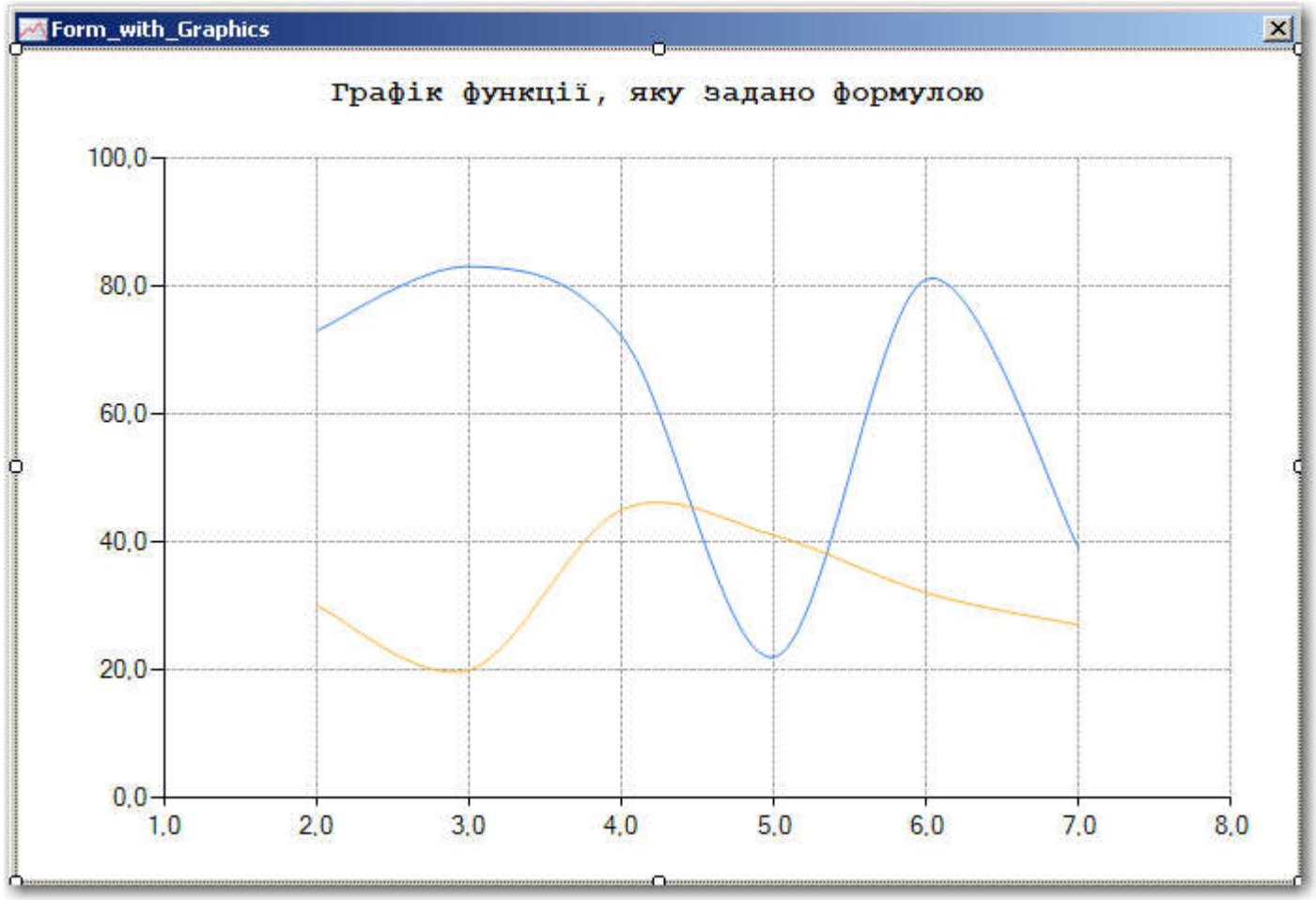


Задаємо стиль **`LineDashStyle`** та колір **`LineColor`** для ліній координатної сітки (нові значення, які ми задали, відображаються тепер жирним шрифтом):

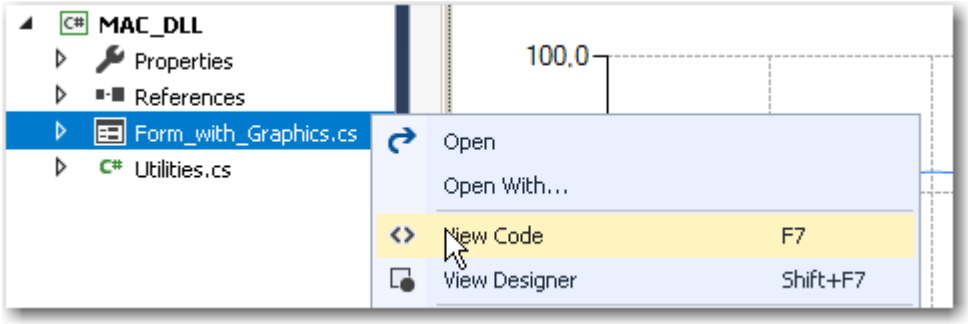




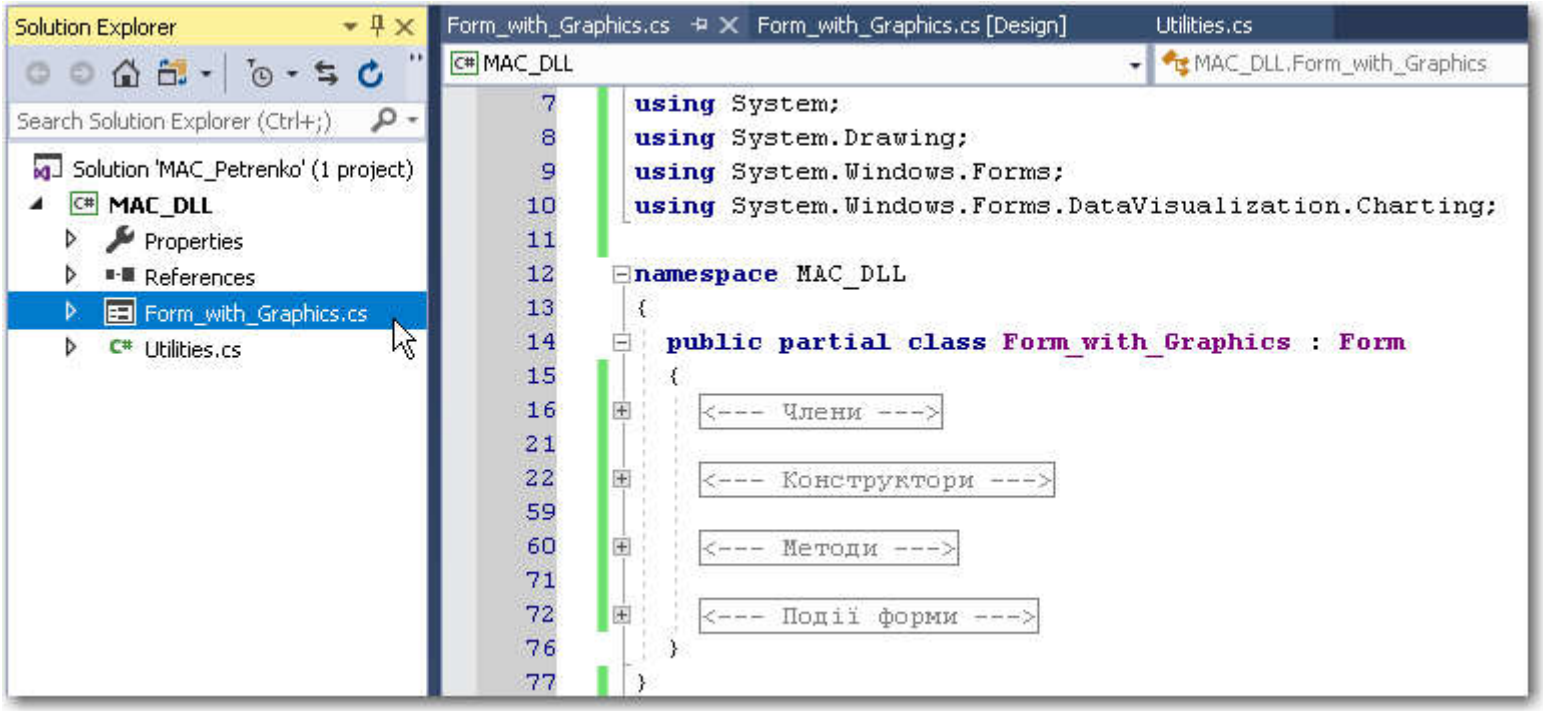
Після всіх виконаних перетворень зовнішній вид компонента **`Form_with_Graphics`** буде наступним:



Переходимо до формування програмного коду класу **`Form_with_Graphics`**:



Загальна структура класу буде наступною:



Перший конструктор (окрім стандартного) буде мати наступний код:

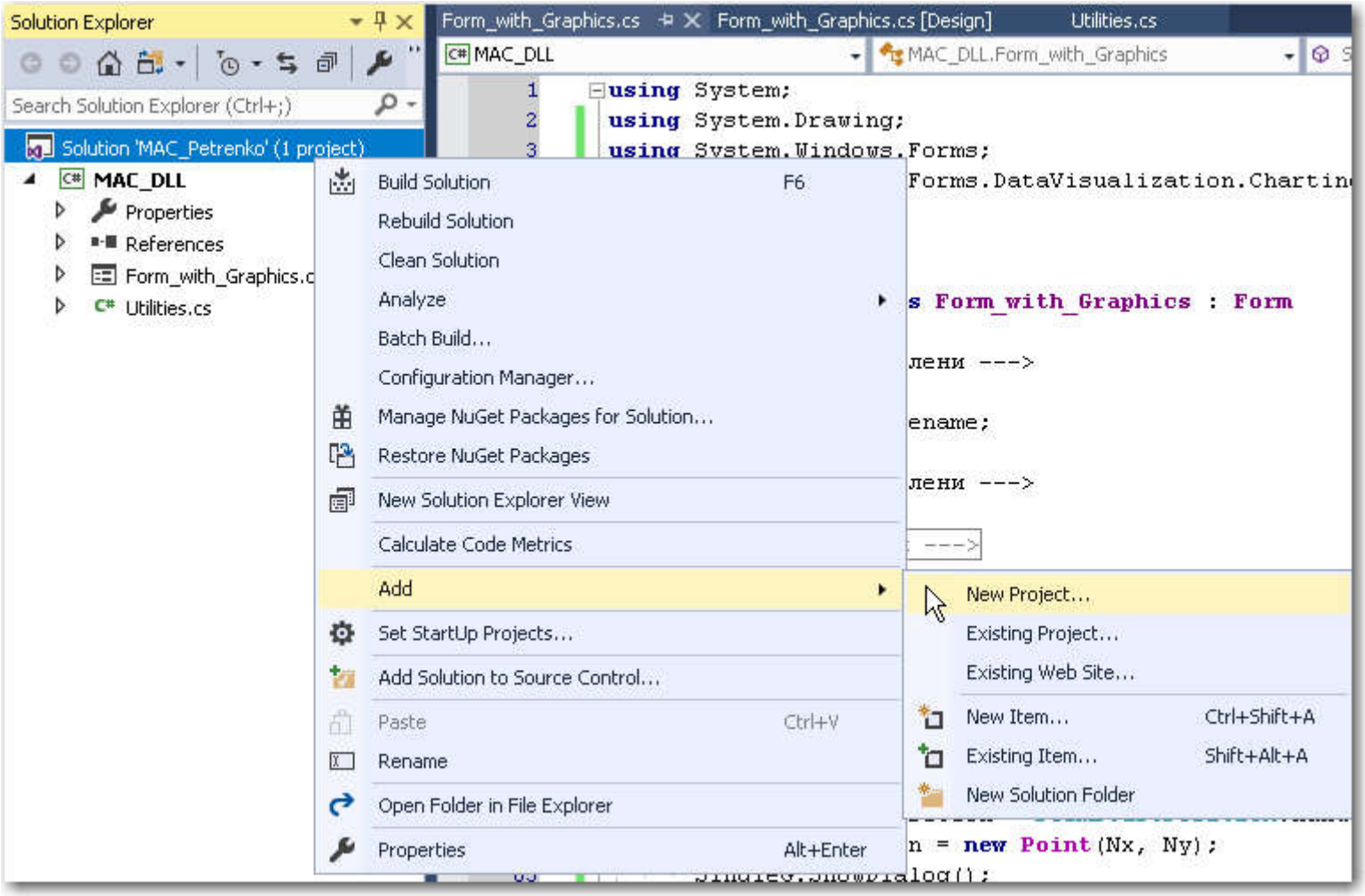
```
#region <--- Конструктори --->

public Form_with_Graphics() { InitializeComponent(); }

private Form_with_Graphics((double x, double f)[] points, string title)
{
  InitializeComponent();
  Chart_with_Graphics.Series[0].Points.Clear();
  Chart_with_Graphics.Series[1].Points.Clear();

  Chart_with_Graphics.Titles[0].Text = title;
  Chart_with_Graphics.ChartAreas[0].AxisX.Interval = 1.0;
  Chart_with_Graphics.ChartAreas[0].AxisY.Interval = 1.0;

  Series S1 = new Series(); int n = points.Length - 1;
  double f_min = double.MaxValue, f_max = double.MinValue;
  foreach ((double x, double f) in points)
  {
    S1.Points.AddXY(x, f);
    f_min = Math.Min(f_min, f); f_max = Math.Max(f_max, f);
  }

  S1.ChartType = SeriesChartType.Spline;
  S1.MarkerStyle = MarkerStyle.None;
  S1.BorderWidth = 3;
  S1.Color = Color.DarkBlue;
  Chart_with_Graphics.Series[0] = S1;

  Chart_with_Graphics.ChartAreas[0].AxisX.Minimum = Math.Floor(points[0].x);
  Chart_with_Graphics.ChartAreas[0].AxisX.Maximum = Math.Ceiling(points[n].x);
  Chart_with_Graphics.ChartAreas[0].AxisY.Minimum = Math.Floor(f_min);
  Chart_with_Graphics.ChartAreas[0].AxisY.Maximum = Math.Ceiling(f_max);

  Chart_with_Graphics.Invalidate();
}

#endregion <--- Конструктори --->
```

Перший метод буде наступним:

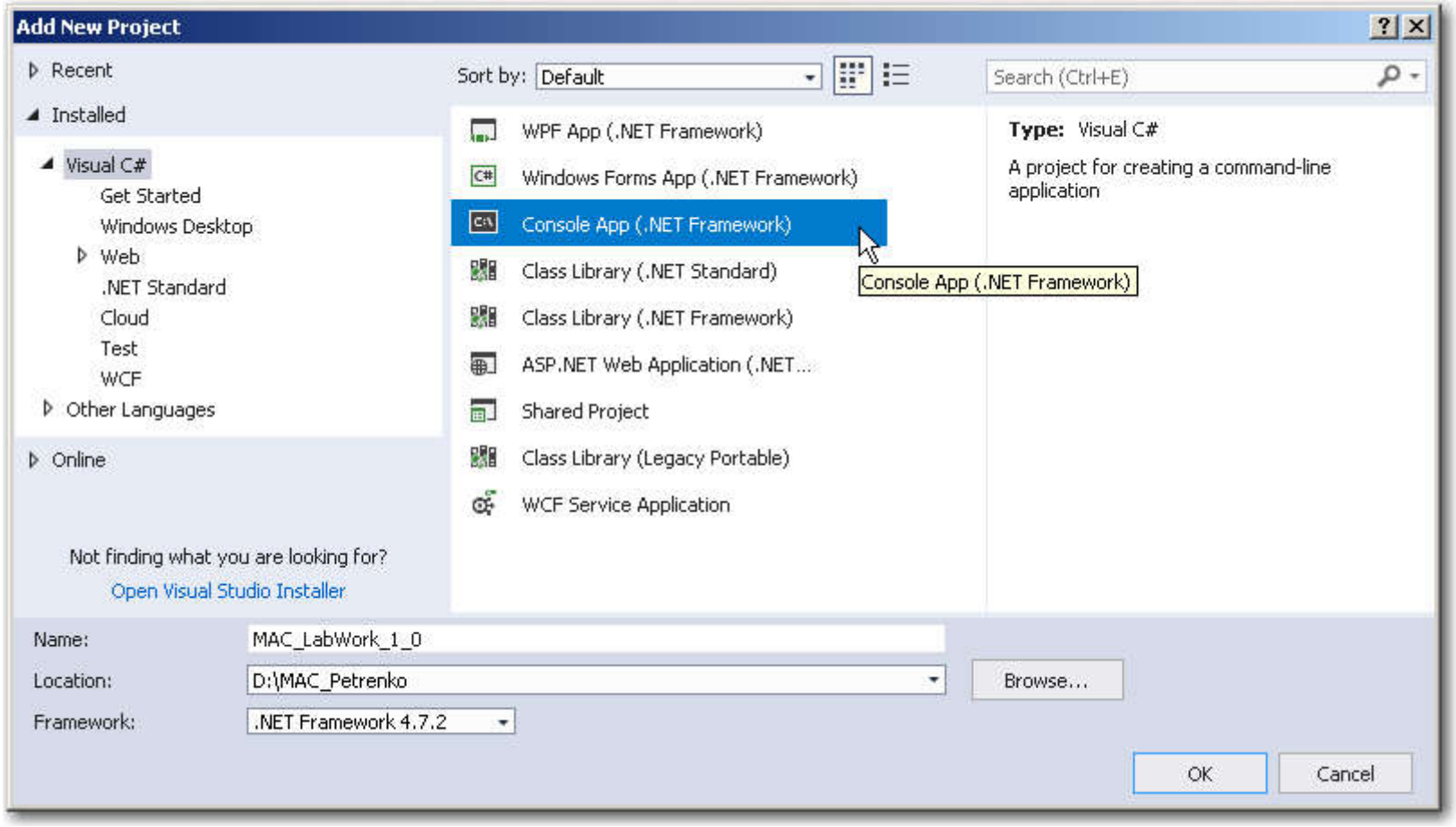
```
1   using System;
2   using System.Drawing;
3   using System.Windows.Forms;
4   using System.Windows.Forms.DataVisualization.Charting;
5
6   namespace MAC_DLL
7   {
8     public partial class Form_with_Graphics : Form
9     {
10      #region    <--- Члени --->
11
12      public string filename;
13
14      #endregion <--- Члени --->
15
16      <--- Конструктори --->
53
54      #region    <--- Методи --->
55
56      public static void SingleGraphic((double, double)[] P, string Title, int Nx, int Ny)
57      {
58        Form_with_Graphics SingleG = new Form_with_Graphics(P, Title)
59        { filename = Title, StartPosition = FormStartPosition.Manual, Location = new Point(Nx, Ny) };
60
61        SingleG.ShowDialog();
62      }
63
64      #endregion <--- Методи --->
65
66      <--- Події форми --->
75    }
76  }
```

Зверніть увагу на додаткові строки коду із директивами **`using`** !

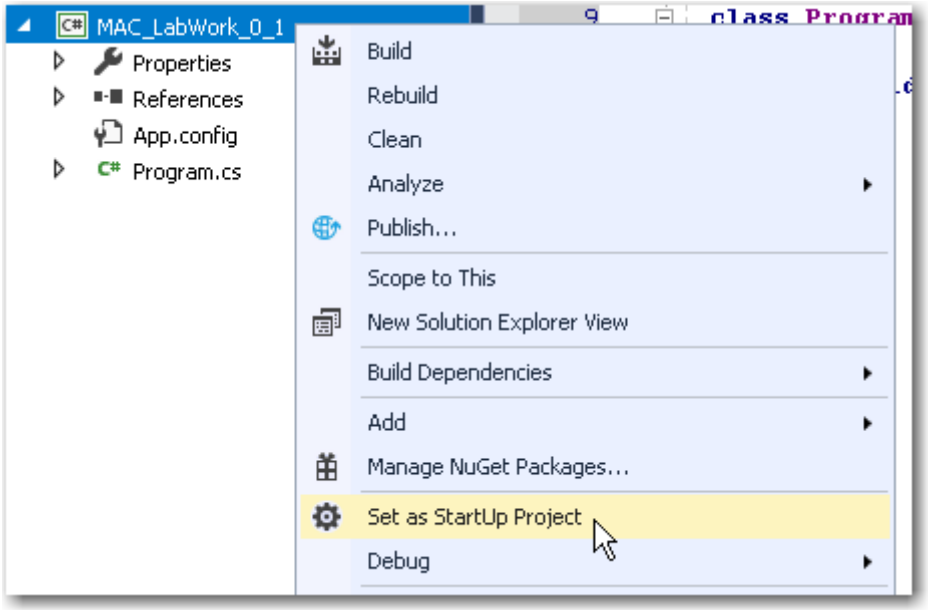
Тепер настав час створення першого ***консольного*** додатка – нового окремого проекту в робочому просторі **MAC_Petrenko – MAC_LabWork_1_0** :

1) 

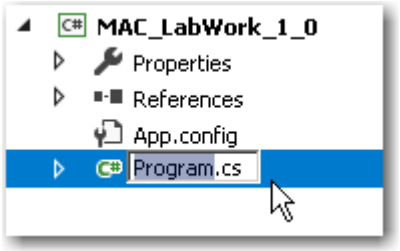
2)


Add New Project
Recent
Installed
Visual C#
Get Started
Windows Desktop
Web
.NET Standard
Cloud
Test
WCF
Other Languages
Online
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
WPF App (.NET Framework)
Windows Forms App (.NET Framework)
Console App (.NET Framework)
Class Library (.NET Standard)
Class Library (.NET Framework)
ASP.NET Web Application (.NET...
Shared Project
Class Library (Legacy Portable)
WCF Service Application
Type: Visual C#
A project for creating a command-line application
Console App (.NET Framework)
Name: MAC_LabWork_1_0
Location: D:\MAC_Petrenko
Framework: .NET Framework 4.7.2
Browse...
OK
Cancel

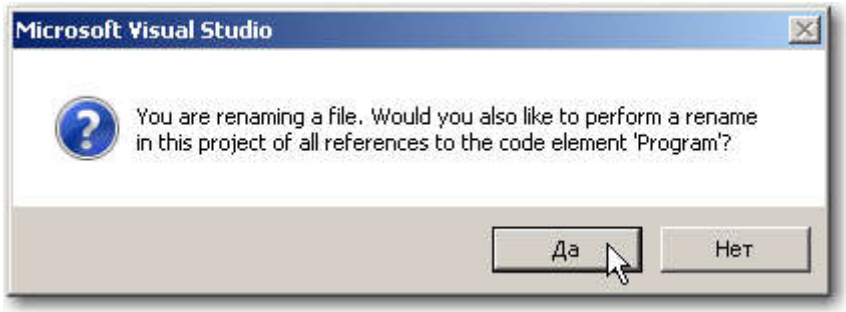
Робимо цей проект «Активним» – опція **«Set As StartUp Project»**:



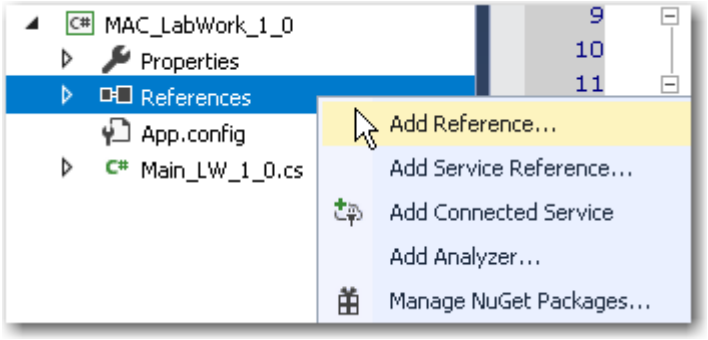
Змінюємо м.'я класу **Program** і файлу **Program.cs** (нове м.'я класу **Main_LW_1_0** і файлу **Main_LW_1_0**):



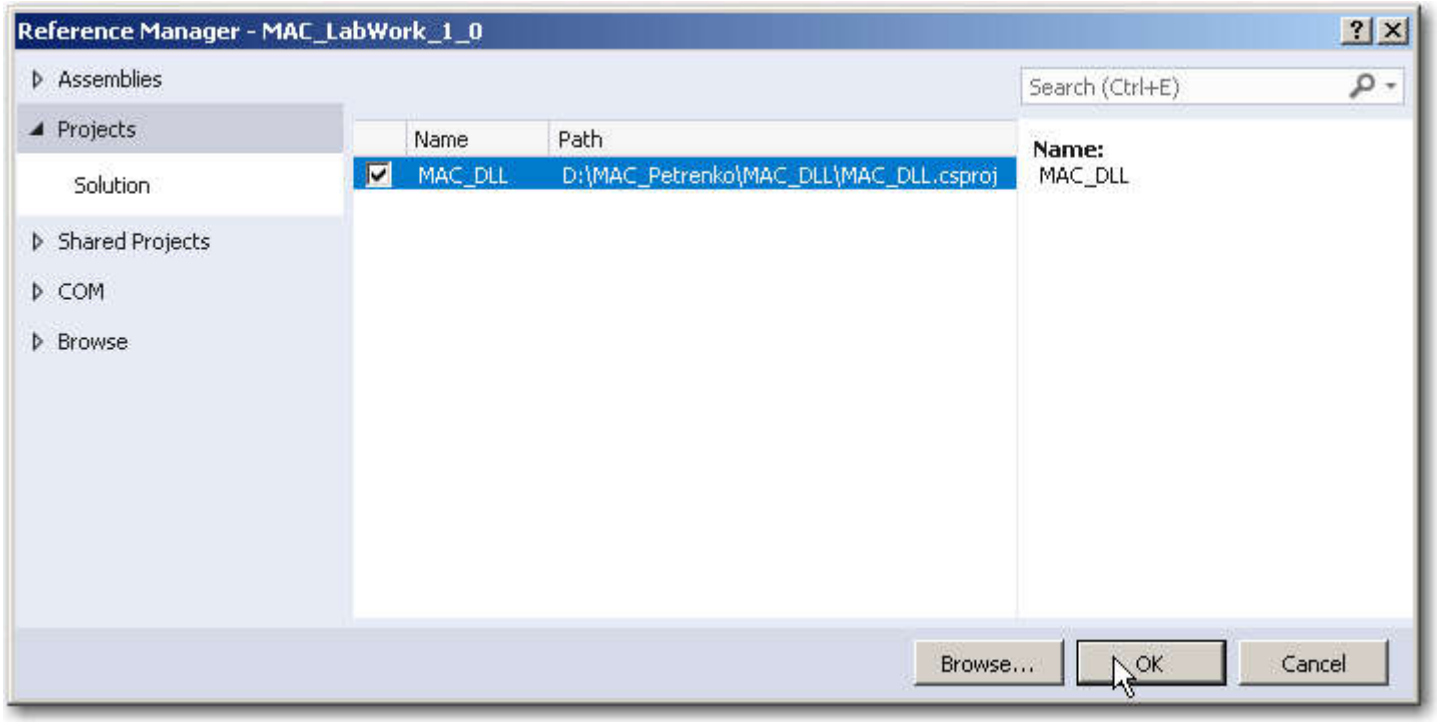
і потім

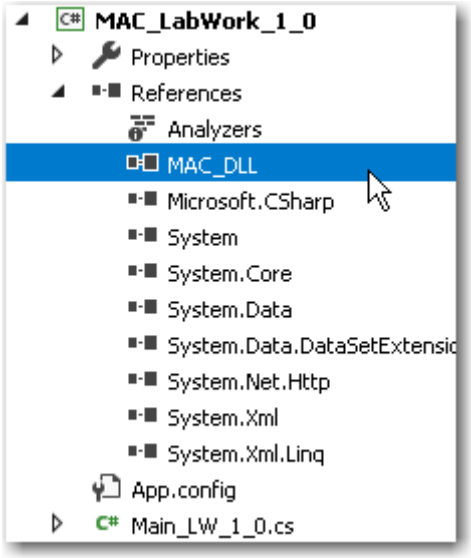


Одразу встановлюємо зв’язки між проектами в розділі посилань **References**:

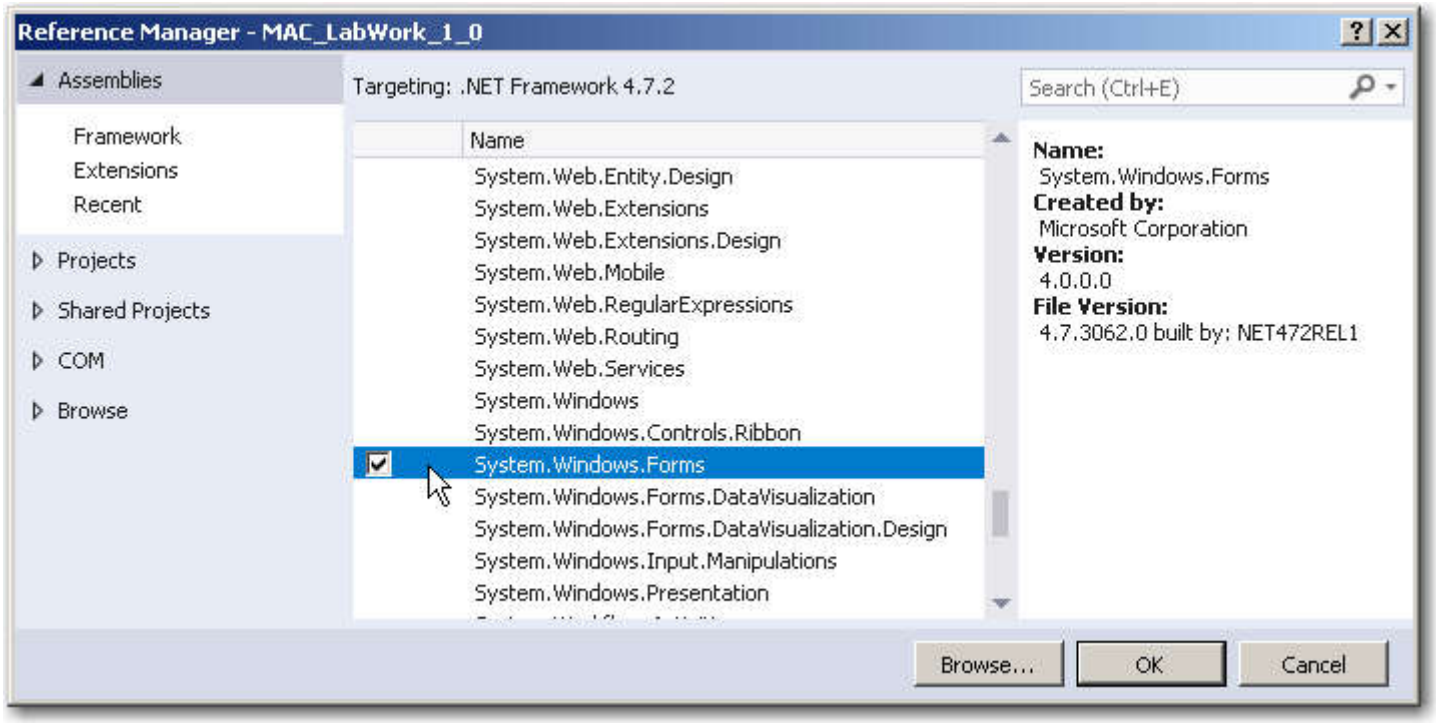


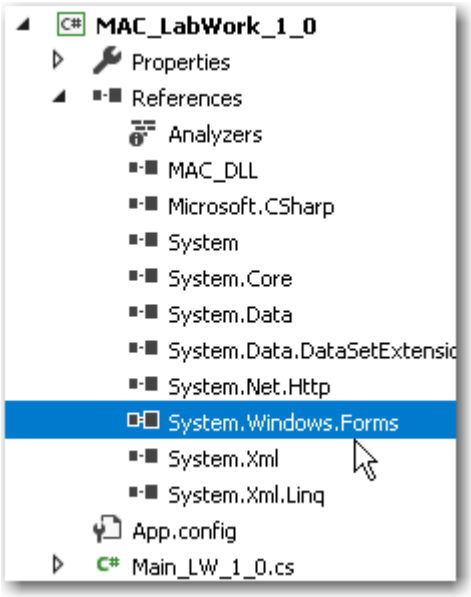
Додаємо посилання на власну бібліотеку **MAC_DLL**:





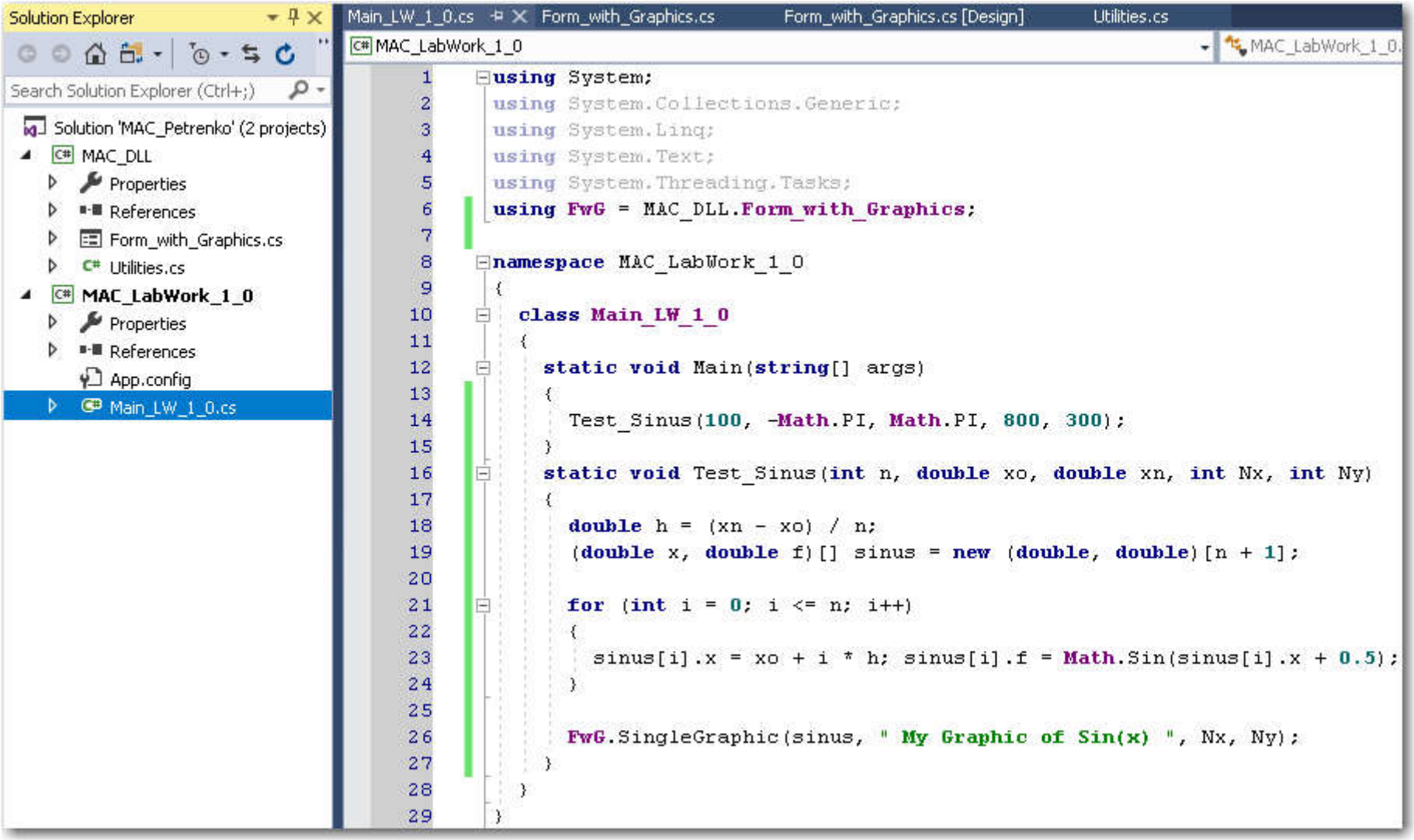
Додаємо посилання на системну бібліотеку **`System.Windows.Forms`**:



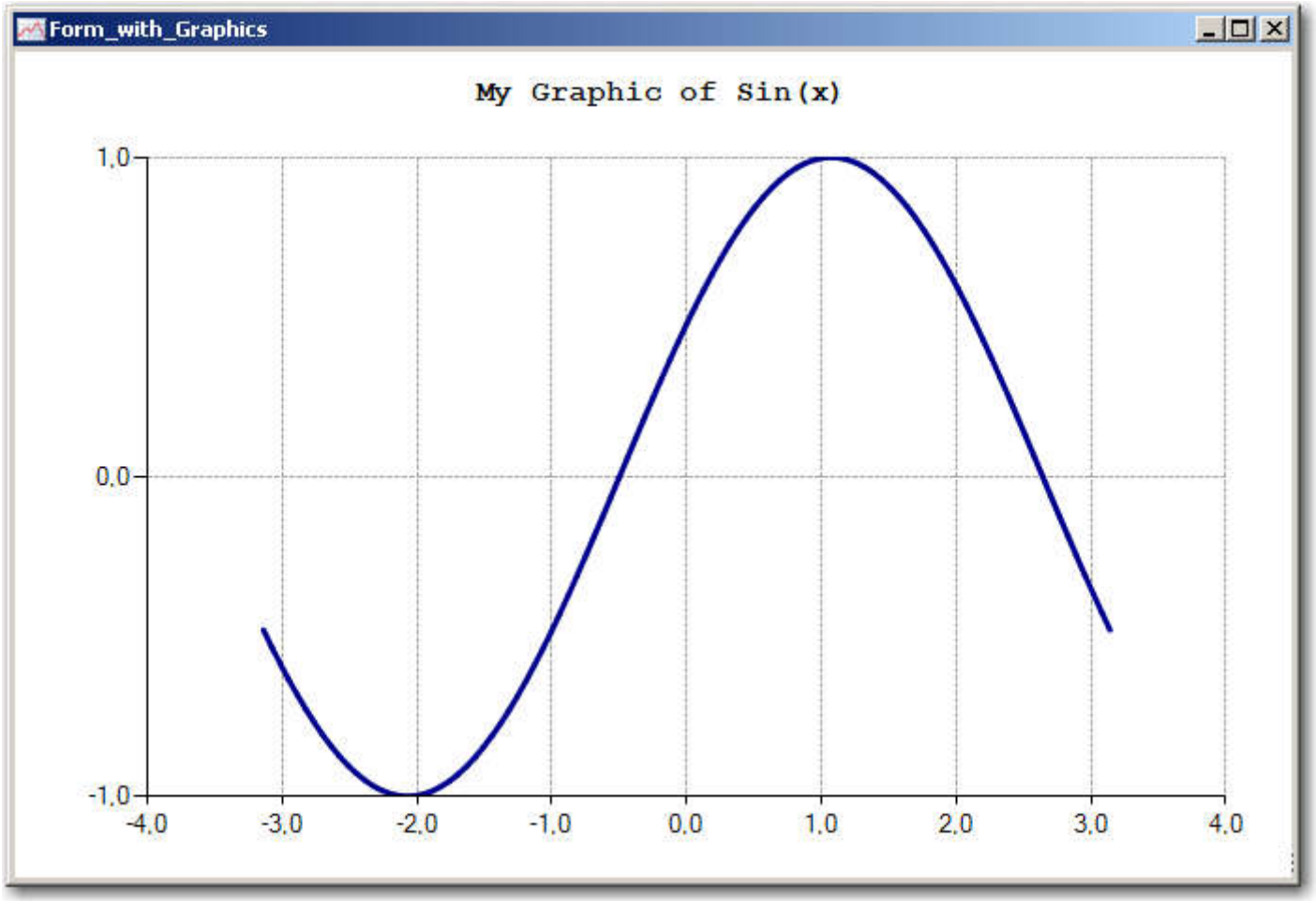


Підготовчу роботу завершено.

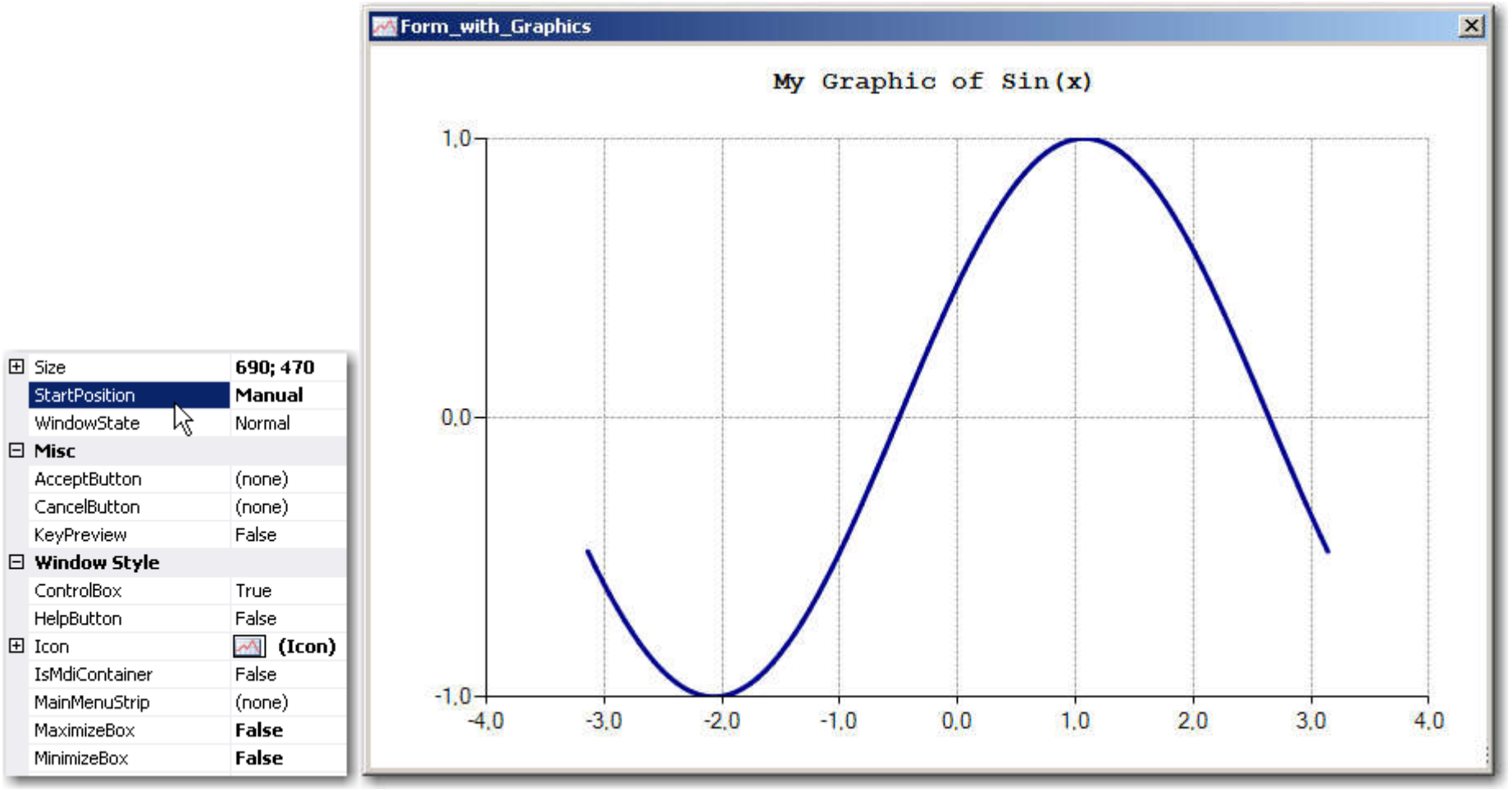
Нарешті реалізуємо код консольного додатка:



Виконання консольного проекту (**`Ctrl+F5`**) дає наступний результат:

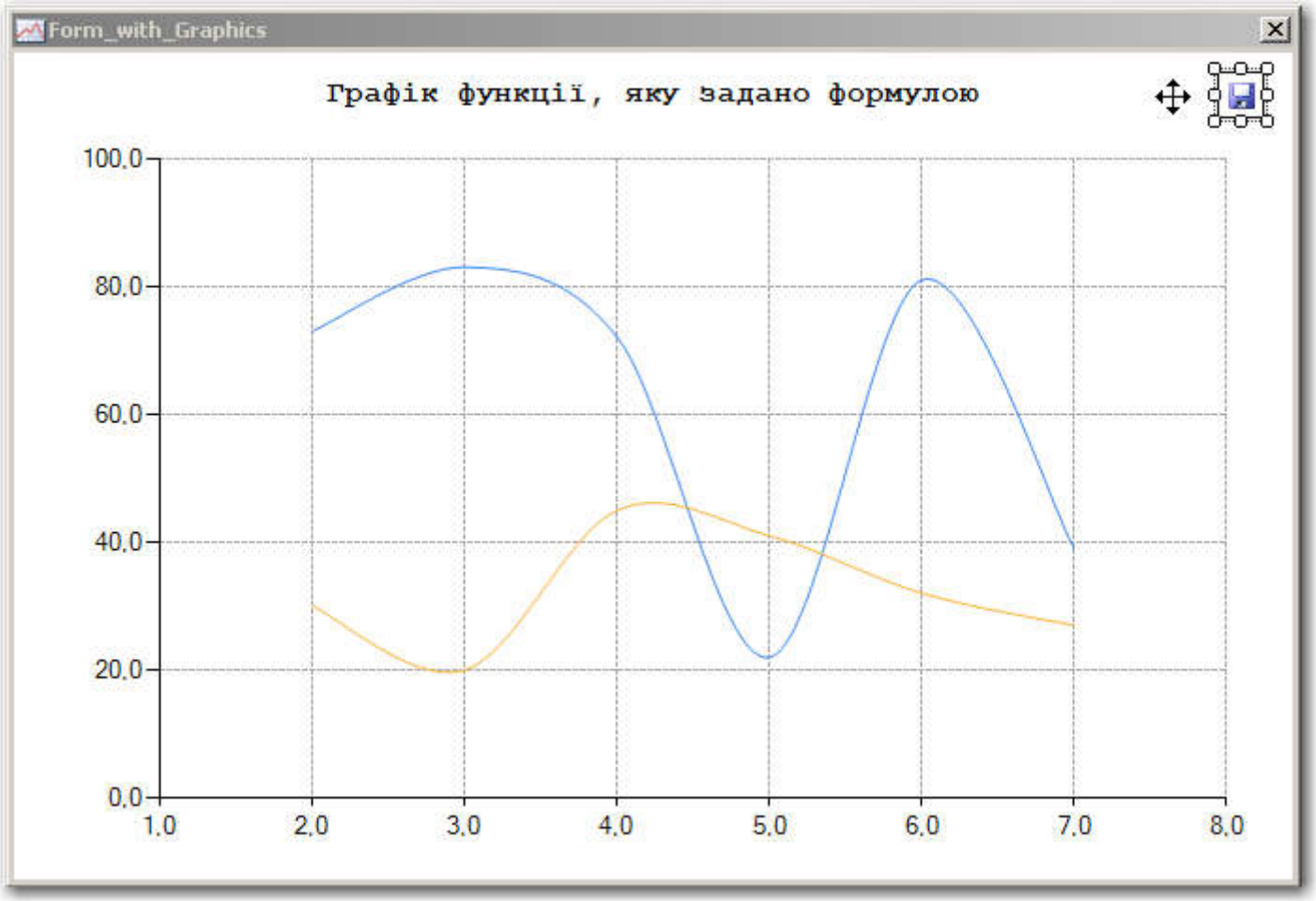


Тепер «приберемо» зайві управляючі елементи на графічній формі (**`MaximizeBox->False, MinimizeBox->False`**), та задамо значення властивості **`StartPosition->Manual`**:

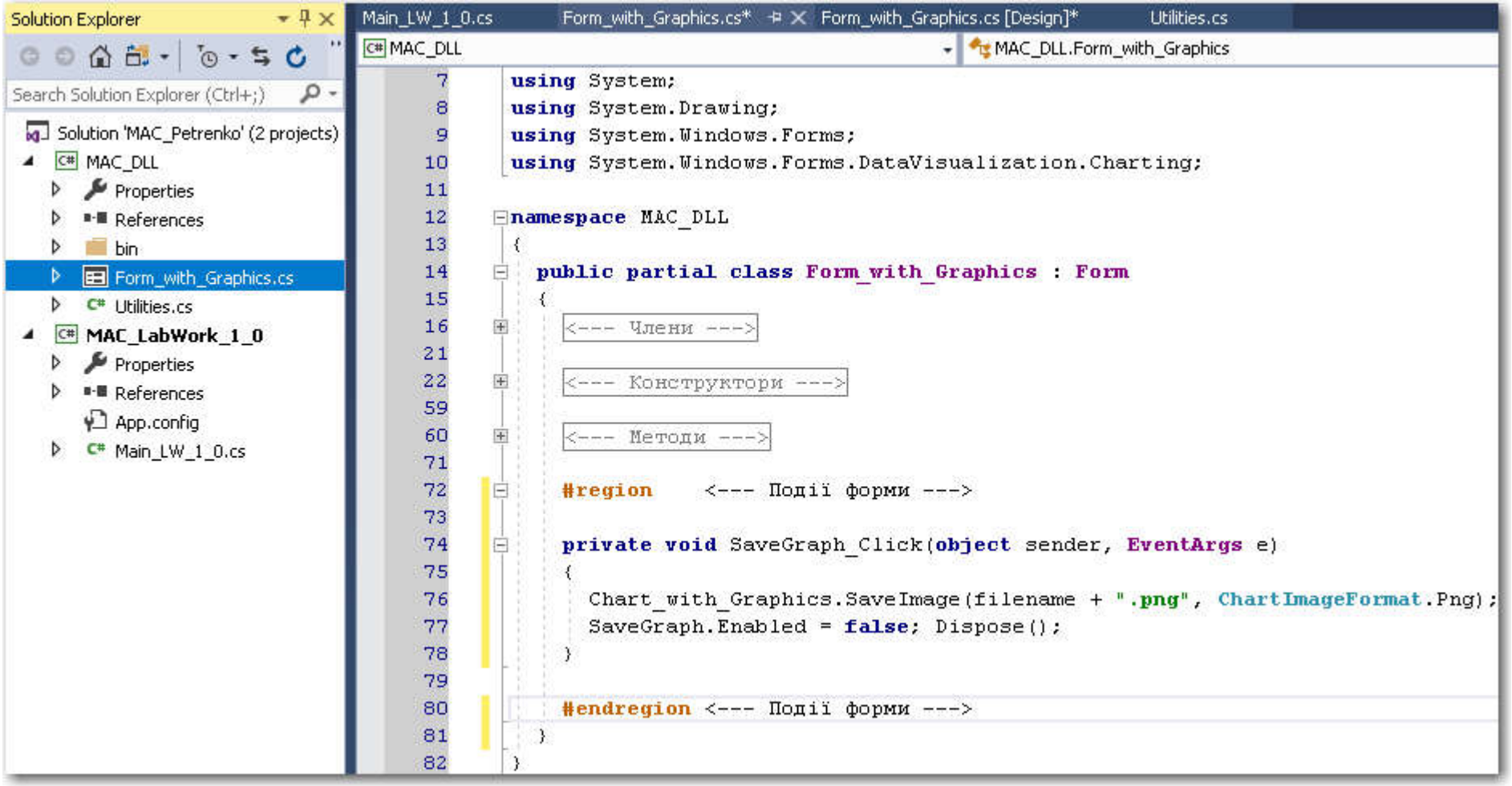


Завершимо нашу роботу додаванням компонента типу **Button** на форму **Form_with_Graphics**, який забезпечить нас графічним файлом із зображенням даної форми, задля того, аби ми мали можливість використовувати графіки функцій у звітах про виконання лабораторних робіт та контрольних завдань. Обираємо м.'я компонента – **SaveGraph**.

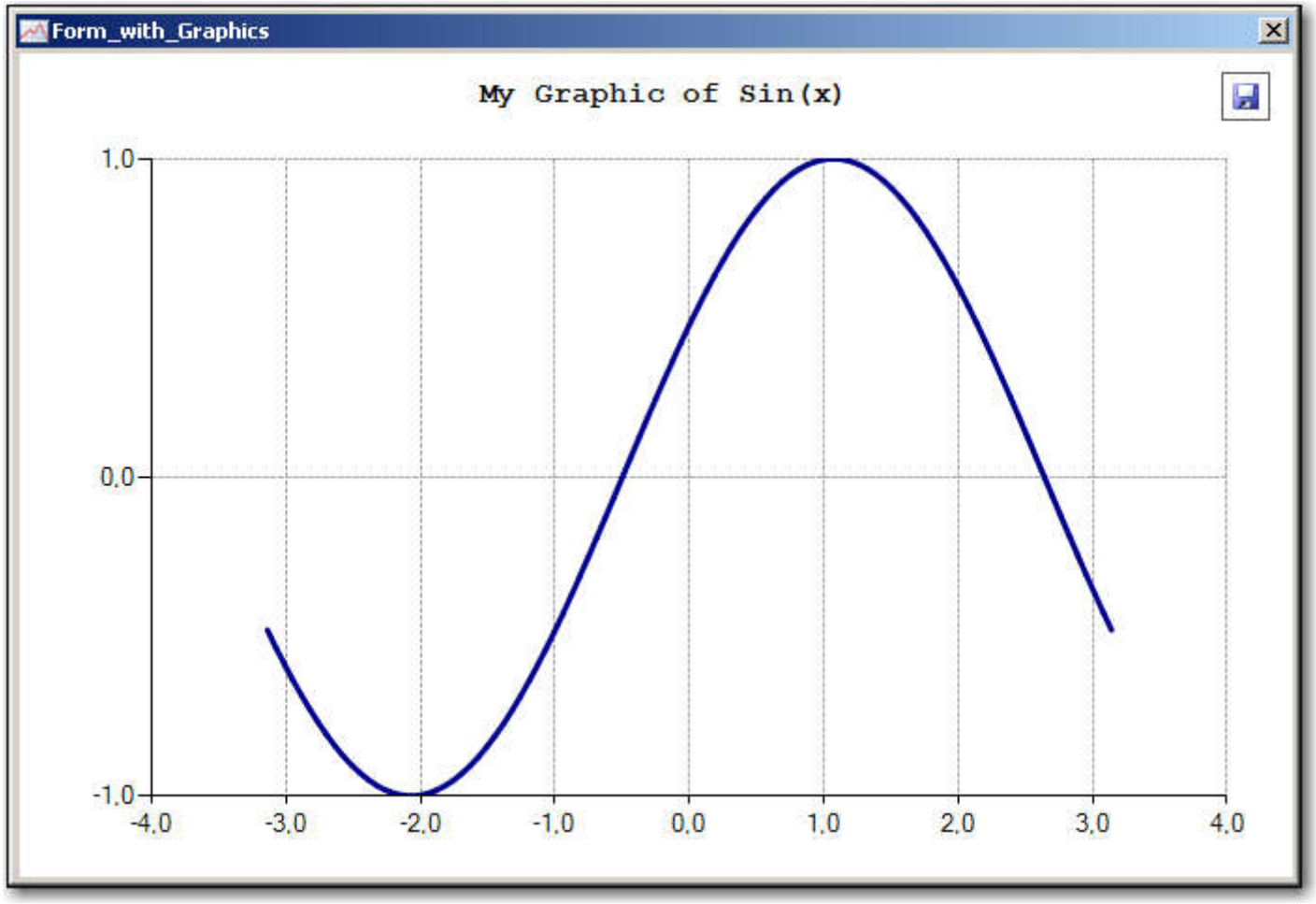
Пропонується наступний вид цього компонента, який ми розташуємо у верхньому правому куті форми **Form_with_Graphics**. Дизайн нового компонента ви можете відредагувати за власним смаком.



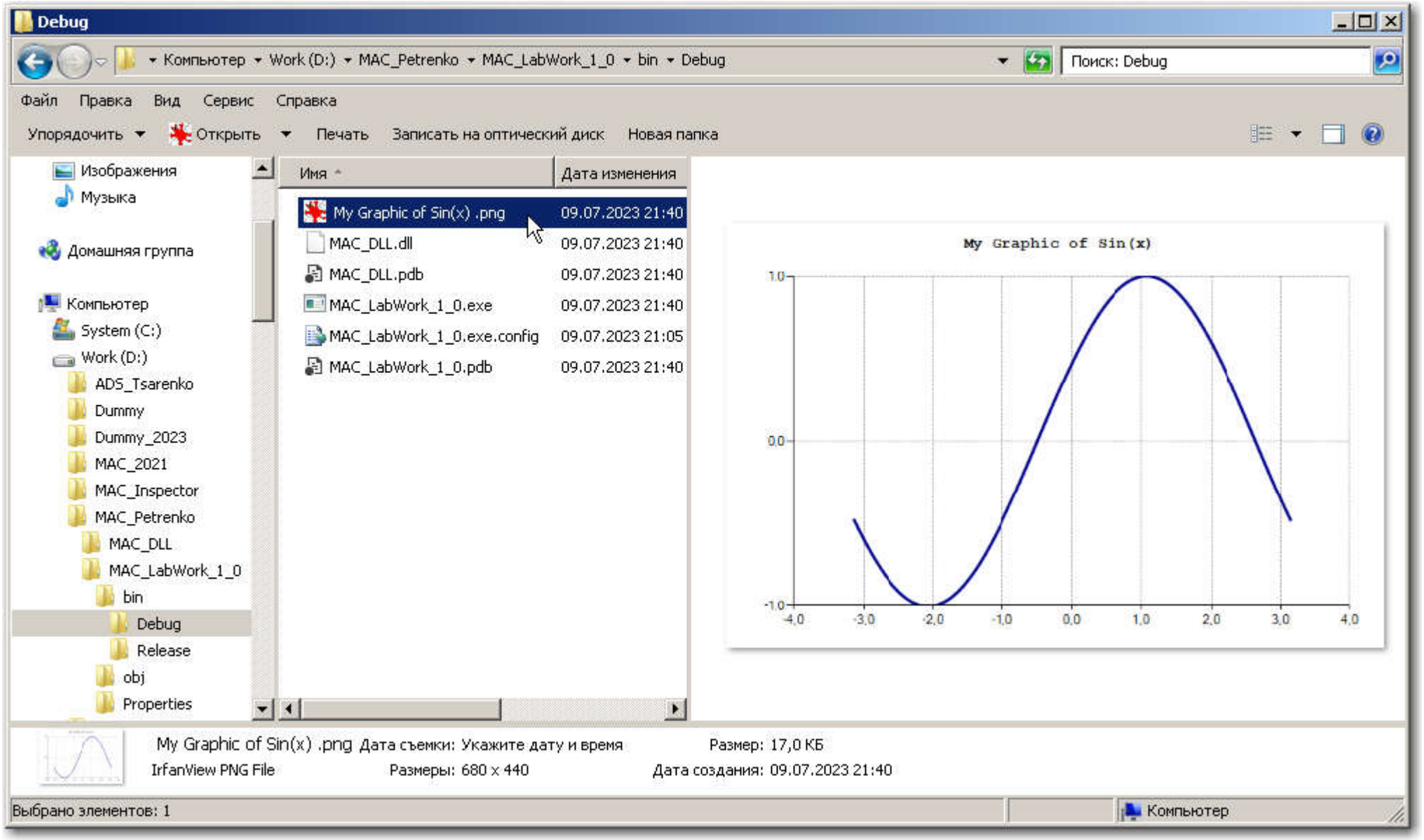
Тепер лише створимо відповідну функцію події **`Click`** для компонента **`SaveGraph`**:



Залишається здійснити тестове випробування нового компонента в дії:



Після натискання на нову кнопку програма завершує свою роботу, а ми відкриваємо в Провіднику директорію проекту **`MAC_LabWork_1_0`** аби знайти новий файл **`"My Graphic of Sin(x) .png"`** та переглянути його.



Кнопка **`SaveGraph`** справно виконує своє «завдання» та зберігає нам графік функції у вигляді файлу із заданим м.’ям.

Отже основну мету першої лабораторної роботи досягнуто.

По-перше, ми маємо робочий простір **MAC_Petrenko**, в якому у подальшому будемо накопичувати нові проекти за даною навчальною дисципліною.

По-друге, ми маємо проект бібліотеки класів **MAC_DLL**, в якій вже є допоміжний компонент **Form_with_Graphics**, із використанням якого ми можемо розглядати або певні результати наших обчислень, або отримувати уяву про функції (аналітичні або дискретні), які ми досліджуємо.

Допоміжний компонент **Form_with_Graphics** і бібліотека класів **MAC_DLL** будуть розширяти свій функціонал у відповідності до майбутніх вимог та завдань.

В якості звіту про виконання даної лабораторної роботи Вам слід надіслати викладачу на перевірку саме графічний файл, який ви отримали та зберегли у Вашому робочому просторі, та м.'я якого **обов'язково** містить Ваше прізвище.

Наприклад, **"Graphic of Sin(x) by PetrenkoIM .png"**.

На основі даної роботи Ви повинні виконати невеличке контрольне завдання, яке теж пов'язане із зображенням графіка певної функції. Методичні вказівки до виконання цього завдання йдуть далі за текстом.

Методика виконання завдання самостійної роботи 1.0

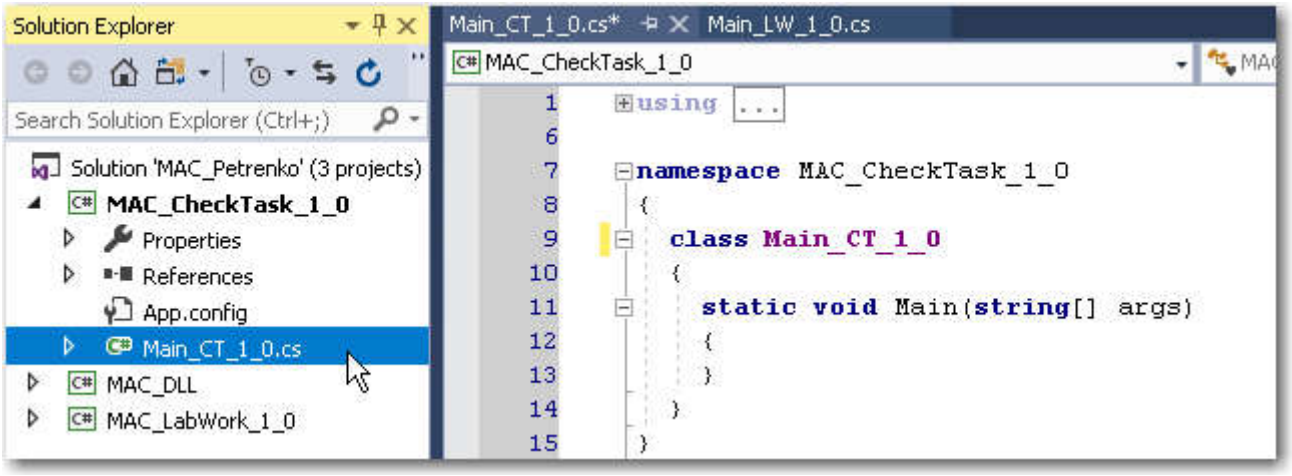
«Побудування графіка бібліотечної функції»

Стартуємо середовище розробки **MS Visual Studio**.

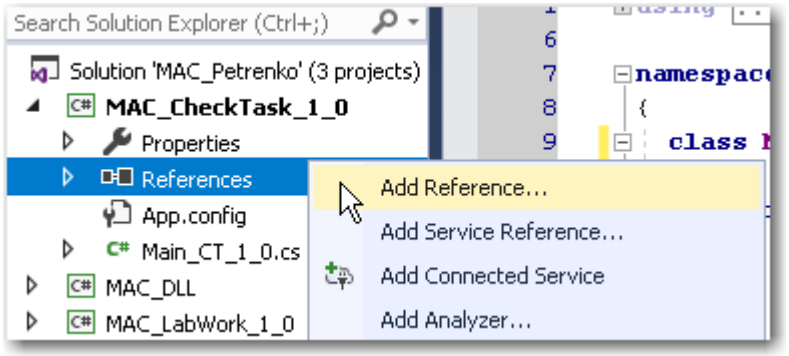
В існуючому робочому просторі **MAC_Petrenko** генеруємо новий проект консольного додатку – **MAC_CheckTask_1_0**.

Обираємо тип конфігурації робочого проекту – **Debug**, і робимо цей проект активним (**Set As StartUp Project**).

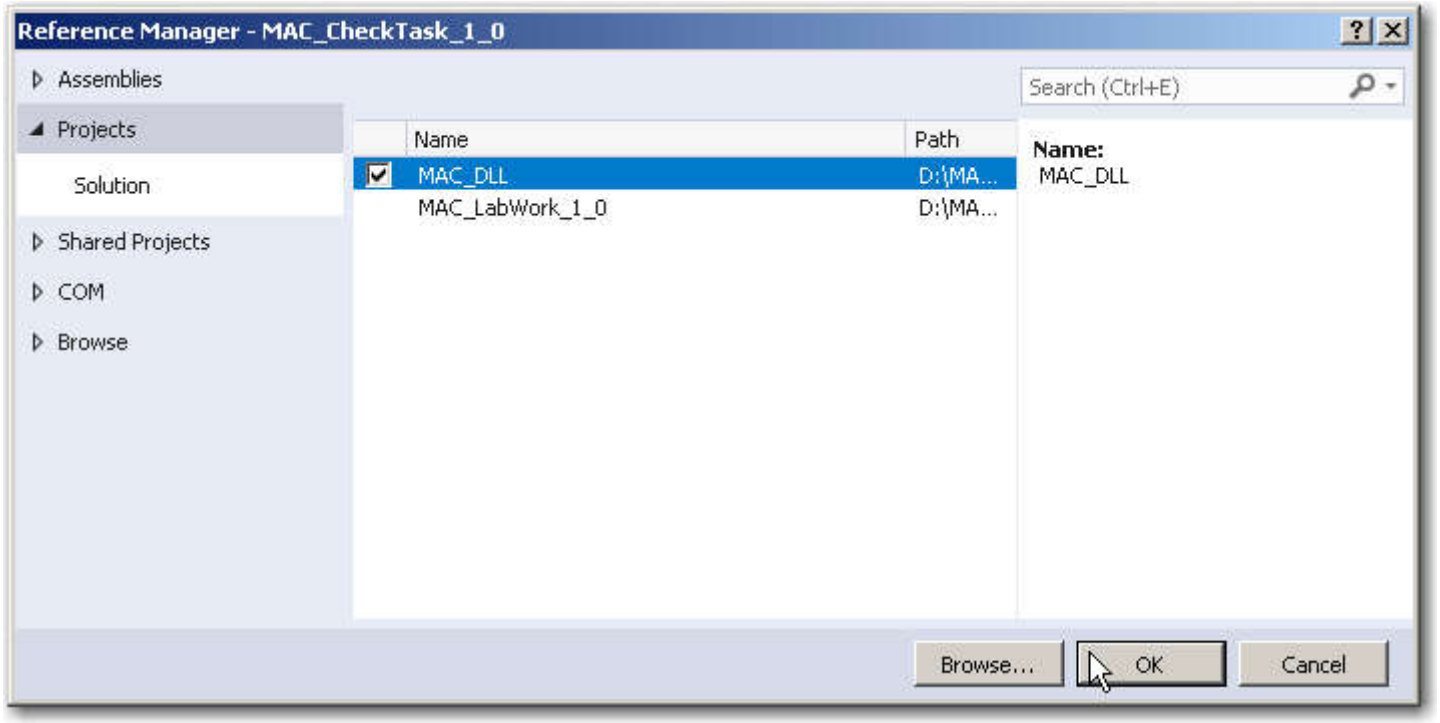
Для зручності одразу перейменуємо клас **Program** в клас **Main_CT_1_0**.



Тепер слід додати бібліотеку класів **MAC_DLL** до переліку посилань **References** проекту **MAC_CheckTask_1_0**:



Це дозволить нам згадувати бібліотеку **MAC_DLL** та її відкриті члени (класи) в директиві **using** у консольному додатку **MAC_CheckTask_1_0**, який розробляється.



Завдання самостійної роботи:

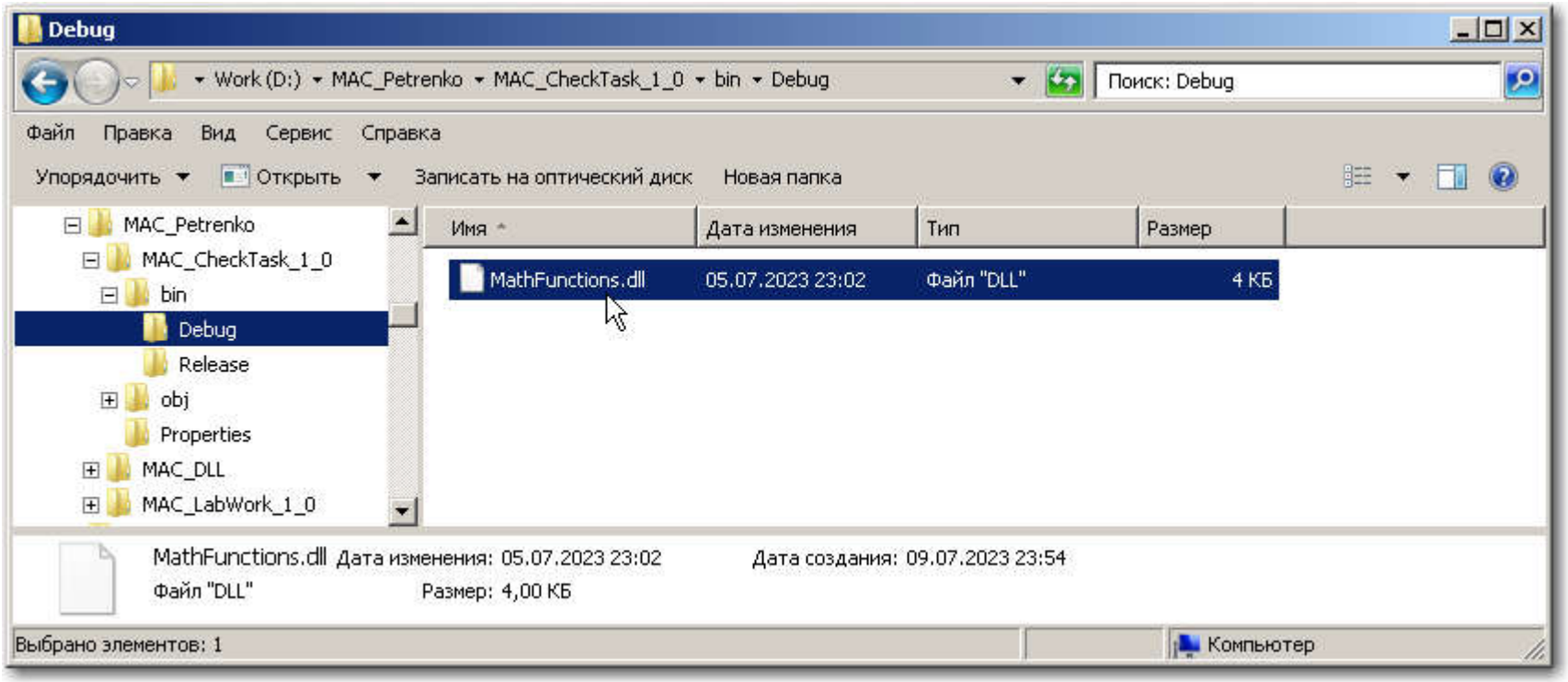
Створити консольний **Windows**–додаток, призначений для побудування графіка функції $f = f(x)$, для якої нам невідоме аналітичне зображення, та яка міститься в формі програмного компонента в заданій бібліотеці класів **MathFunctions.dll**.

Сама бібліотека **MathFunctions.dll** надається вам у відкомпільованому вигляді.

Всі інші параметри завдання за варіантами будуть наведені в кінці даного методичного матеріалу.

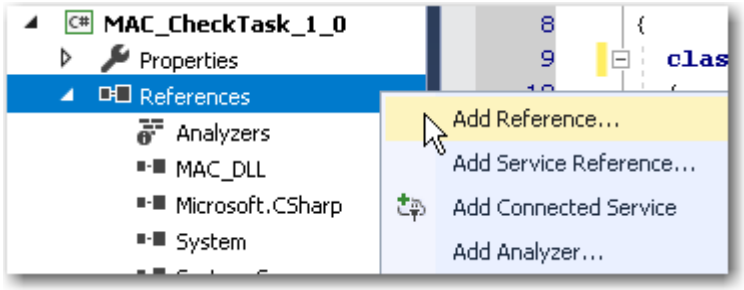
Алгоритм виконання завдання буде продемонстровано на прикладі тестової функції **fx_CT_1_0_v00**.

Крок 1. За допомогою «Провідника» копіюємо та завантажуємо файл **MathFunctions.dll** в директорію **Debug** даного проекту **MAC_CheckTask_1_0** консольного додатку.



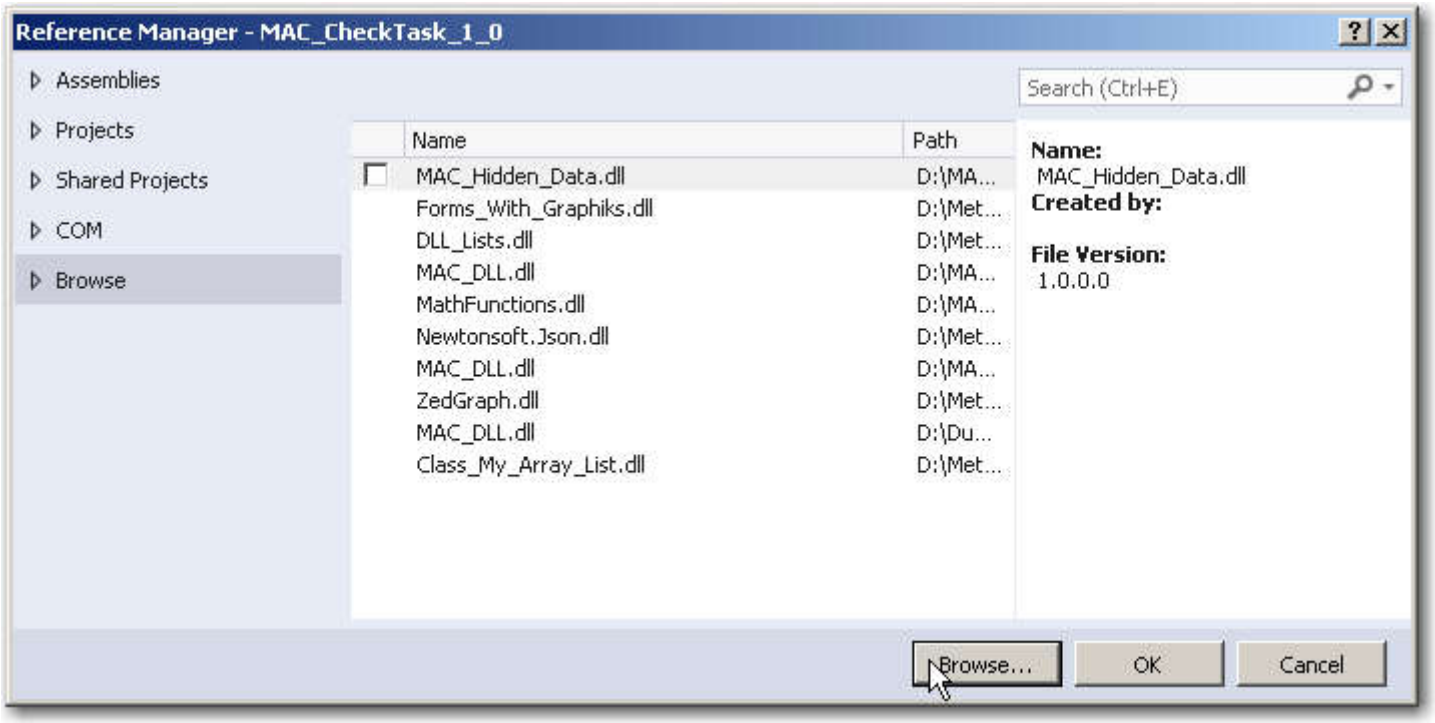
Додаємо цю бібліотеку до переліку **`References`** поточного проекту **`MAC_CheckTask_1_0`**.

1)

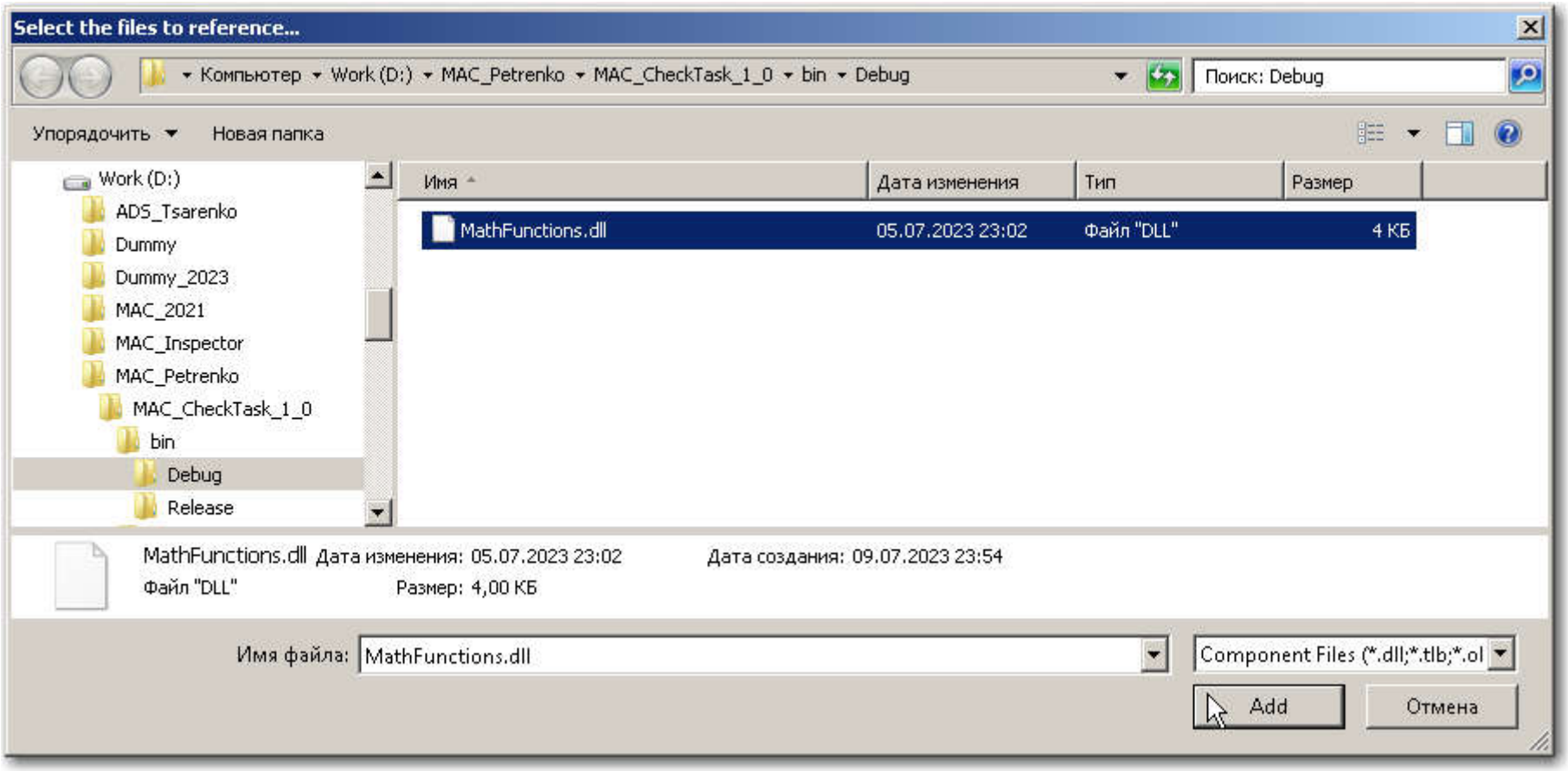


Переходимо на пункт меню «**`Browse`**».

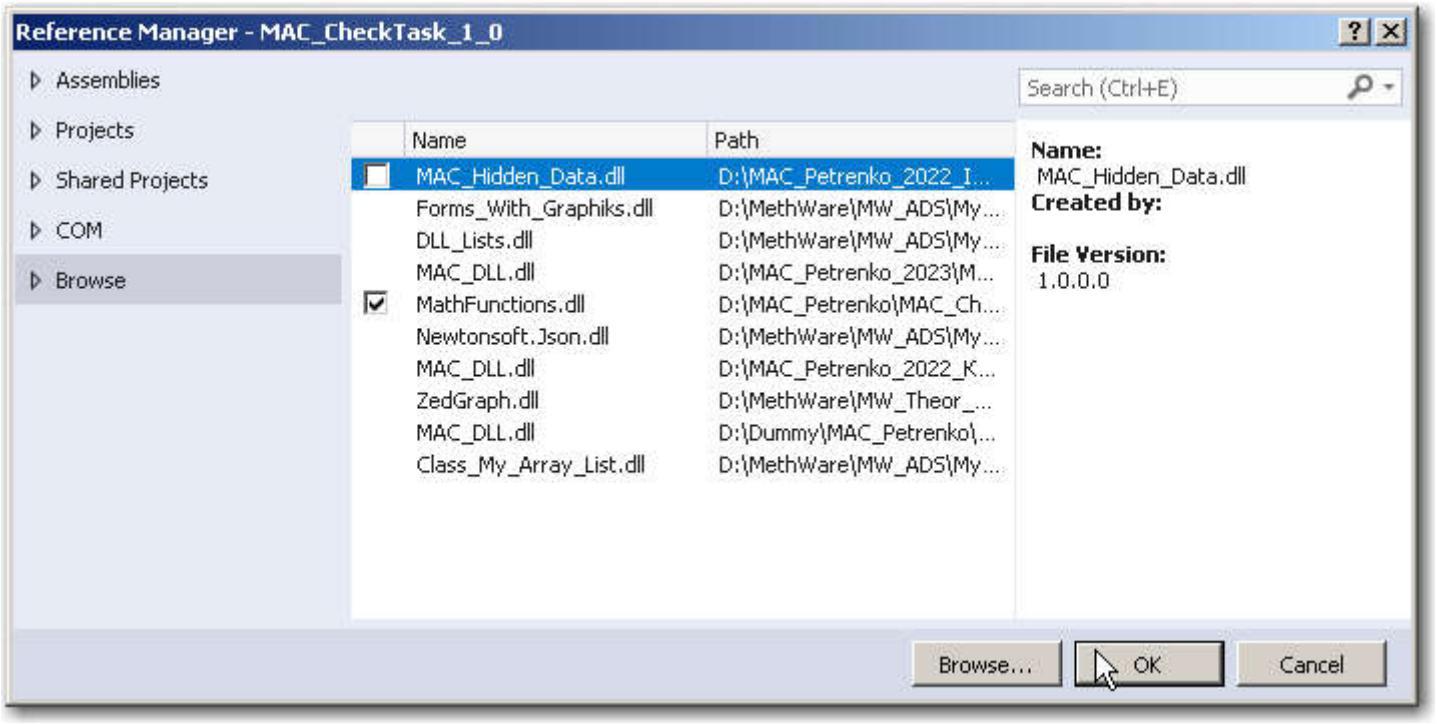
2)



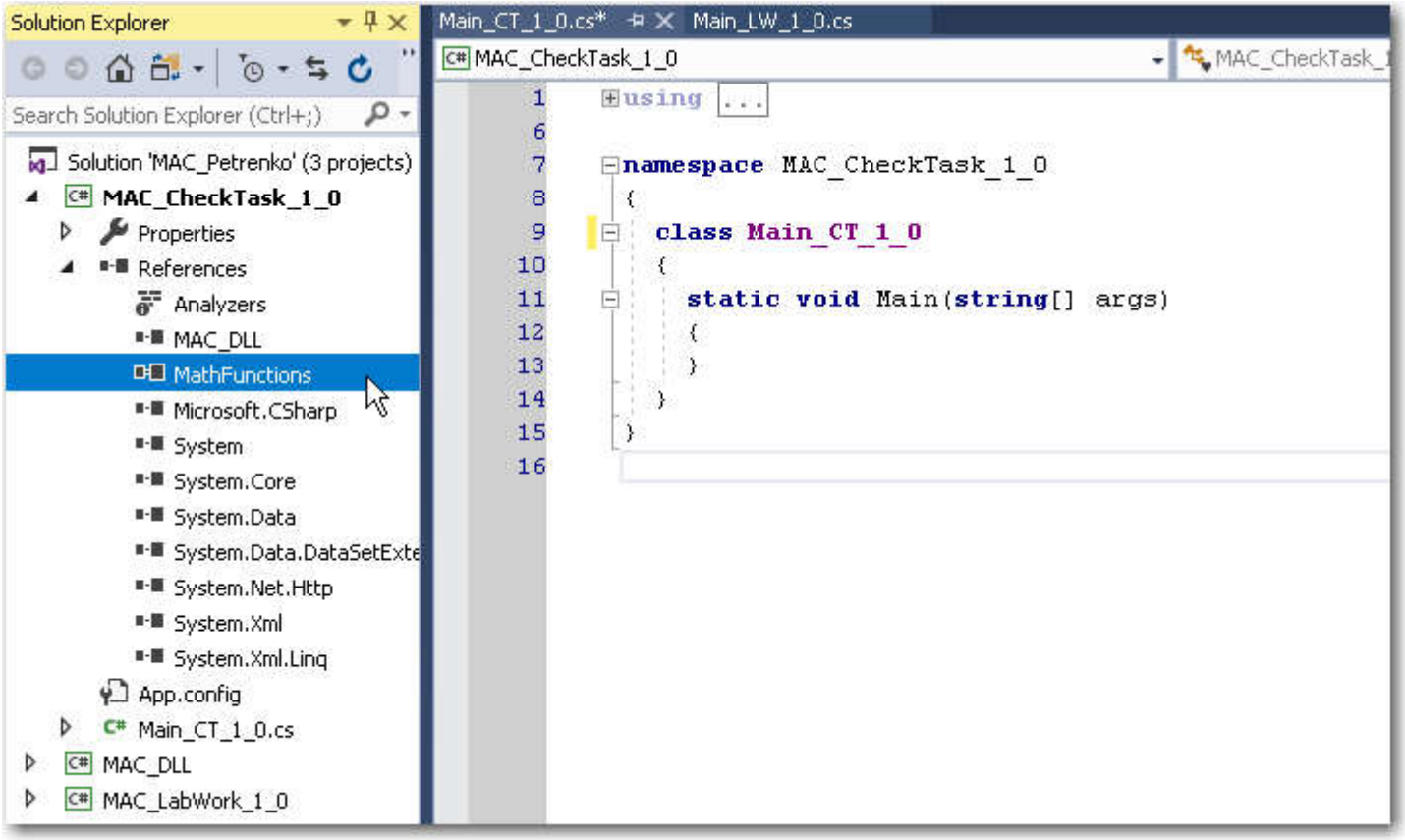
Знаходимо щойно скопійовану бібліотеку **`MathFunctions.dll`**, виділяємо її, після чого тицяємо на кнопку **`Add`**:

3) 

Тицяємо кнопку **Ok**:

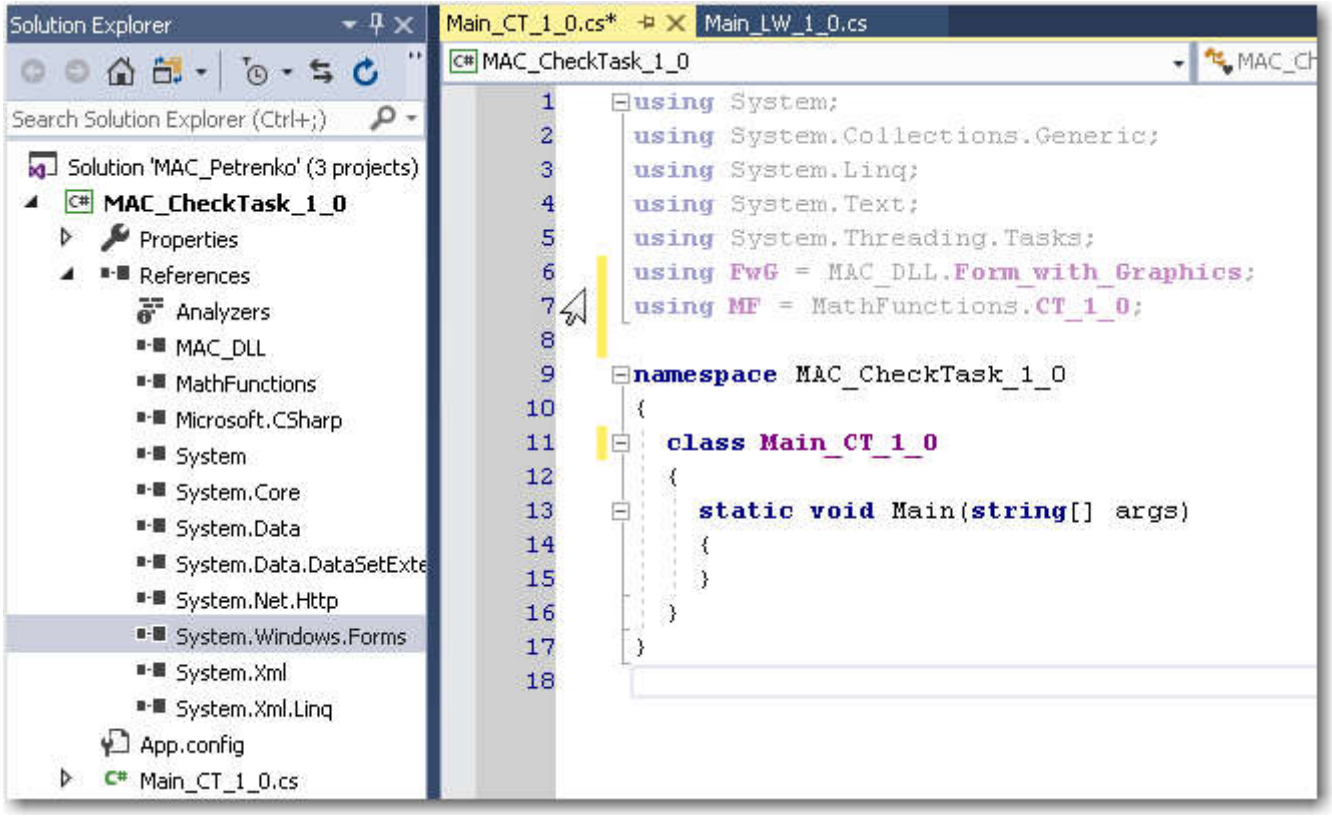
4) 

Отже, потрібна нам бібліотека **`MathFunctions.dll`** вже є приєднаною до проекту **`MAC_CheckTask_1_0`**:



На останок додаємо до переліку посилань **`References`** системну бібліотеку **`System.Windows.Forms`**, саме тим способом, як ми це робили при реалізації проекту **`MAC_LabWork_1_0`**.

Додаємо необхідні директиви **`using`** в код проекту:

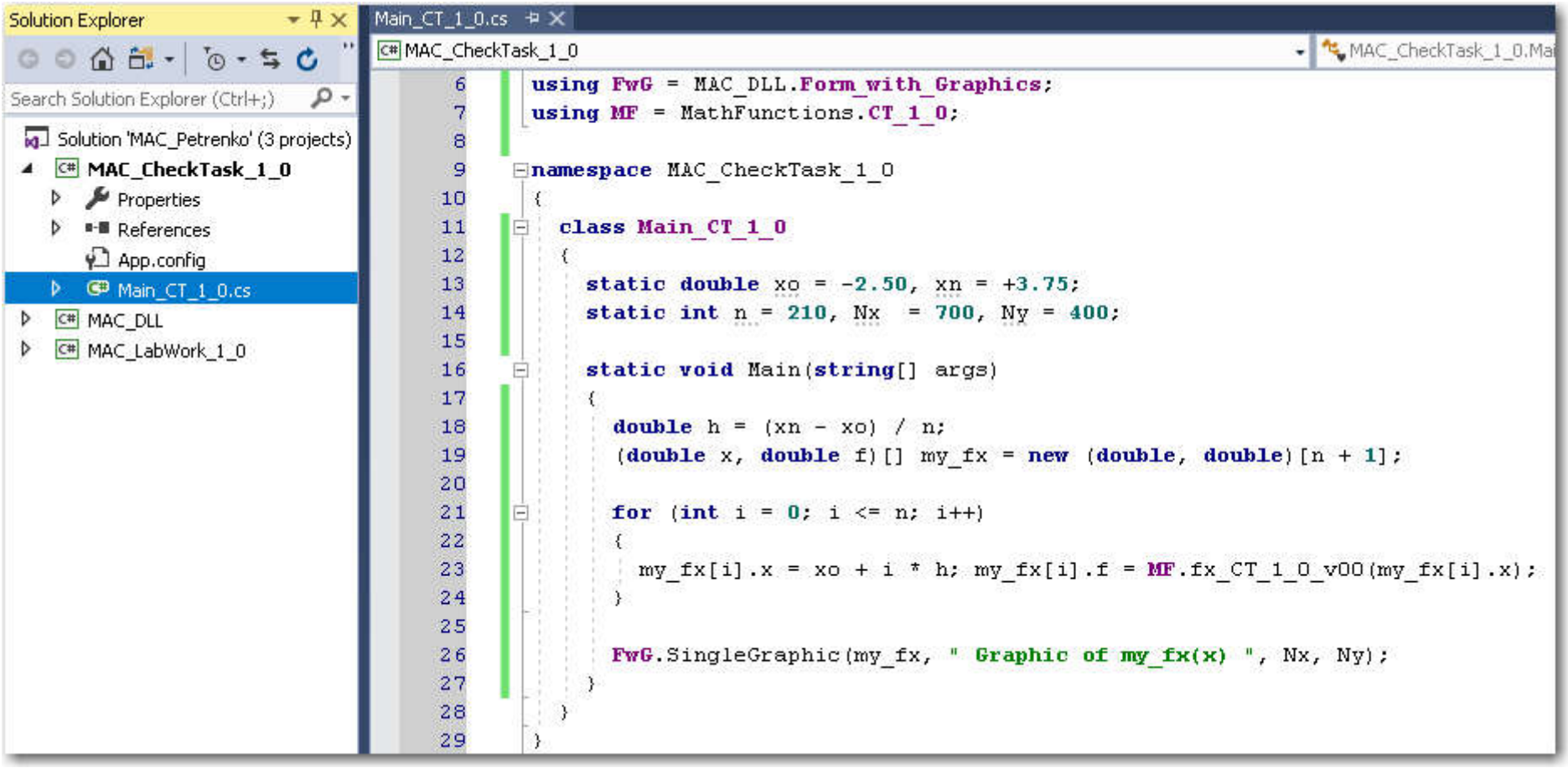


Нехай ми маємо наступну таблицю параметрів, які слід використовувати для побудування графіку.

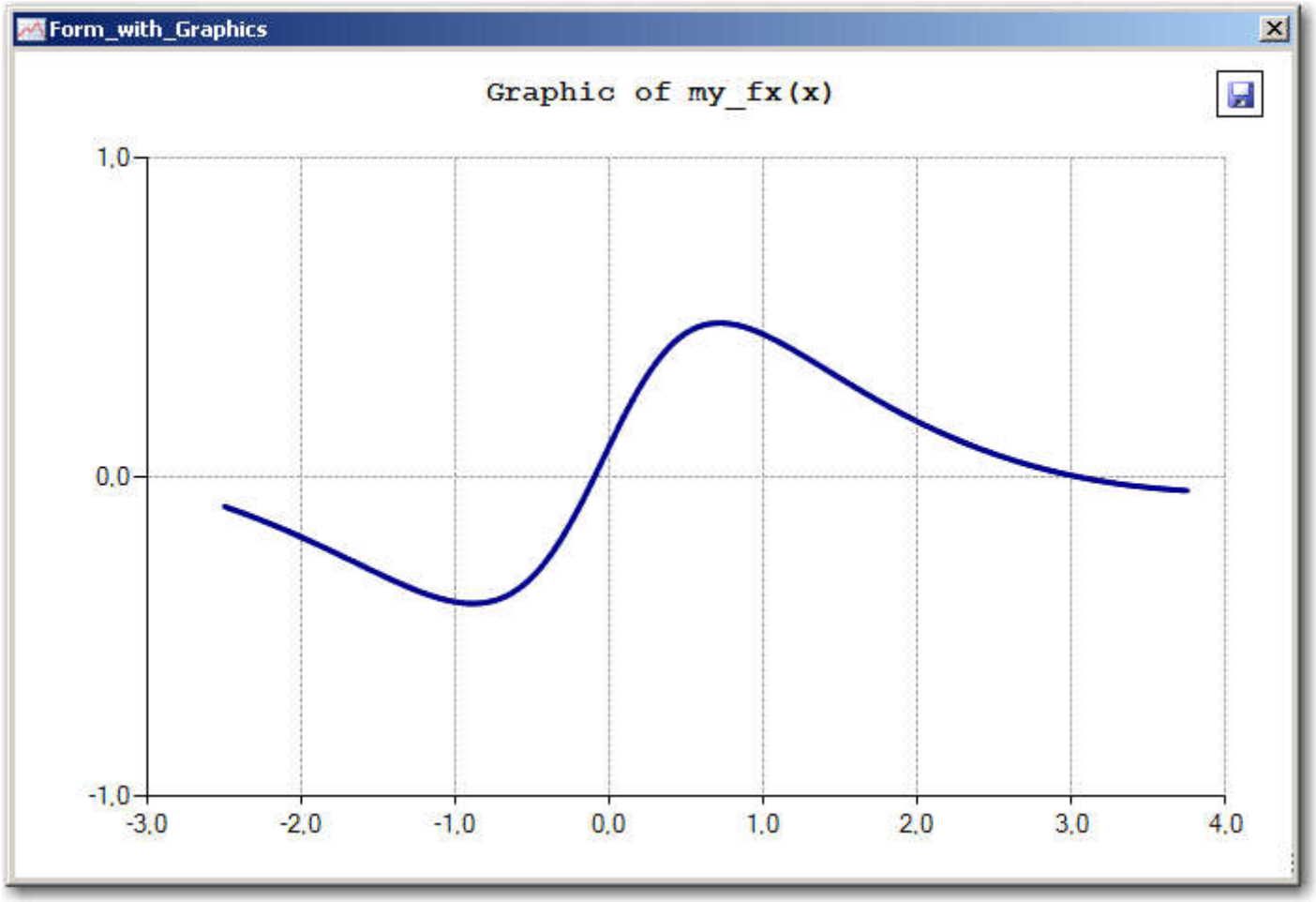
Таблиця ***тестових*** значень параметрів завдання для **0**-го варіанту:

n	x_o	x_n	$f(x)$
210	**-2.50**	**+3.75**	**fx_CT_1_0_v00**

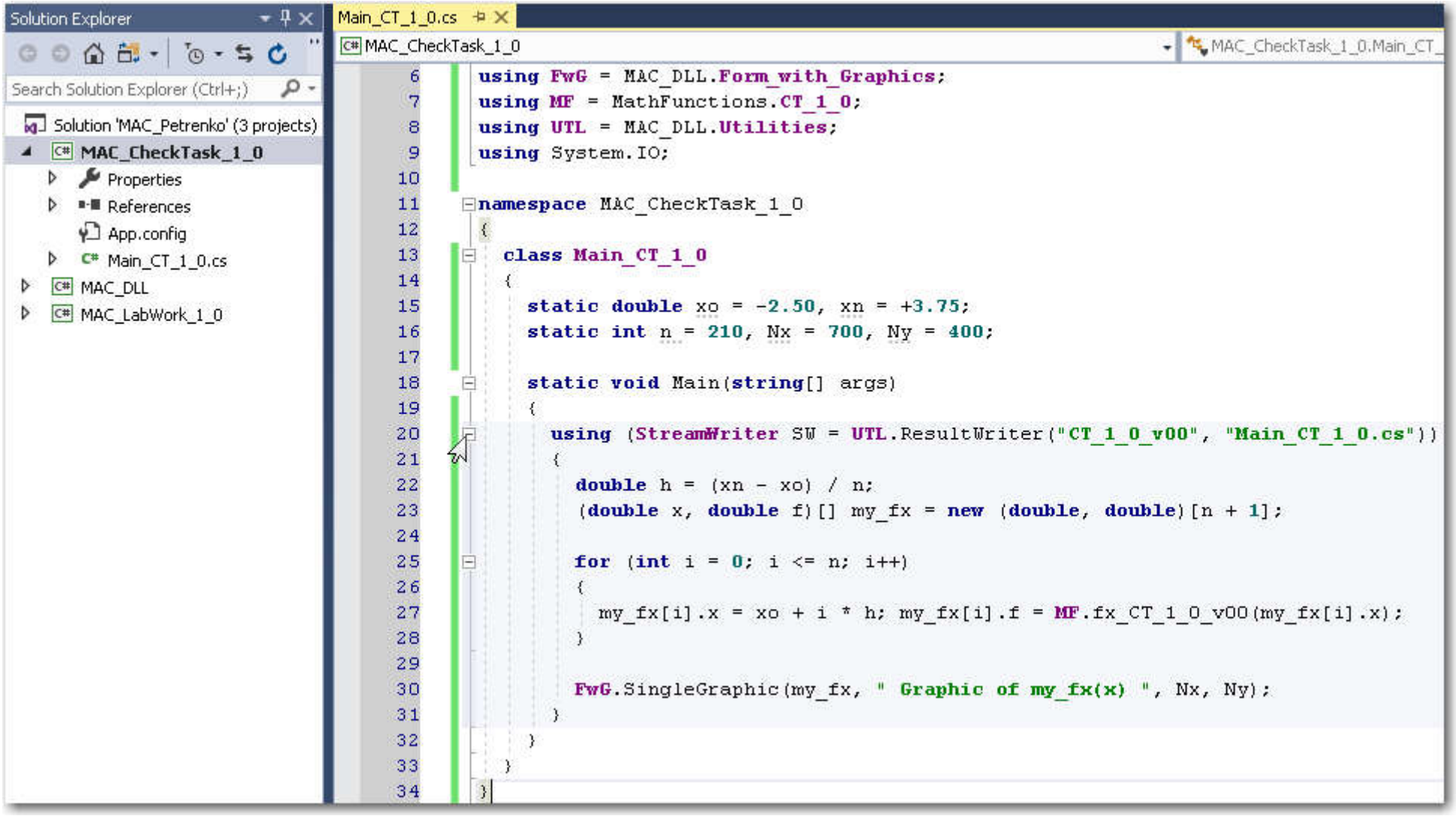
Тоді основна консольна програма, яка виконуватиме поставлене завдання, може бути, наприклад, такою:



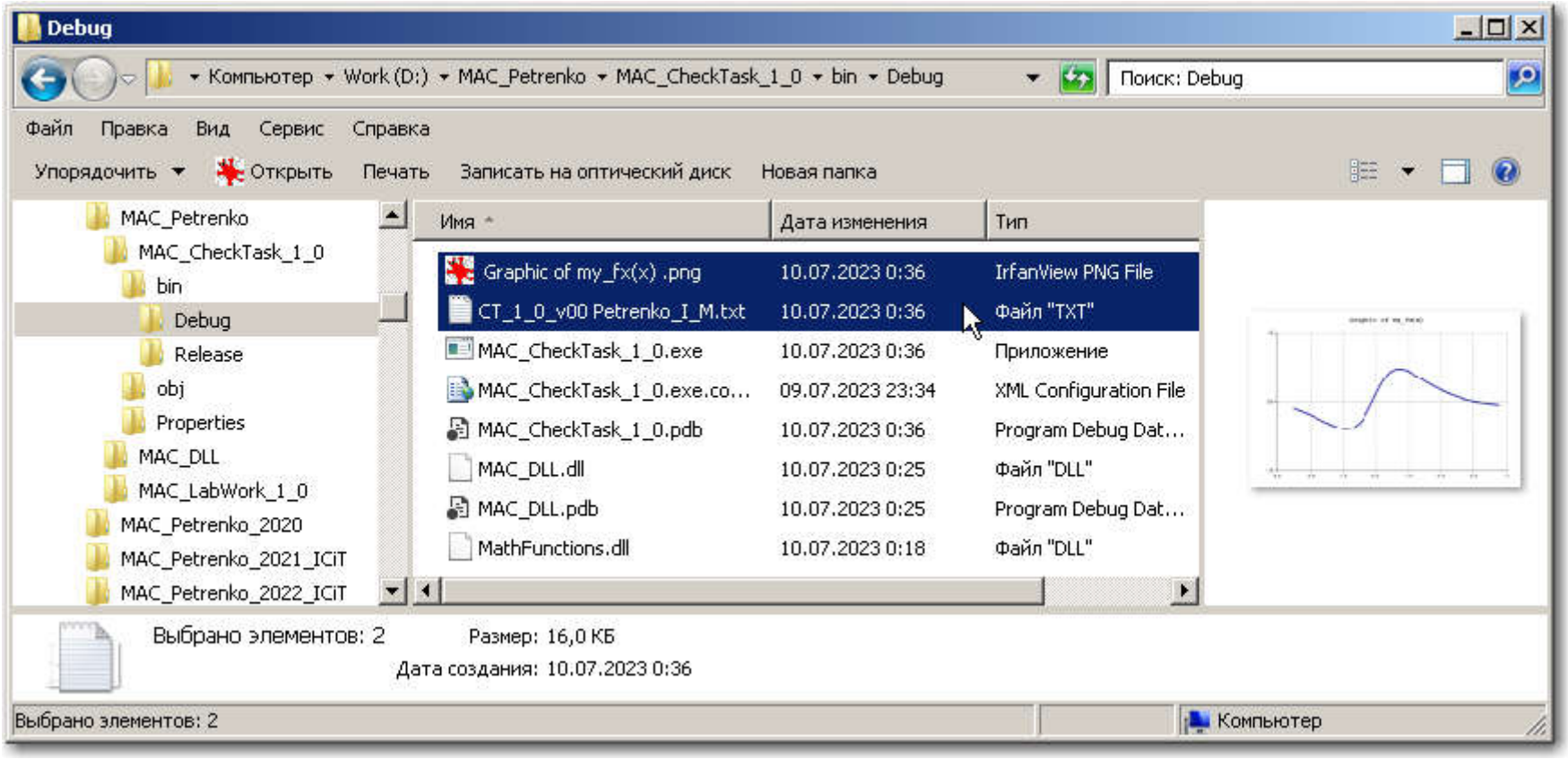
Результатом виконання програми повинен бути наступний графік:



Тепер продемонструємо, як із використанням методів класу **MAC_DLL.Utilities**, ми підготуємо текстовий файл, який буде звітом про виконання даного завдання.



Після запуску програми на виконання та натискання на відповідну кнопку на формі із графіком ми отримаємо два файли в робочій директорії нашого проекту:



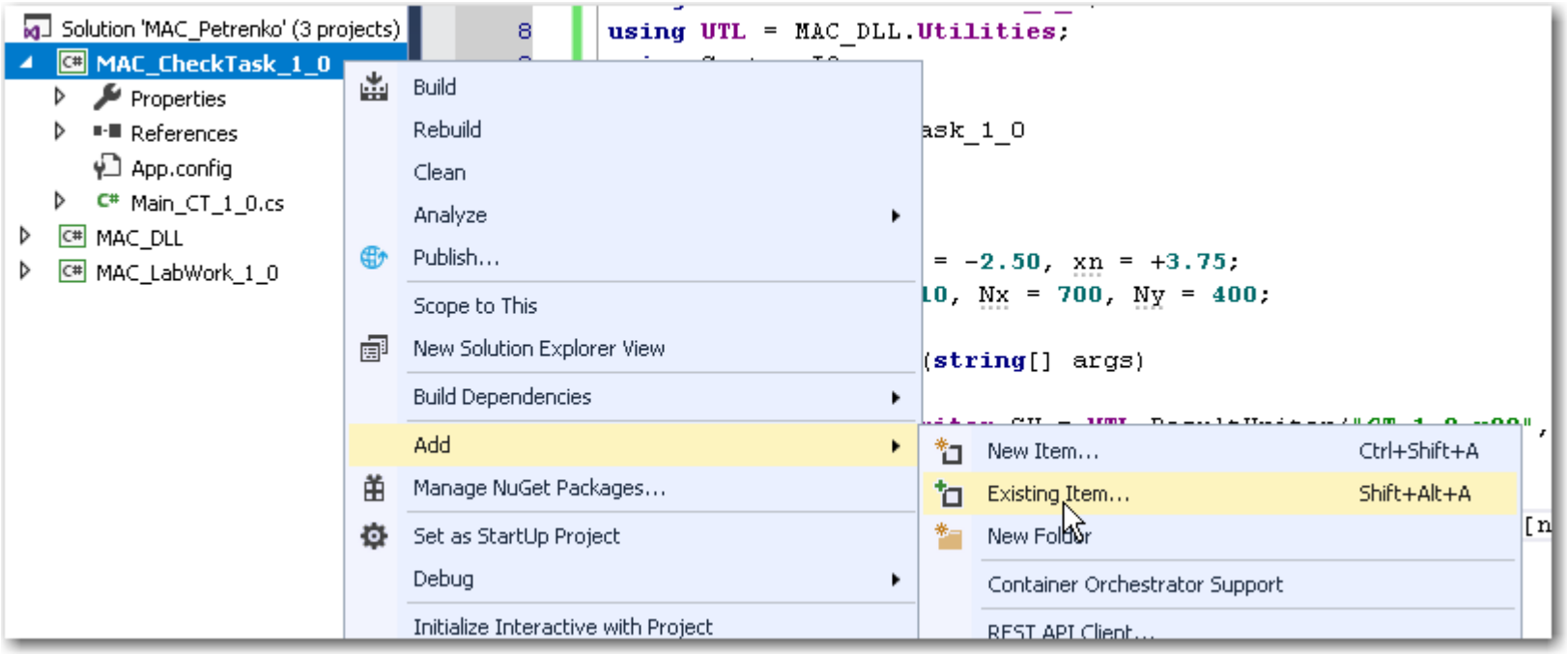
Ці два файли і будуть звітом про виконання вашого завдання, які ви надсилатимете вашому викладачу.

Перший файл – графічний, із зображенням відповідного графіку функції.

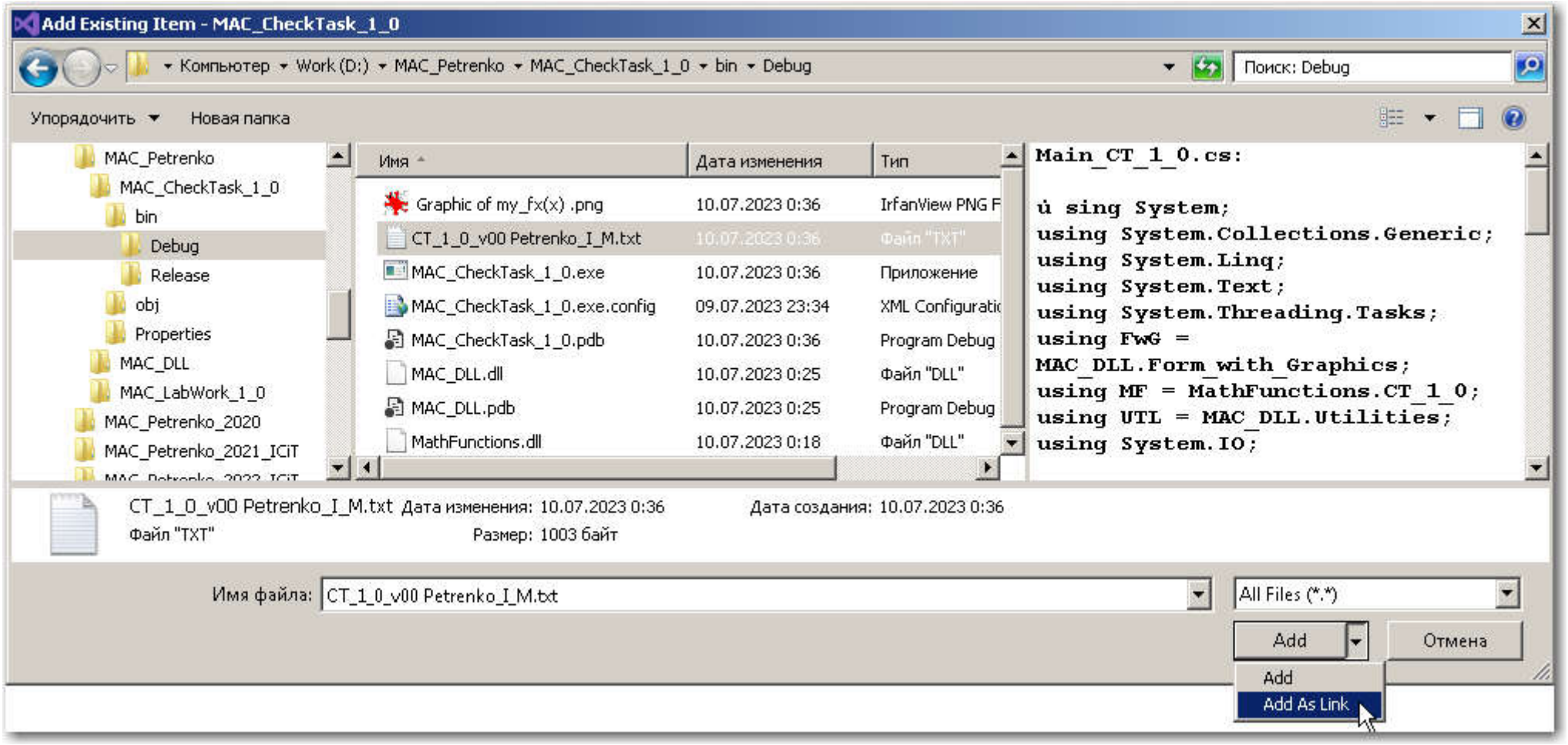
Другий файл, який містить в своєму імені код завдання, варіант та ваші особисті дані, є текстовим.

Його можна додати до робочого проекту (приєднати) та переглядати безпосередньо в середовищі розробки **MSVS**.

Для цього тицьнемо правою кнопкою на імені проекту:

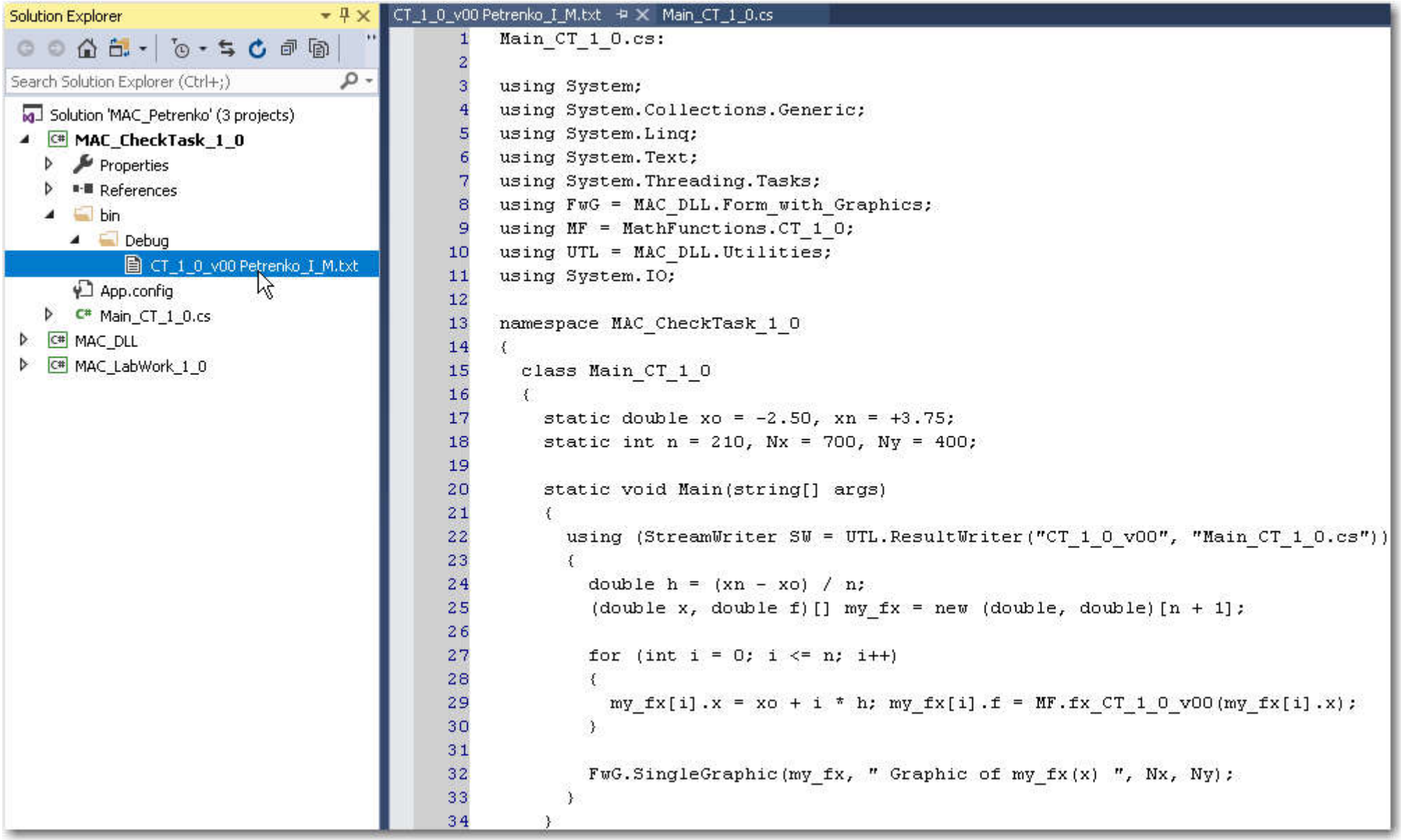


І оберемо послідовно пункти меню: **`Add -> Existing Item`**…



Знайдемо потрібний текстовий файл, після чого тицьнемо на кнопку **`Add As Link`**.

Тепер файл результатів **`CT_1_0_v00 Petrenko_I_M.txt`** ми можемо переглядати безпосередньо в робочому просторі, та він буде автоматично оновлюватися, якщо ми будемо виконувати ще якісь дії із кодом даного проекту та компілювати його:



Після того, як Ви отримали представлені вище результати, Вам слід самостійно реалізувати обчислення для власного варіанту даних, що наведений нижче у таблиці.

Саме ці результати будуть контрольними та будуть оцінюватися відповідними балами.

Надалі позначення **NN** буде завжди ідентифікувати двозначний номер вашого варіанту.

Таблиця *контрольних* значень параметрів завдання:

NN	n	x_o	x_n	$f(x)$	NN	n	x_o	x_n	$f(x)$
01	250	-2.75	+3.45	fx_CT_1_0_v01	11	350	-3.40	+3.70	fx_CT_1_0_v11
02	280	-4.25	+1.60	fx_CT_1_0_v02	12	330	-3.25	+2.90	fx_CT_1_0_v12
03	190	-1.05	+5.85	fx_CT_1_0_v03	13	310	-1.40	+4.15	fx_CT_1_0_v13
04	220	-1.95	+4.30	fx_CT_1_0_v04	14	290	-4.50	+0.60	fx_CT_1_0_v14
05	240	-1.80	+4.90	fx_CT_1_0_v05	15	275	-2.95	+3.25	fx_CT_1_0_v15
06	270	-1.65	+5.20	fx_CT_1_0_v06	16	245	-3.10	+3.50	fx_CT_1_0_v16
07	320	-3.50	+2.30	fx_CT_1_0_v07	17	340	-3.30	+3.20	fx_CT_1_0_v17
08	265	-4.10	+1.85	fx_CT_1_0_v08	18	355	-4.05	+4.10	fx_CT_1_0_v18
09	315	-6.70	-1.10	fx_CT_1_0_v09	19	410	-2.50	+2.60	fx_CT_1_0_v19
10	280	-5.40	-0.65	fx_CT_1_0_v10	20	420	-2.60	+2.50	fx_CT_1_0_v20

Файл **CT_1_0_vNN Petrenko_I_M.txt** з програмним кодом додатку та відповідний графічний файл **Graphic of my_fx(x) vNN.png** висилається викладачу для перевірки на адресу його електронної пошти.

Лабораторна робота 1.1

«Алгоритми обчислення сум числових рядів»

В даній лабораторній роботі розглядаються способи (алгоритми) обчислення сум S числових рядів $\sum a_k$, коли кількість їх членів-доданків є або заданим конкретним числом N, наприклад $S = \sum^{N} a_k$, або коли цей числовий ряд є нескінченним, тобто коли маємо $S = \sum^{\infty} a_k$.

Задача обчислення кінцевої суми числового ряду $S = \sum^{N} a_k$, вочевидь, є більш простішою, ніж задача обчислення суми аналогічного нескінченного числового ряду $S = \sum^{\infty} a_k$.

Причина криється у тому, що для нескінченного числового ряду ми не в змозі реально здійснити додавання всіх членів цього ряду.

Нескінченний числовий ряд, у підсумку, все одно буде замінений на відповідний кінцевий числовий ряд.

Питання лише в тому, як ця заміна остаточно вплине на точність підсумкового результату, який буде нами отриманий після проведення кінцевих обчислень.

Бажана точність результату (або похибка обчислень), як правило, задається в якості ключового параметра обчислювального процесу, яка обмежує його тривалість за часом.

Мета лабораторної роботи

1. Розробити алгоритми та на їх основі реалізувати програмні методи обчислення сум числових рядів, коли кількість доданків-членів числового ряду є кінцевою і коли вона передбачається нескінченною.

Зазначені методи обчислень слід розташовувати в спеціальному класі **MAC_Series** проекту динамічної бібліотеки **MAC_DLL**.

2. Створити та налагодити консольний додаток (окремий проект в існуючому робочому просторі **MAC_Petrenko**), призначений для обчислення «тестових» сум для двох заданих числових рядів.

3. Отримати підтвердження тестових результатів, які наводяться нижче в даному параграфі, що буде свідчити про завершення першого етапу даної лабораторної роботи.

Другий етап – проведення обчислень на основі індивідуального варіанта даної лабораторної роботи.

Результати обчислень, які отримує Студент, повинні відповідати визначеному формату (певній формі).

По завершенні лабораторного заняття ця форма із заповненими результатами подається викладачеві (або надсилається на електронну пошту разом із кодом програми) для перевірки.

Далі Вам буде продемонстрований формат звіту з даної лабораторної роботи.

Всі розроблені програмні компоненти зберігаються студентом до кінця навчального семестру.

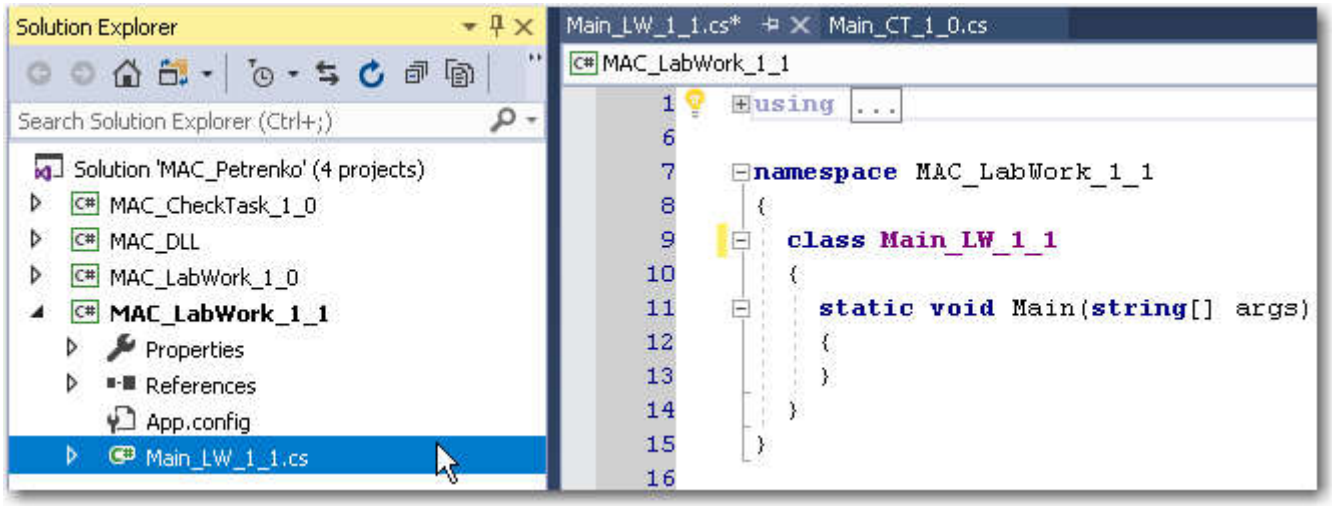
Порядок виконання завдання лабораторної роботи

Етап 1. Стартуємо середовище розробки **`MS Visual Studio`**.

В існуючому робочому просторі **`MAC_Petrenko`** генеруємо новий проект консольного додатку **`MAC_LabWork_1_1`**. Обираємо тип конфігурації робочого проекту – **`Debug`**.

Для зручності одразу перейменуємо стандартний клас **`Program`** в клас **`Main_LW_1_1`**.

Це допоможе Вам розрізняти файли **`*.cs`** різних проектів за їх унікальними іменами, коли подібних проектів у відкритому робочому просторі **`MAC_Petrenko`** буде декілька.



Перший етап лабораторної роботи буде полягати в створенні універсального метода, який здійснюватиме обчислення кінцевої суми членів деякого чисельного ряду.

Зазначена вище задача має наступне математичне формулювання:

$$S = \sum_{k=k_I}^{k_L} a_k , \qquad \text{(л.1.1.1)}$$

де S – шукане значення суми, (**sum**),

k – індекс додавання, (**index**), інкремент індексу додавання дорівнює 1,

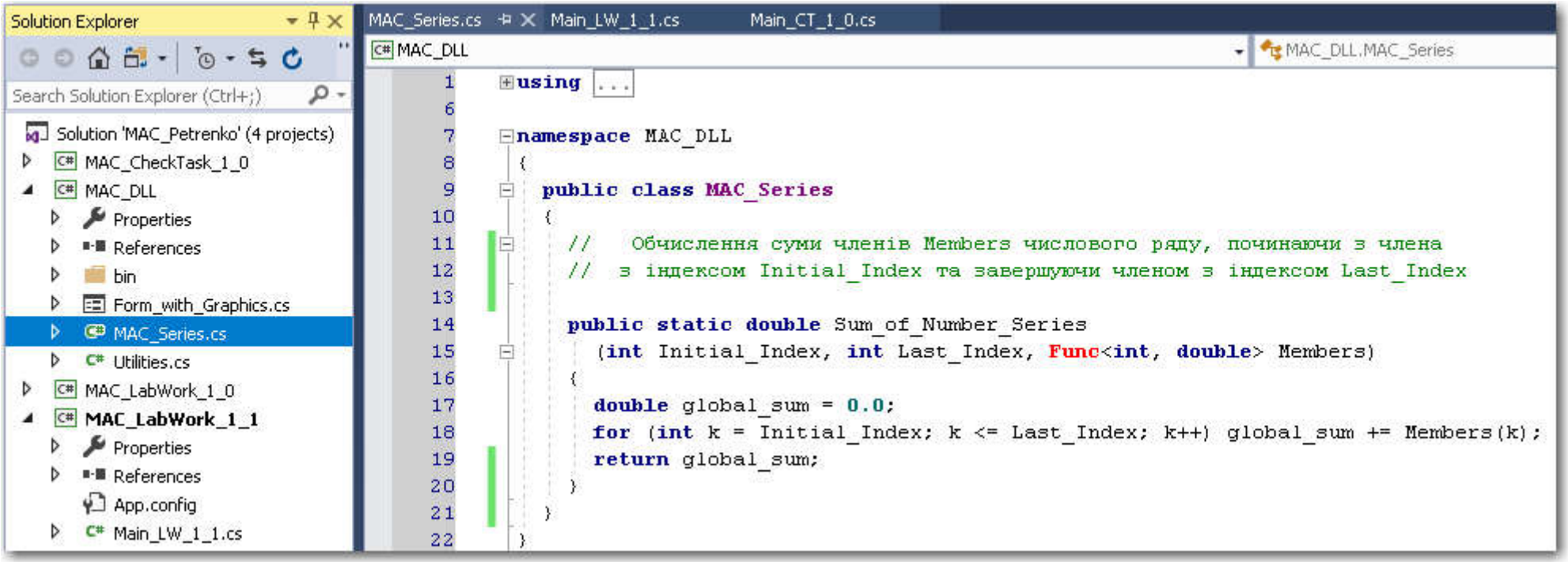
k_I – початкове значення індексу, (**initial index**),

k_L – кінцеве значення індексу, (**last index**),

a_k – член числового ряду, який функціонально залежить від індексу k, (**member**),

наприклад, $a_k = \dfrac{k+1}{(k+2)^3}$.

Метод **`Sum_of_Number_Series()`**, який нами розробляється, розташуємо в динамічній бібліотеці **`MAC_DLL`** в окремому новому класі – **`MAC_Series`**:



Суть алгоритму, який реалізовано в методі **`Sum_of_Number_Series()`**, можна легко зрозуміти з його математичного визначення (л.1.1.1).

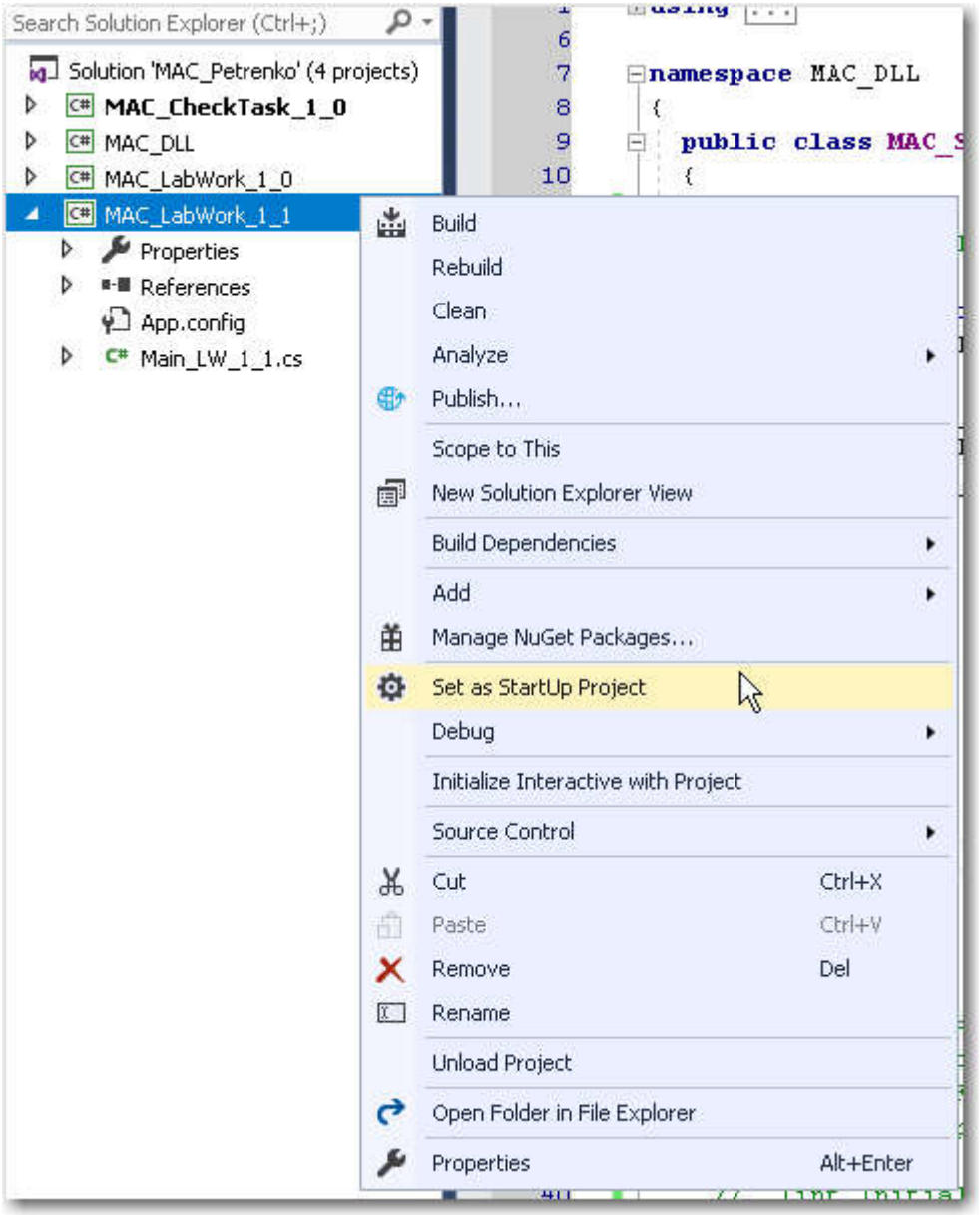
Нехай в межах завдання на лабораторну роботу нам пропонуються наступні нескінченні числові ряди, які будемо використовувати в якості «тестових»:

$$S_1 = \sum_{k=1}^{\infty} \frac{1}{(2k+1)^2} = \frac{\pi^2}{8} - 1 \quad \text{та} \quad S_2 = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)^3} = \frac{\pi^3}{32}. \qquad \text{(л.1.1.2)}$$

Для тестових рядів (л.1.1.2) аналітичним шляхом вже обчислені вирази для точних значень їх сум, коли кількість членів в цих рядах є саме нескінченною.

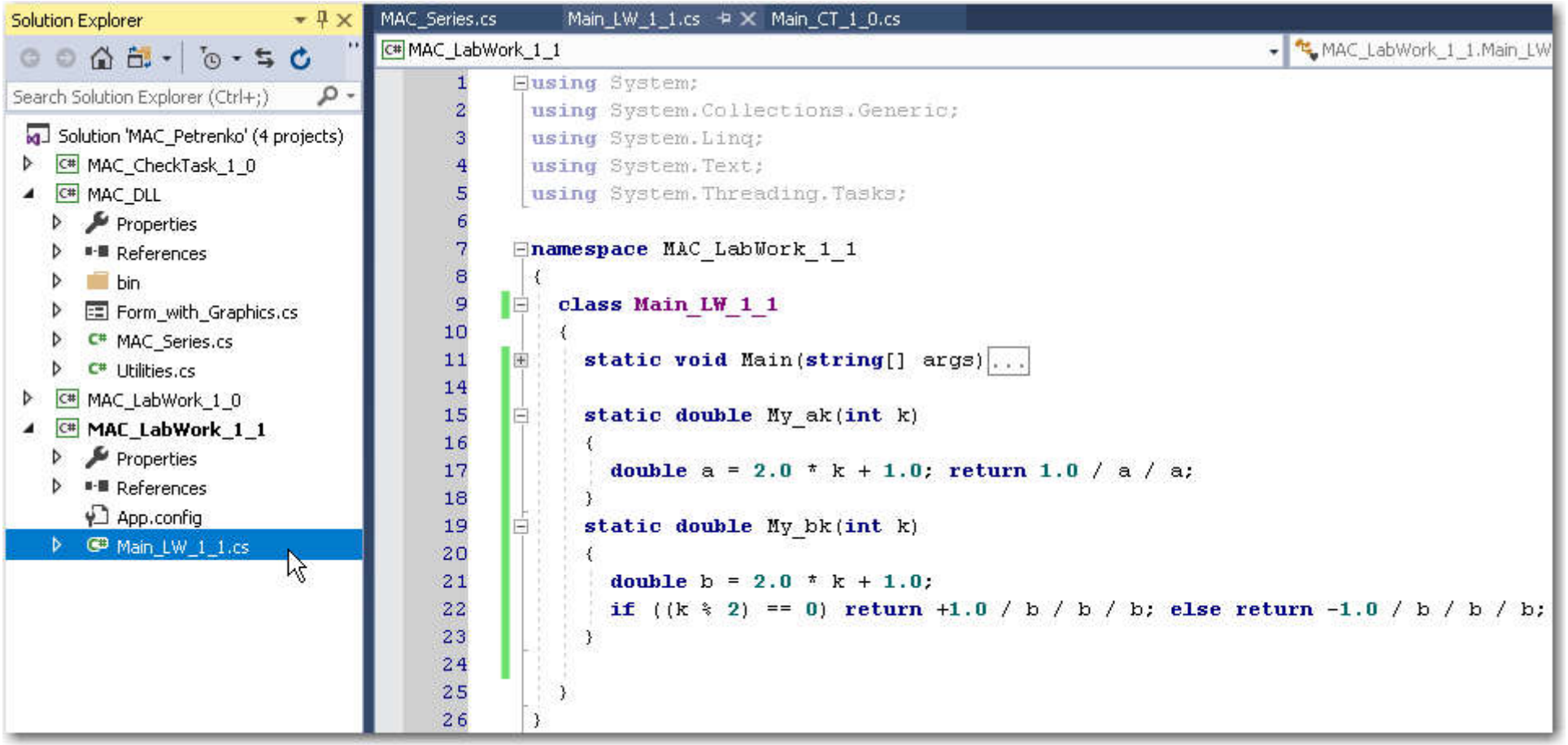
Зверніть також увагу на те, що другий числовий ряд S_2, на відміну від першого S_1, є знакозмінним (члени другого числового ряду по черзі змінюють свій знак).

Переходимо до проекту **`MAC_LabWork_1_1`**, який зробимо «стартовим»:

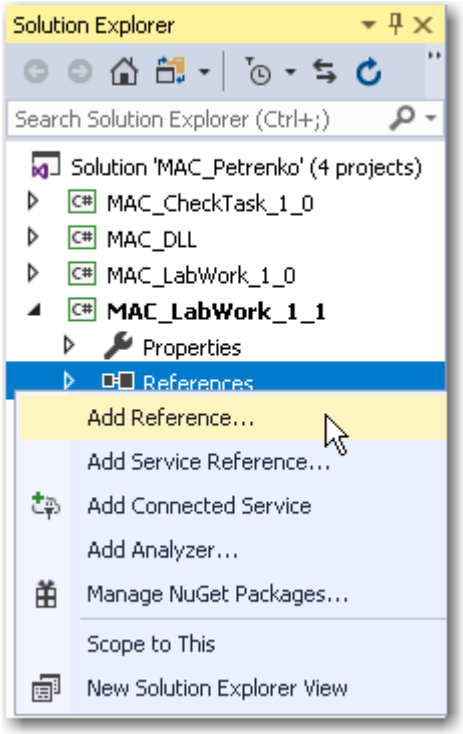


В класі **Main_LW_1_1** ми створимо коди статичних методів, які відповідатимуть сигнатурі узагальненого делегата **Func<int,double>**, та які реалізовуватимуть обчислення членів тестових рядів (л.1.1.2):

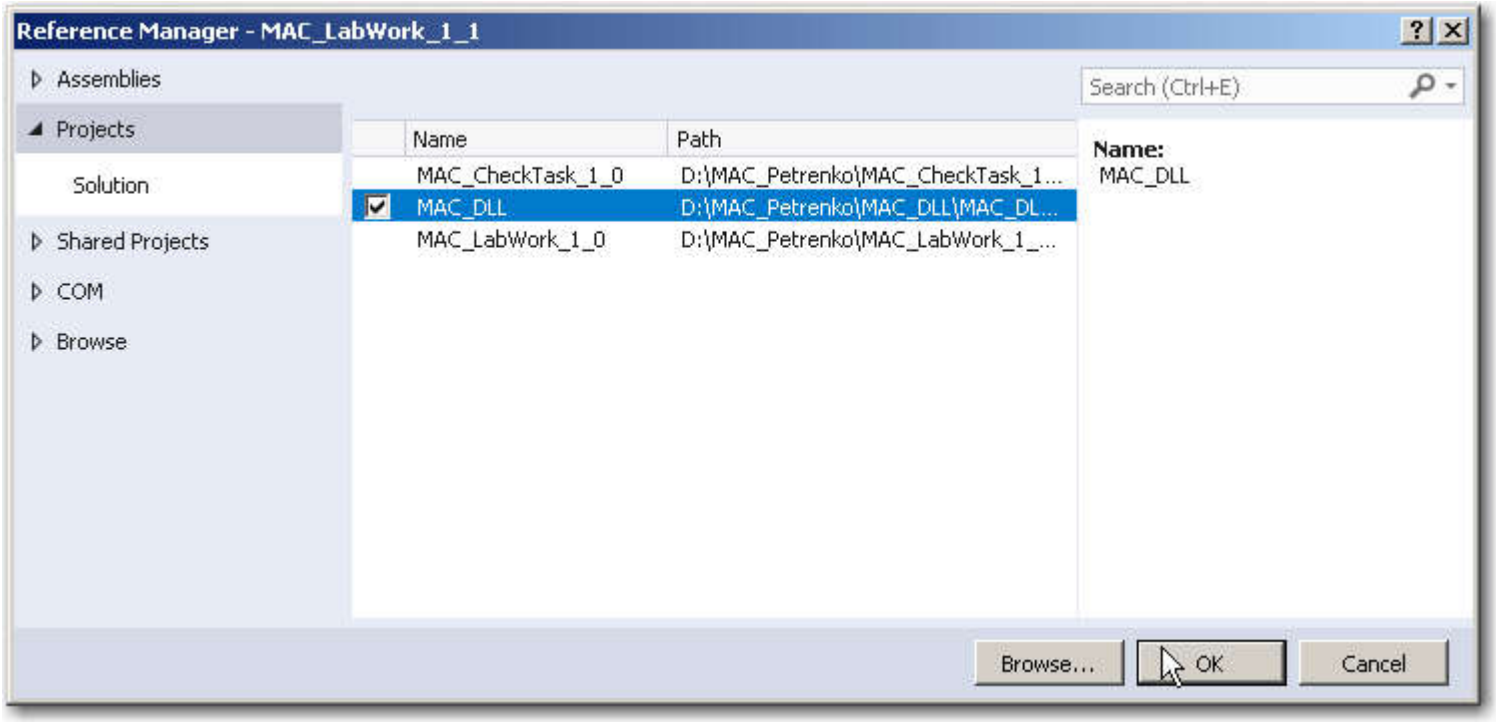
$$a_k = \frac{1}{(2k+1)^2} \quad \text{та} \quad b_k = \frac{(-1)^k}{(2k+1)^3}. \qquad \text{(л.1.1.3)}$$



Тепер (способом, який для Вас вже є відомим) слід добавити динамічну бібліотеку **`MAC_DLL`** до розділу **`References`** консольного проекту **`MAC_LabWork_1_1`**:



і

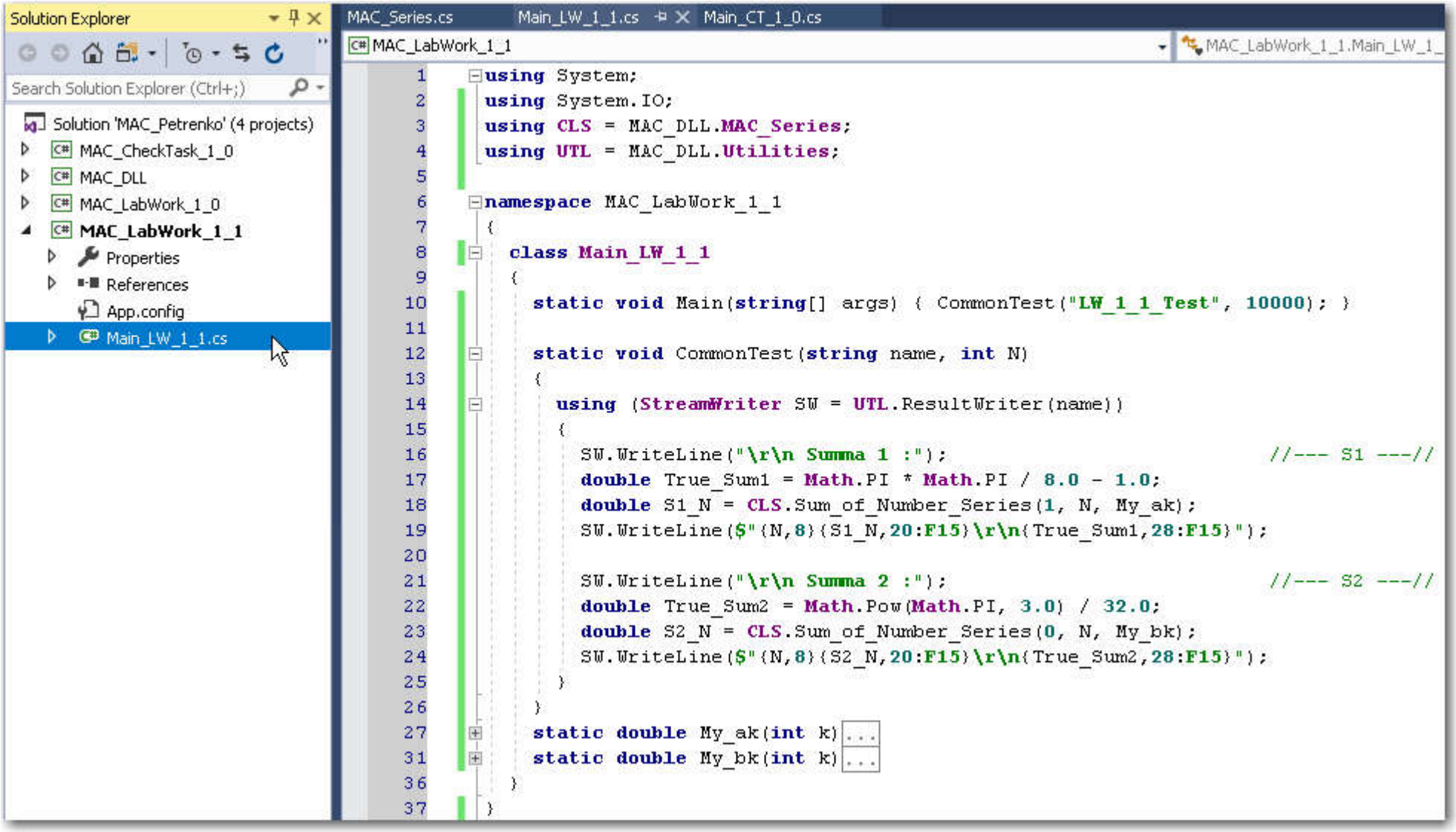


Тепер все готово для розробки першої частини основного додатку даної лабораторної роботи, де обчислюються, наприклад, такі «кінцеві» тестові суми (л.1.1.2):

$$S_1 = \sum_{k=1}^{N} \frac{1}{(2k+1)^2} \quad \text{та} \quad S_2 = \sum_{k=0}^{N} \frac{(-1)^k}{(2k+1)^3}, \qquad \text{(л.1.1.4)}$$

коли задано $N = 10\ 000$ – верхню границю (найбільше значення) для індексу додавання.

Розглянемо наступний код консольного додатку:



Тестування виконується в окремому методі **`CommonTest()`**, який має два формальні параметри.

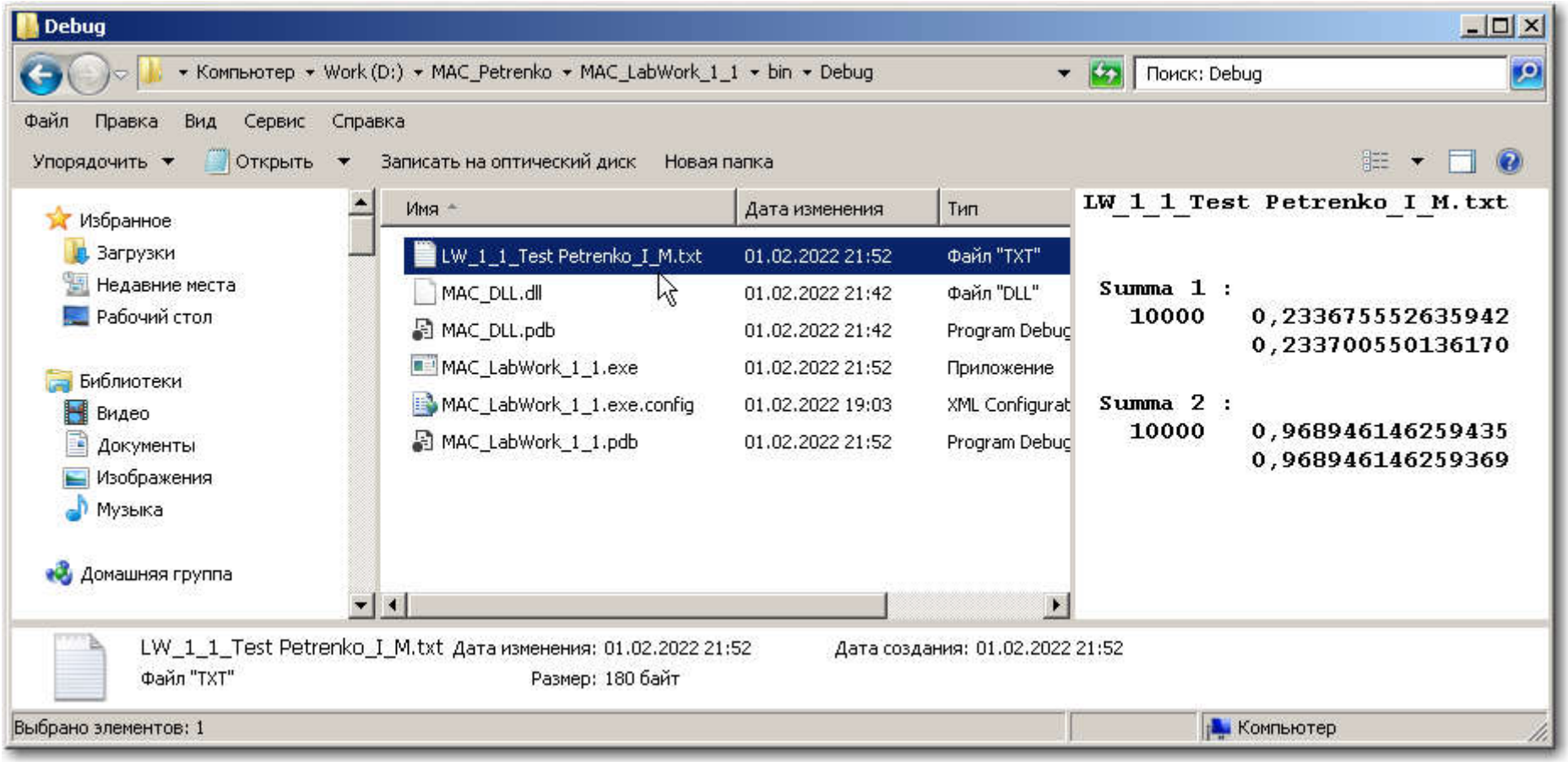
Виклик цього методу здійснюється в основній функції **`Main()`** консольного додатку.

Під час виклику формальні параметри набувають фактичних значень.

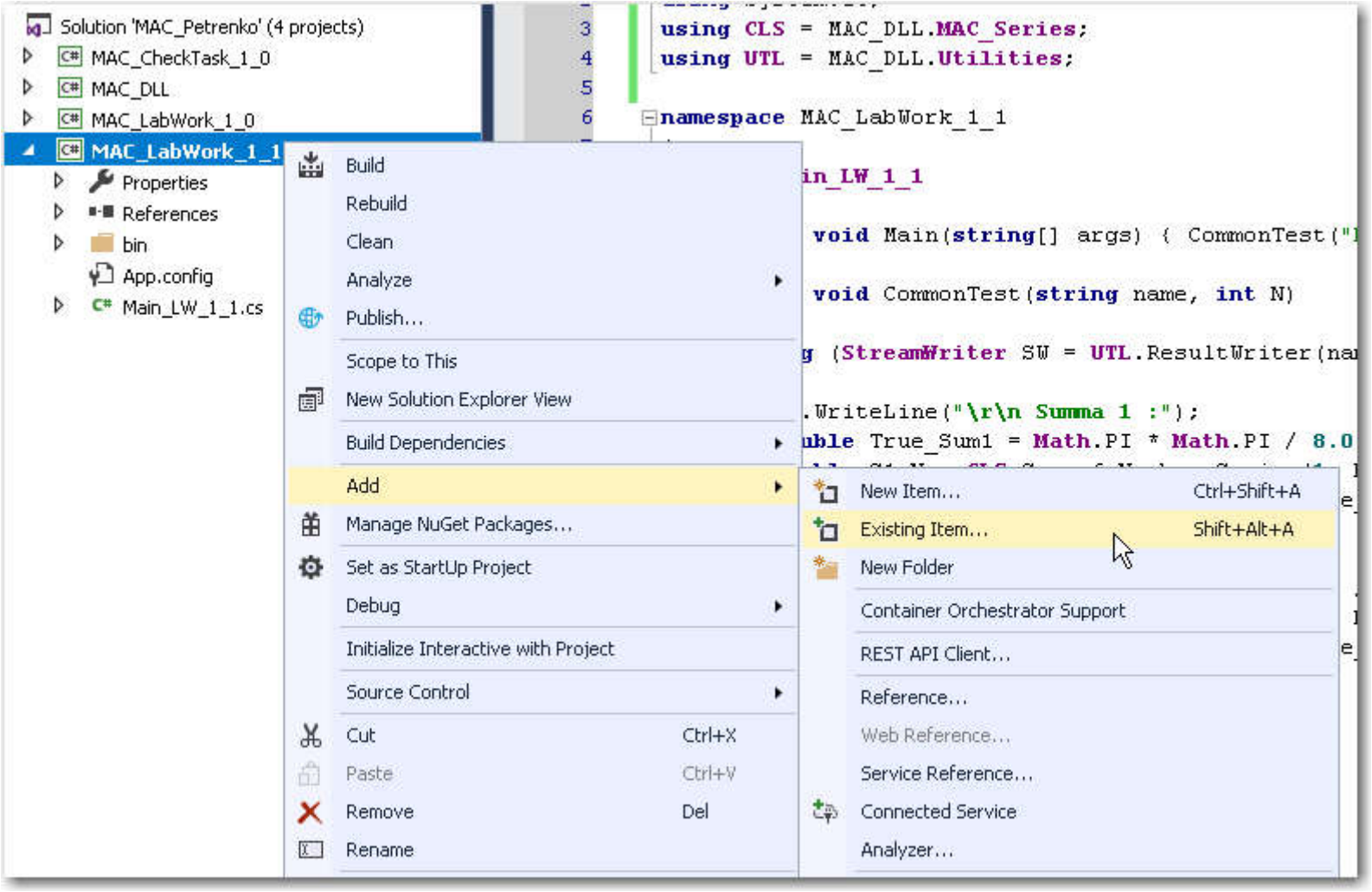
Перший фактичний параметр – рядкова змінна, яка є м.'ям текстового файлу, в якому буде збережено результати обчислень.

Другий фактичний параметр – кількість членів числових рядів (л.1.1.4), які повинні увійти до відповідних сум.

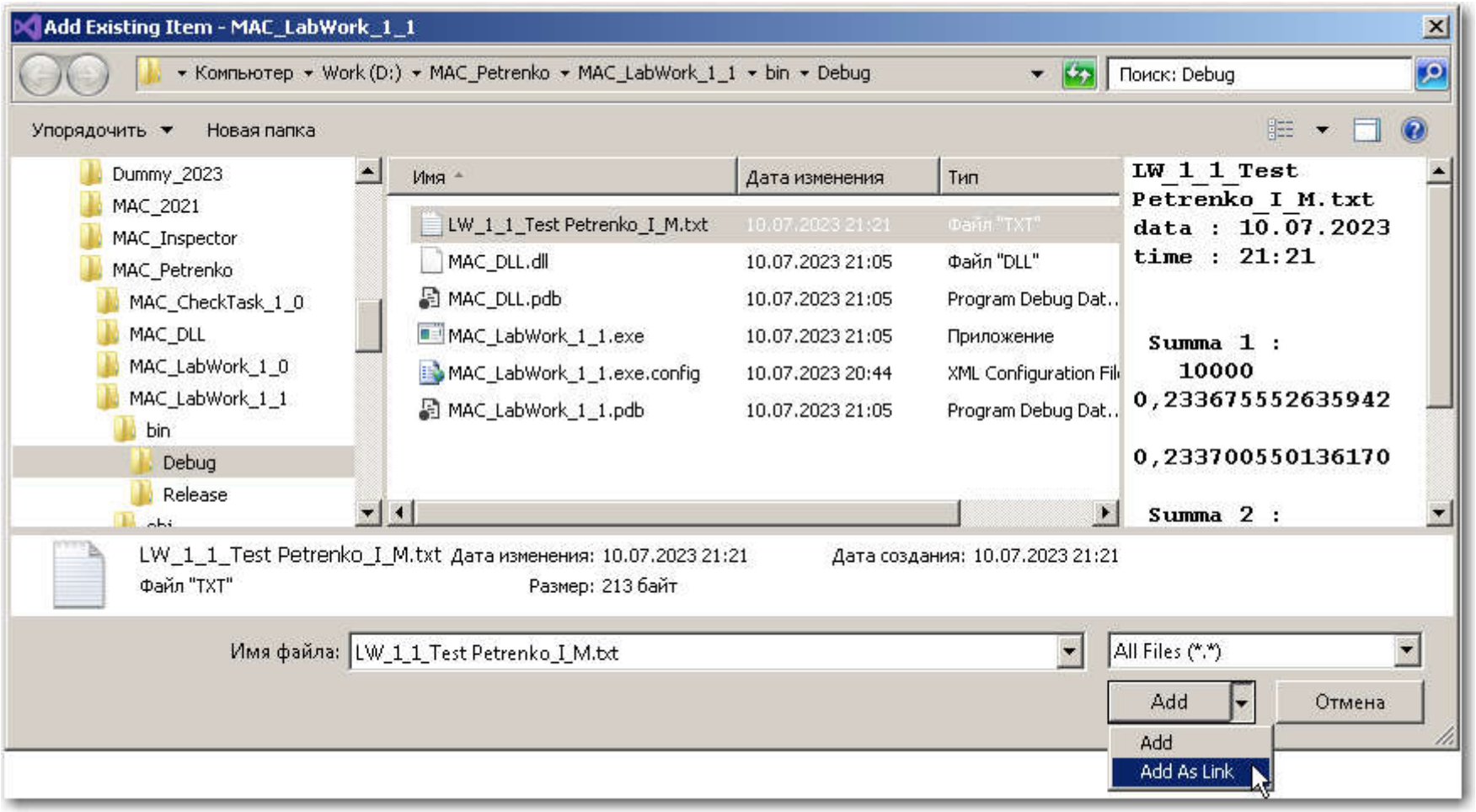
Отже запустимо проект **`MAC_LabWork_1_1`** на виконання (**`CTRL+F5`**). Маємо файл результатів (якщо його передивлятися через «Провідник» в директорії проекту **`MAC_LabWork_1_1`**):


Debug
Компьютер ▸ Work (D:) ▸ MAC_Petrenko ▸ MAC_LabWork_1_1 ▸ bin ▸ Debug
Поиск: Debug
Файл Правка Вид Сервис Справка
Упорядочить Открыть Записать на оптический диск Новая папка
Избранное
Загрузки
Недавние места
Рабочий стол
Библиотеки
Видео
Документы
Изображения
Музыка
Домашняя группа
Имя
Дата изменения
Тип
LW_1_1_Test Petrenko_I_M.txt 01.02.2022 21:52 Файл "TXT"
MAC_DLL.dll 01.02.2022 21:42 Файл "DLL"
MAC_DLL.pdb 01.02.2022 21:42 Program Debug
MAC_LabWork_1_1.exe 01.02.2022 21:52 Приложение
MAC_LabWork_1_1.exe.config 01.02.2022 19:03 XML Configurat
MAC_LabWork_1_1.pdb 01.02.2022 21:52 Program Debug
LW_1_1_Test Petrenko_I_M.txt
Summa 1 :
10000 0,233675552635942
0,233700550136170
Summa 2 :
10000 0,968946146259435
0,968946146259369
LW_1_1_Test Petrenko_I_M.txt Дата изменения: 01.02.2022 21:52 Дата создания: 01.02.2022 21:52
Файл "TXT" Размер: 180 байт
Выбрано элементов: 1
Компьютер

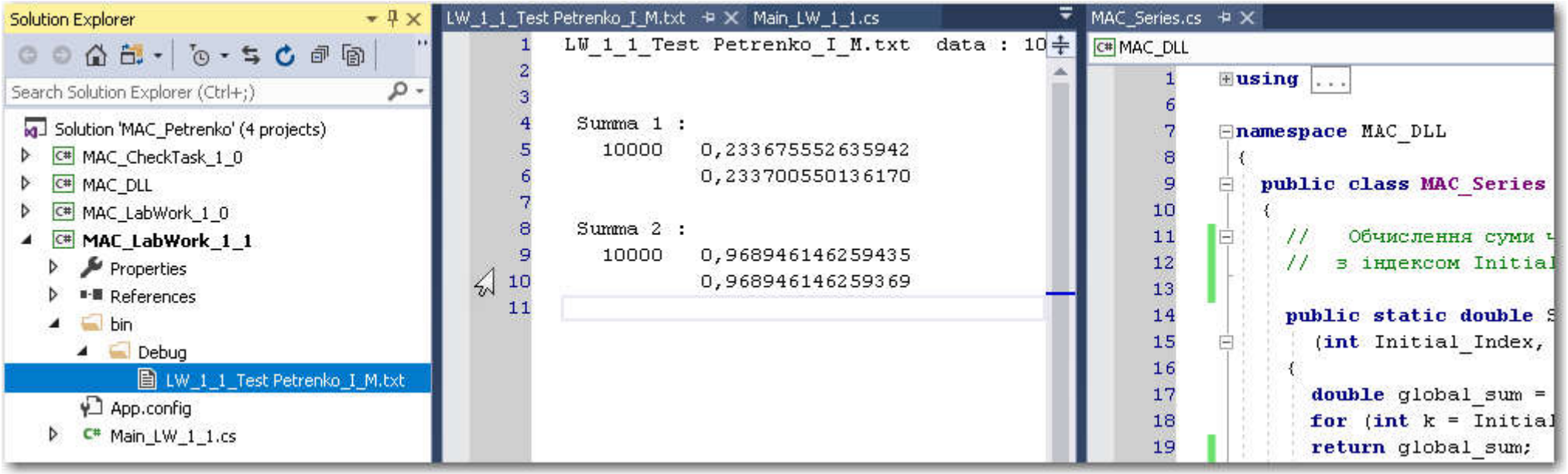
Одразу включимо цей файл до робочого простору **MAC_Petrenko** до складу проекту **MAC_LabWork_1_1**, аби ми мали можливість переглядати результати обчислень безпосередньо у середовищі розробки **MS VS**:



Обираємо текстовий файл результатів **"LW_1_1_Test Petrenko_I_M.txt"** та приєднуємо його до проекту **MAC_LabWork_1_1** у якості посилання (**Add As Link**):



Тепер переглядаємо його безпосередньо в середовищі розробки:



Для контролю якості результату одночасно обчислюються ***точні*** значення сум (л.1.1.2) для нескінченних рядів.

Вочевидь, що із зростанням значення параметра N, різниця відповідних значень кінцевих та нескінченних сум повинна зменшуватися.

Отже, ми маємо достатньо переконливий результат – збіг певної кількості значущих цифр в обох сумах, який можна спостерігати візуально завдяки формату виводу числових даних (в стовпчик).

Таким чином, алгоритм обчислення суми кінцевої кількості доданків для заданого чисельного ряду, який реалізовано в статичному методі **`Sum_of_Number_Series()`**, можна вважати математично та програмно налагодженим.

Етап 2. Другий етап лабораторної роботи буде полягати в створенні універсального метода, який здійснює ***наближене*** обчислення «нескінченої» суми членів деякого числового ряду із заданою ***абсолютною*** похибкою ε.

Зазначена задача має наступне математичне формулювання:

$$S = \sum_{k=k_I}^{\infty} a_k = \sum_{k=k_I}^{k_F} a_k + \sum_{k=k_F+1}^{\infty} a_k = S(k_F) + \tilde{S} \approx S(k_F), \qquad |\tilde{S}| < \varepsilon, \tag{л.1.1.5}$$

де $S(k_F)$ – шукане наближене значення суми, (**sum**),

k – індекс додавання, (**index**), інкремент індексу додавання, який дорівнює 1,

k_I – початкове значення індексу, (**initial index**),

k_F – кінцеве значення індексу, (**final index**), яке визначається саме по завершенню обчислень,

$\tilde{S}$ – сума тієї частини нескінченого числового ряду, яка відкидається, і для якої передбачається виконання нерівності $|\tilde{S}| < \varepsilon$.

Дану задачу, взагалі кажучи, не можна признати математично коректною, оскільки для довільного числового ряду ми не зможемо програмним шляхом «оцінити» порядок суми $\tilde{S}$ тієї частини ряду, яка «відкидається», по відношенню до підсумкового остаточного результату $S(k_F)$.

Можна запропонувати деякий алгоритм, який буде нам давати задовільний результат для числових рядів, які добре (швидко) сходяться, та буде давати певну похибку для числових рядів, що сходяться погано (повільно).

Наша основна мета – правильно запрограмувати зазначений числовий алгоритм та потім перевірити його на зрозумілих тестових прикладах.

Отже, суть запропонованого алгоритму наступна.

На ***першому етапі*** обчислення суми ряду здійснюється підрахунок суми s_0 членів від початкового члена a_{k_I}, до того члена ряду a_n, модуль якого буде меншим ніж задана похибка обчислень, тобто $s_0 = \sum_{k=k_I}^{n} a_k$, де $|a_n| < \varepsilon$.

Таким чином, сума s_0 буде обчислена від початкового члена ряду a_{k_I} до члена a_n включно, а загальна кількість членів N в цій сумі становитиме $N = n - k_I + 1$.

Чим повільніше (гірше) числовий ряд збігається, тем більшою виявляється ця кількість N.

Наприклад, якщо $k_I = 0$, а $|a_5| < \varepsilon$, тобто $n = 5$, то сума s_0 буде обчислена на основі $N = 5 - 0 + 1 = 6$ саме шести перших членів ряду: a_0, a_1, …, a_5.

Другий етап алгоритму.

Наступна сума, яка обчислюється за даним алгоритмом, – s_1 – це сума саме з N наступних членів даного числового ряду. При цьому першим доданком цієї суми буде член a_{n+1}, а останнім – a_{n+1+N}.

В нашому «прикладі» це будуть члени від a_6 і до a_{11} включно.

Отримана сума s_1 одразу додається до вже наявного значення s_0. Якщо при цьому виконується нерівність $|s_1| < \varepsilon$, то даний обчислювальний процес припиняється і результатом буде вже «оновлена» сума s_0.

Якщо ж має місце $|s_1| \geq \varepsilon$, то обчислюється ***нова*** сума s_1 з N членів заданого ряду.

І так далі… до виконання умови $|s_1| < \varepsilon$.

Зазначений алгоритм оформимо у вигляді нового метода **Sum_of_Number_Series_A()**, який додамо до класу **MAC_Series** динамічної бібліотеки **MAC_DLL**, який нами щойно розробляється.

```
public class MAC_Series
{
  //   Обчислення суми членів Members числового ряду, починаючи з члена
  //  з індексом Initial_Index та завершуючи членом з індексом Last_Index

  public static double Sum_of_Number_Series
    (int Initial_Index, int Last_Index, Func<int, double> Members)...

  //  Обчислення суми членів Members числового ряду, починаючи з члену
  //   з індексом Initial_Index, та із заданою абсолютною похибкою Eps.
  //  Параметр Final_Index повертає індекс останнього члена ряду, що
  //   був використаний під час підрахунку загальної суми ряду.

  public static double Sum_of_Number_Series_A
    (int Initial_Index, double Eps, Func<int, double> Members, ref int Final_Index)
  {
    double ak, sum_1, sum_0 = 0.0; int k = Initial_Index; bool flag;
    do
    {
      ak = Members(k); sum_0 += ak; flag = (Math.Abs(ak) >= Eps); k++;
    } while (flag);
    int N = k - Initial_Index;
    do
    {
      sum_1 = Sum_of_Number_Series(k, k + N, Members);
      sum_0 += sum_1; flag = (Math.Abs(sum_1) >= Eps); if (flag) k += N + 1;
    } while (flag);
    Final_Index = k + N; return sum_0;
  }
}
```

Примітка. При розробці частини програмного коду, який реалізує другий етап даного алгоритму – етап обчислення сум s_1, ми скористалися методом **Sum_of_Number_Series()**, який вже існує в бібліотеці **MAC_DLL** та призначений для обчислення скінчених сум.

Іноді наближені обчислення треба виконувати із заданою ***відносною*** похибкою δ.

Це також стосується і задач обчислення сум нескінчених числових рядів.

Вочевидь, що ми можемо скористатися вже розробленим попереднім алгоритмом, де обчислення здійснювалися з урахуванням ***абсолютної*** похибки ε.

Перший етап нового алгоритму є схожим до першого етапу попереднього алгоритму, на якому перевіряється виконання умови $| a_n / s_0 | < \delta$ та визначається значення N.

На другому етапі нового алгоритму циклічні обчислення повинні продовжуватися за наступним правилом.

Якщо виконується нерівність $| s_1/s_0 | < \delta$, то даний ітераційний процес завершується і результатом буде признана «оновлена» сума s_0. Мається на увазі те, що сума ряду s_0 на кожній ітерації повинна обов'язково «оновлюватися» шляхом додавання до неї нового значення часткової суми s_1.

Якщо ж має місце $| s_1/s_0 | \geq \delta$, то обчислюється нова сума s_1 з наступних N членів заданого числового ряду.

І так далі… до виконання умови $| s_1/s_0 | < \delta$.

Даний алгоритм обчислення суми числового ряду (з урахуванням відносної похибки) оформимо у вигляді нового статичного метода **`Sum_of_Number_Series_D()`**, і також додамо до класу **`MAC_Series`** динамічної бібліотеки **`MAC_DLL`**:

```
9   public class MAC_Series
10  {
11    //   Обчислення суми членів Members числового ряду, починаючи з члена
12    //  з індексом Initial_Index та завершуючи членом з індексом Last_Index
13    public static double Sum_of_Number_Series
14      (int Initial_Index, int Last_Index, Func<int, double> Members)...
20
21    //  Обчислення суми членів Members числового ряду, починаючи з члену
22    //   з індексом Initial_Index, та із заданою абсолютною похибкою Eps.
23    //  Параметр Final_Index повертає індекс останнього члена ряду, що
24    //   був використаний під час підрахунку загальної суми ряду.
25    public static double Sum_of_Number_Series_A
26      (int Initial_Index, double Eps, Func<int, double> Members, ref int Final_Index)...
41
42    //  Обчислення суми членів Members числового ряду, починаючи з члену
43    //   з індексом Initial_Index, та із заданою відносною похибкою Delta.
44    //  Параметр Final_Index повертає індекс останнього члена ряду, що
45    //   був використаний під час підрахунку загальної суми ряду.
46
47    public static double Sum_of_Number_Series_D
48      (int Initial_Index, double Delta, Func<int, double> Members, ref int Final_Index)
49    {
50      double mem_k, partial_sum, global_sum = 0.0;
51      int k = Initial_Index; bool flag;
52      do...
57
58      int N = k - Initial_Index;
59      do...
66      Final_Index = k + N; return global_sum;
67    }
68  }
```

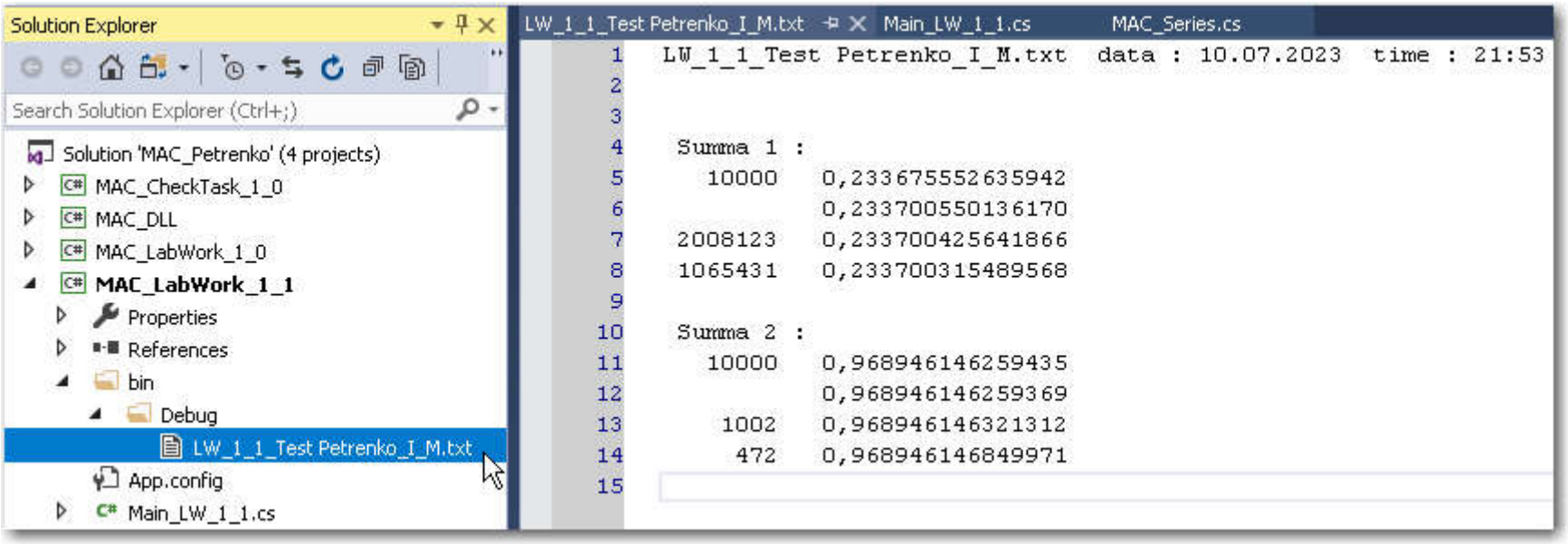
Цю частину лабораторної роботи Ви повинні виконати самостійно.

Тестування розроблених методів (алгоритмів обчислення сум числових рядів) будемо виконувати за допомогою консольного додатку **MAC_LabWork_1_1**.

Збільшимо обсяг обчислень в рамках тестового метода **CommonTest()**:

```
6   namespace MAC_LabWork_1_1
7   {
8     class Main_LW_1_1
9     {
10      static int kF = 0; static double Eps = 1.0E-9, Dlt = 1.0E-8;
11
12      static void Main(string[] args) { CommonTest("LW_1_1_Test", 10000); }
13
14      static void CommonTest(string name, int N)
15      {
16        using (StreamWriter SW = UTL.ResultWriter(name))
17        {
18          SW.WriteLine("\r\n Summa 1 :");                              //--- S1 ---//
19          double True_Sum1 = Math.PI * Math.PI / 8.0 - 1.0;
20          double S1_N = CLS.Sum_of_Number_Series(1, N, My_ak);
21          SW.WriteLine($"{N,8}{S1_N,20:F15}\r\n{True_Sum1,28:F15}");
22
23          double S1_A = CLS.Sum_of_Number_Series_A(1, Eps, My_ak, ref kF); SW.WriteLine($"{kF,8}{S1_A,20:F15}");
24          double S1_D = CLS.Sum_of_Number_Series_D(1, Dlt, My_ak, ref kF); SW.WriteLine($"{kF,8}{S1_D,20:F15}");
25
26          SW.WriteLine("\r\n Summa 2 :");                              //--- S2 ---//
27          double True_Sum2 = Math.Pow(Math.PI, 3.0) / 32.0;
28          double S2_N = CLS.Sum_of_Number_Series(0, N, My_bk);
29          SW.WriteLine($"{N,8}{S2_N,20:F15}\r\n{True_Sum2,28:F15}");
30
31          double S2_A = CLS.Sum_of_Number_Series_A(0, Eps, My_bk, ref kF); SW.WriteLine($"{kF,8}{S2_A,20:F15}");
32          double S2_D = CLS.Sum_of_Number_Series_D(0, Dlt, My_bk, ref kF); SW.WriteLine($"{kF,8}{S2_D,20:F15}");
33        }
34      }
35      static double My_ak(int k)...
39      static double My_bk(int k)...
44    }
45  }
```

Результати нового виконання функції **`Main()`** поточного проекту повинні опинитися в текстовому файлі **`LW_1_1_Test Petrenko_A_P.txt`**, який ми вже приєднали до цього проекту:



Після здійснення розрахунків із «тестовими» сумами (л.1.1.2) і (л.1.1.4) Ви повинні реалізувати обчислення у відповідності із індивідуальним варіантом даної лабораторної роботи.

Індивідуальне завдання:

Побудувати алгоритм та налагодити консольний **`Windows-`**додаток, який призначений для обчислення:

1. суми $S_N = \sum_{k}^{N} \varphi_k$ числового ряду із заданим верхнім значенням N для індексу додавання;

2. суми $S_1 = \sum_{i}^{\infty} \alpha_i$ нескінченого числового ряду с ***абсолютною*** похибкою не гірше ніж $\varepsilon \sim 10^{-9}$.

У зазначеному додатку повинні використовуватися програмні компоненти (класи, методи), які були розроблені та налагоджені Вами під час виконання лабораторної роботи 1.1.

Результати обчислень заносяться в окремий файл, який повинен також містити код основного консольного додатку (для перевірки викладачем).

Числові дані записуються в файл результатів із **десятьма** цифрами після десяткової точки.

Для нескінченої суми S_1 додатково вказується верхнє значення індексу додавання, при якому було зафіксовано представлений результат, тобто обчислене значення суми ряду.

Зразок виконання індивідуального варіанту

Нехай «віртуальний» студент Петренко отримав наступний варіант лабораторної роботи за номером 0:

Сума $S(N)$ із заданою кількістю членів

Варіант	$S(N)$	N
0	$S(N) = \sum_{k=0}^{N} \frac{1}{(2k+1)\cdot(4k+1)\cdot(4k+3)}$	20000

Сума S_1 нескінченного числового ряду

Варіант	$S_1(\infty)$	точне значення
0	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)\cdot(4i+1)\cdot(4i+3)}$	$S_1 = \left(\sqrt{2}-1\right)\cdot\frac{\pi}{4}$

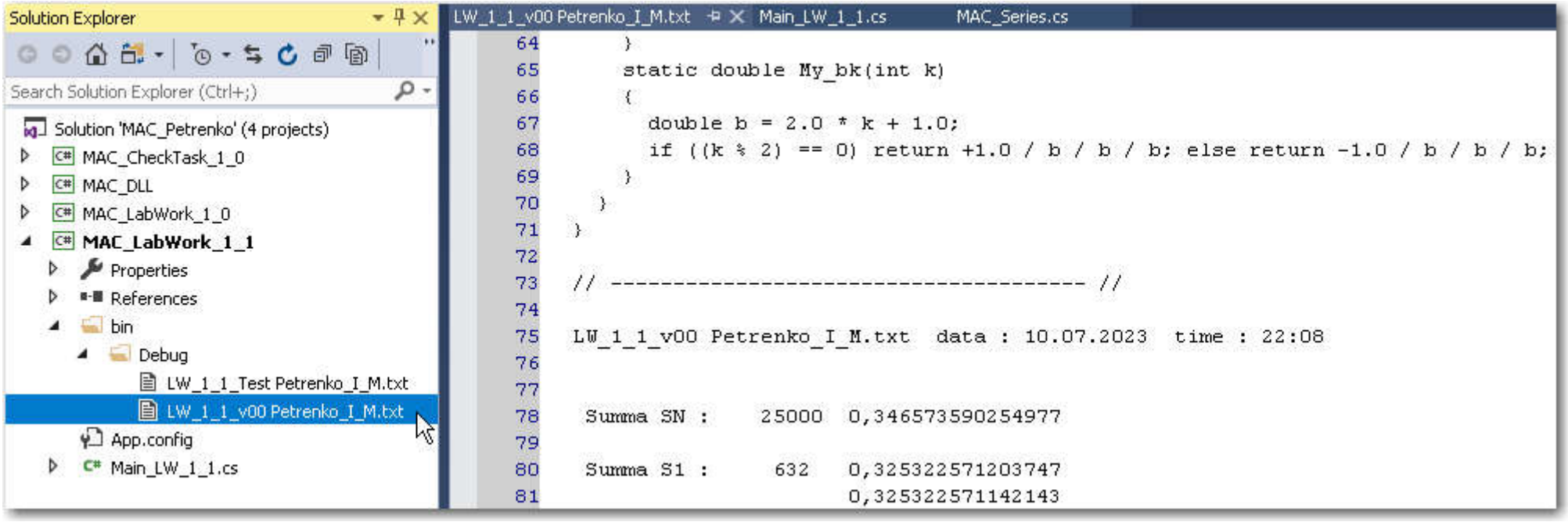
Не треба розробляти окремий проект, а слід скористатися існуючим проектом **MAC_LabWork_1_1**, в який можна внести відповідні зміни, та додати необхідний код.

На наступній сторінці наводиться один із можливих варіантів такого коду з індивідуальним варіантом роботи для «віртуального студента» Петренка.

Обов'язково зверніть увагу на те, яким чином в методі **MyVariant()** реалізовано додавання до файлу результатів того програмного коду, який міститься в файлі **Main_LW_1_1.cs** із кодом консольної програми.

```
6   namespace MAC_LabWork_1_1
7   {
8     class Main_LW_1_1
9     {
10      static int kF = 0; static double Eps = 1.0E-9, Dlt = 1.0E-8;
11
12      static void Main(string[] args)
13      {
14        // CommonTest("LW_1_1_Test", 10000);
15        MyVariant("LW_1_1_v00", 25000);
16      }
17
18      static void MyVariant(string name, int N)
19      {
20        using (StreamWriter SW = UTL.ResultWriter(name, "Main_LW_1_1.cs"))
21        {
22          double SN = CLS.Sum_of_Number_Series(0, N, My_sk);          //--- SN ---//
23          SW.WriteLine($"\r\n Summa SN : {N,8}{SN,19:F15}");
24
25          double True_Sum1 = (Math.Sqrt(2.0) - 1.0) * Math.PI / 4.0; //--- S2 ---//
26          double S1 = CLS.Sum_of_Number_Series_A(0, Eps, My_si, ref kF);
27          SW.WriteLine($"\r\n Summa S1 :{kF,8}{S1,20:F15}\r\n{True_Sum1,39:F15}");
28        }
29      }
30
31      static double My_sk(int k) => 1.0 / (2.0 * k + 1.0) / (4.0 * k + 1.0) / (4.0 * k + 3.0);
32
33      static double My_si(int i)
34      {
35        double si = 1.0 / (2.0 * i + 1.0) / (4.0 * i + 1.0) / (4.0 * i + 3.0);
36        if ((i % 2) == 0) return si; else return -si;
37      }
38      static void CommonTest(string name, int N)...
59      static double My_ak(int k)...
63      static double My_bk(int k)...
68    }
69  }
```

Перегляньте й зразок файлу результатів **LW_1_1_v00 Petrenko_I_M.txt** (який теж слід обов'язково приєднати до даного консольного проекту):



На цьому скріншоті наводиться лише нижня частина файлу результатів, яка містить саме результати розрахунків, але ми також спостерігаємо і код програми, який знаходиться вище.

Саме цей файл слід висилати викладачу (на електронну пошту за його адресою) на перевірку та отримання відповідних залікових балів.

Контрольні дані за варіантами v Сума $S(N)$ із заданою кількістю членів

v	$S(N)$	N	v	$S(N)$	N
1	$S(N)=\sum_{k=1}^{N}\frac{1}{k\cdot(k+4)}$	17400	11	$S(N)=\sum_{k=1}^{N}\frac{1}{k\cdot(2k+1)}$	16300
2	$S(N)=\sum_{k=1}^{N}\frac{1}{k\cdot(3k+2)}$	15100	12	$S(N)=\sum_{k=1}^{N}\frac{1}{k\cdot(4k+3)}$	19500
3	$S(N)=\sum_{k=0}^{N}\frac{1}{(k+1)\cdot(3k+1)}$	16300	13	$S(N)=\sum_{k=0}^{N}\frac{1}{(k+1)\cdot(4k+3)}$	18200
4	$S(N)=\sum_{k=0}^{N}\frac{1}{(k+2)\cdot(2k+1)}$	23100	14	$S(N)=\sum_{k=0}^{N}\frac{1}{(2k+1)\cdot(3k+1)}$	15400
5	$S(N)=\sum_{k=0}^{N}\frac{1}{(2k+1)\cdot(3k+2)}$	19600	15	$S(N)=\sum_{k=0}^{N}\frac{1}{(4k+3)\cdot(2k+1)}$	17600
6	$S(N)=\sum_{k=0}^{N}\frac{1}{(3k+1)\cdot(3k+2)}$	10400	16	$S(N)=\sum_{k=0}^{N}\frac{1}{(4k+3)\cdot(2k+3)}$	21700
7	$S(N)=\sum_{k=0}^{N}\frac{1}{(3k+2)\cdot(3k+4)}$	11700	17	$S(N)=\sum_{k=0}^{N}\frac{1}{(3k+2)\cdot(3k+5)}$	22900
8	$S(N)=\sum_{k=0}^{N}\frac{1}{(4k+1)\cdot(4k+7)}$	12700	18	$S(N)=\sum_{k=0}^{N}\frac{1}{(k+1)\cdot(3k+2)}$	25100
9	$S(N)=\sum_{k=0}^{N}\frac{1}{(4k+3)\cdot(4k+5)}$	18300	19	$S(N)=\sum_{k=0}^{N}\frac{1}{(6k+5)\cdot(6k+7)}$	15800
10	$S(N)=\sum_{k=0}^{N}\frac{1}{(6k+1)\cdot(6k+5)}$	14500	20	$S(N)=\sum_{k=1}^{N}\frac{1}{k\cdot(k+1)\cdot(2k+3)}$	11500

Сума S_1 нескінченного числового ряду

V	$S_1(\infty)$	точне значення
1	$S_1 = \sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i \cdot (i+4)}$	$S_1 = \frac{7}{48}$
2	$S_1 = \sum_{i=1}^{\infty} \frac{(-1)^{i}}{i \cdot (3i+2)}$	$S_1 = \frac{3}{4} \cdot \left(1 - \frac{2\pi}{\sqrt{27}}\right)$
3	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^{i}}{(i+1) \cdot (3i+1)}$	$S_1 = \frac{\pi}{2\sqrt{3}}$
4	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^{i}}{(i+2) \cdot (2i+1)}$	$S_1 = \frac{\pi - 2 + \ln 4}{6}$
5	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^{i}}{(2i+1) \cdot (3i+2)}$	$S_1 = \pi \cdot \frac{3 - 2\sqrt{3}}{6} + \ln 2$
6	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^{i}}{(3i+1) \cdot (3i+2)}$	$S_1 = \frac{2}{3} \ln 2$

7	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(3i+2)\cdot(3i+4)}$	$S_1 = \frac{2\pi\sqrt{3}-9}{18}$
8	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(4i+1)\cdot(4i+7)}$	$S_1 = \frac{3\pi\sqrt{2}-4}{72}$
9	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(4i+3)\cdot(4i+5)}$	$S_1 = \frac{\pi\sqrt{2}-4}{8}$
10	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(6i+1)\cdot(6i+5)}$	$S_1 = \frac{\sqrt{3}\cdot\ln(2+\sqrt{3})}{12}$
11	$S_1 = \sum_{i=1}^{\infty} \frac{(-1)^i}{i\cdot(2i+1)}$	$S_1 = 2 - \frac{\pi}{2} - \ln 2$
12	$S_1 = \sum_{i=1}^{\infty} \frac{(-1)^i}{i\cdot(4i+3)}$	$S_1 = \frac{4}{9} - \frac{\pi}{3\sqrt{2}} - \frac{1}{3}\ln 2 + \frac{\sqrt{2}}{3}\ln(1+\sqrt{2})$
13	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(i+1)\cdot(4i+3)}$	$S_1 = \frac{\pi}{\sqrt{2}} - \ln 2 - \frac{1}{\sqrt{2}}\ln\left(3+2\sqrt{2}\right)$
14	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)\cdot(3i+1)}$	$S_1 = \pi\left(\frac{\sqrt{3}}{3} - \frac{1}{2}\right) + \ln 2$

15	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)\cdot(4i+3)}$	$S_1 = \frac{\pi}{4}\left(1-\sqrt{2}\right) - \frac{\sqrt{2}}{2}\ln(\sqrt{2}-1)$
16	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+3)\cdot(4i+3)}$	$S_1 = \frac{\pi}{12}\left(1+\sqrt{2}\right) - \frac{1}{3} - \frac{1}{3\sqrt{2}}\ln(\sqrt{2}+1)$
17	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(3i+2)\cdot(3i+5)}$	$S_1 = \frac{2}{9}\left(\frac{\pi}{\sqrt{3}} - \ln 2\right) - \frac{1}{6}$
18	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(i+1)\cdot(3i+2)}$	$S_1 = \frac{\pi}{\sqrt{3}} - 2\ln(2)$
19	$S_1 = \sum_{i=0}^{\infty} \frac{(-1)^i}{(6i+5)\cdot(6i+7)}$	$S_1 = \frac{\pi}{6} - \frac{1}{2}$
20	$S_1 = \sum_{i=1}^{\infty} \frac{(-1)^i}{i\cdot(i+1)\cdot(2i+3)}$	$S_1 = \frac{17}{9} - \frac{\pi}{3} - \frac{4}{3}\ln 2$

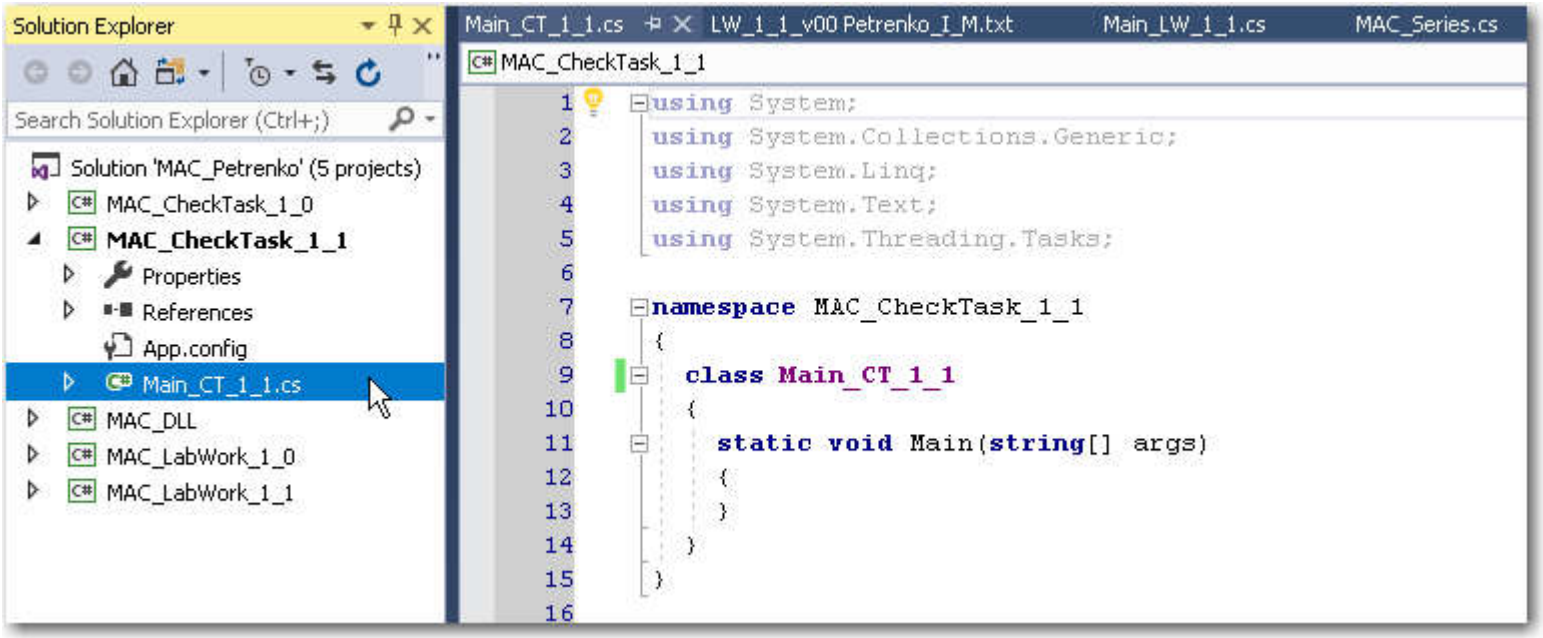
Методика виконання завдання самостійної роботи 1.1
«Обчислення сум числових рядів з абсолютною і відносною похибкою»

Стартуємо середовище розробки **MS Visual Studio**.

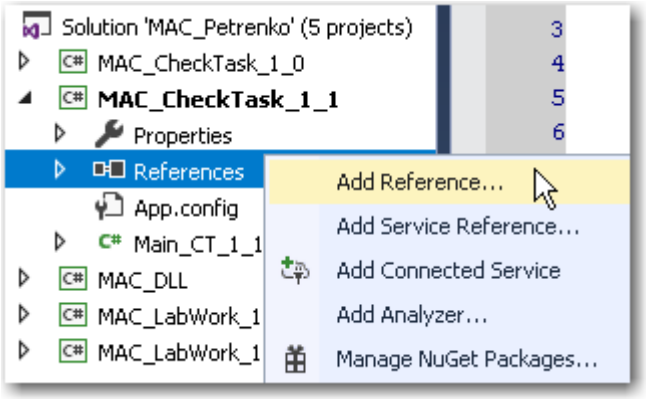
В існуючому робочому просторі **MAC_Petrenko** генеруємо новий проект консольного додатку – **MAC_CheckTask_1_1**.

Обираємо тип конфігурації робочого проекту – **Debug**, і робимо цей проект активним (**Set As StartUp Project**).

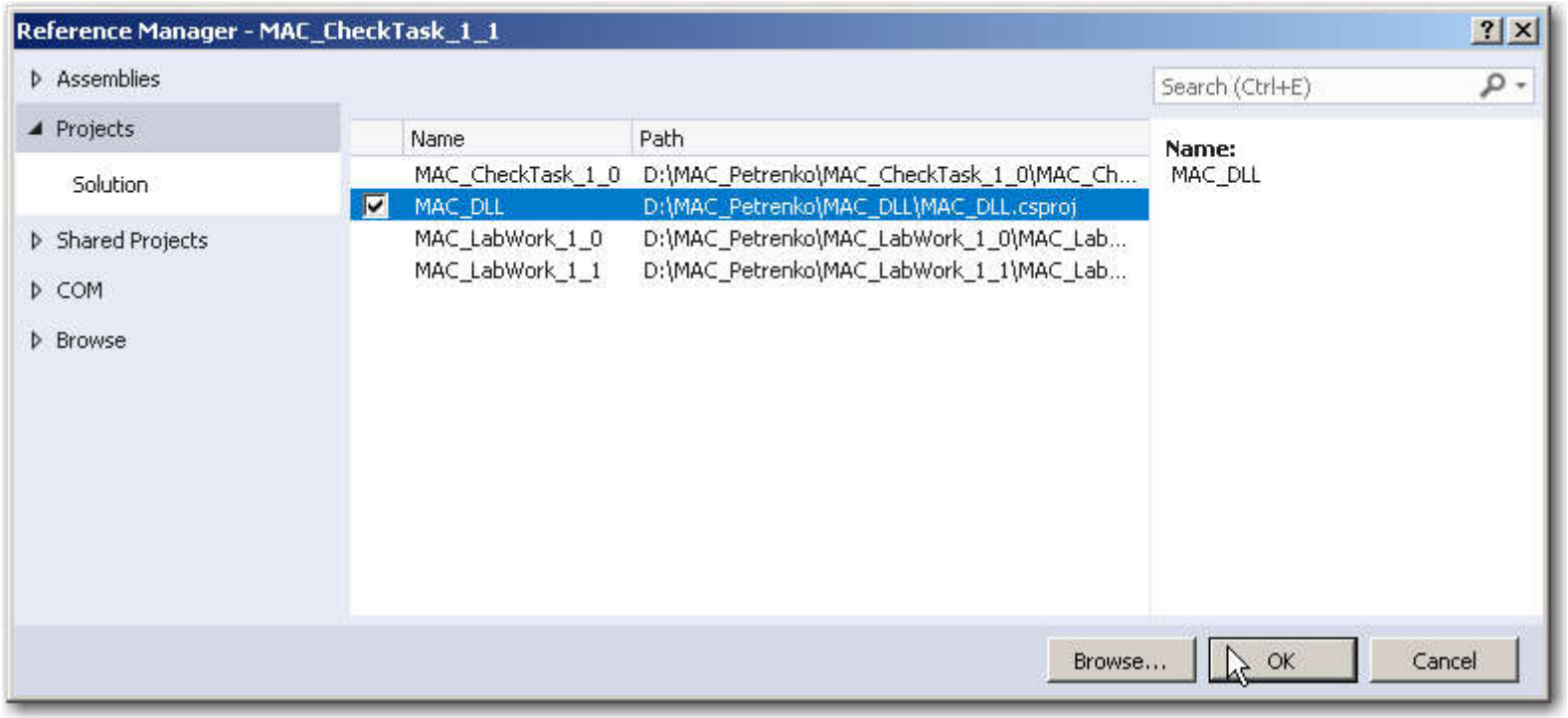
Для зручності одразу перейменуємо клас **Program** в клас **Main_CT_1_1**.



Тепер слід додати бібліотеку класів **MAC_DLL** до переліку посилань **References** проекту **MAC_CheckTask_1_1**:



Це дозволить нам згадувати бібліотеку **MAC_DLL** та її відкриті члени (класи) в директиві **using** у консольному додатку **MAC_CheckTask_1_1**, який розробляється.



Завдання на самостійну роботу:

Створити консольний **`Windows`**–додаток, призначений для обчислення:

1) суми $S_N = \sum_{k}^{N} s_k$ заданого числового ряду;

2) суми $S_1 = \sum_{i}^{\infty} \alpha_i$ числового ряду з абсолютною похибкою $\varepsilon \sim 10^{-9}$;

3) суми $S_2 = \sum_{j}^{\infty} \beta_j$ числового ряду з відносною похибкою $\delta \sim 10^{-8}$.

Додаток треба розробляти в щойно створеному проекті **`MAC_CheckTask_1_1`**.

Обчислення повинні здійснюватися із використанням методів з бібліотеки **`MAC_DLL`**.

Нижче наводиться приклад, який відповідає набору ***даних*** для `0`-го варіанту (студент Петренко):

Шукані суми:

$S_N = \sum_{k=0}^{N} \left(\frac{e^{-k/3}}{k^2 + 9} \cdot \frac{\sqrt{1+\sqrt{k}}}{\pi + k} \right)$	$S_2 = \sum_{j=2}^{\infty} \left(\frac{(-1)^{j+1} \cdot \left(d + \frac{c}{j} \right)^{-\ln 7}}{c\sqrt{d + j^3} + d\sqrt{c + j^5}} \right)$
$S_1 = \sum_{i=1}^{\infty} \left(\frac{\sin(ai + b)}{(3i - \pi + 1)^2} + \frac{\cos(bi - a)}{(i + \pi - 1)^3} \right)$	

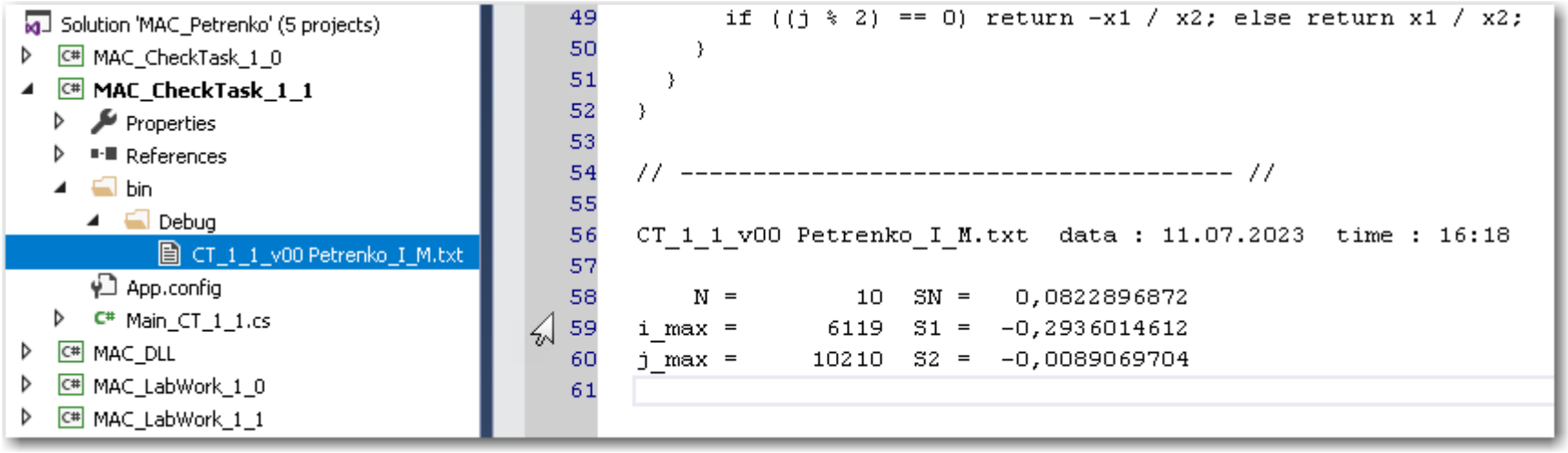
Таблиця ***тестових*** значень параметрів завдання

N	a	b	c	d
10	+0.50	-0.75	+1.12	+1.84

Таблиця ***тестових*** результатів:

$S_N =$	+0.0822896872		
$S_1 =$	-0.2936014612	$i_{max} =$	6119
$S_2 =$	-0.0089069704	$j_{max} =$	10210

Файл результатів (який буде нами отриманий трохи пізніше)



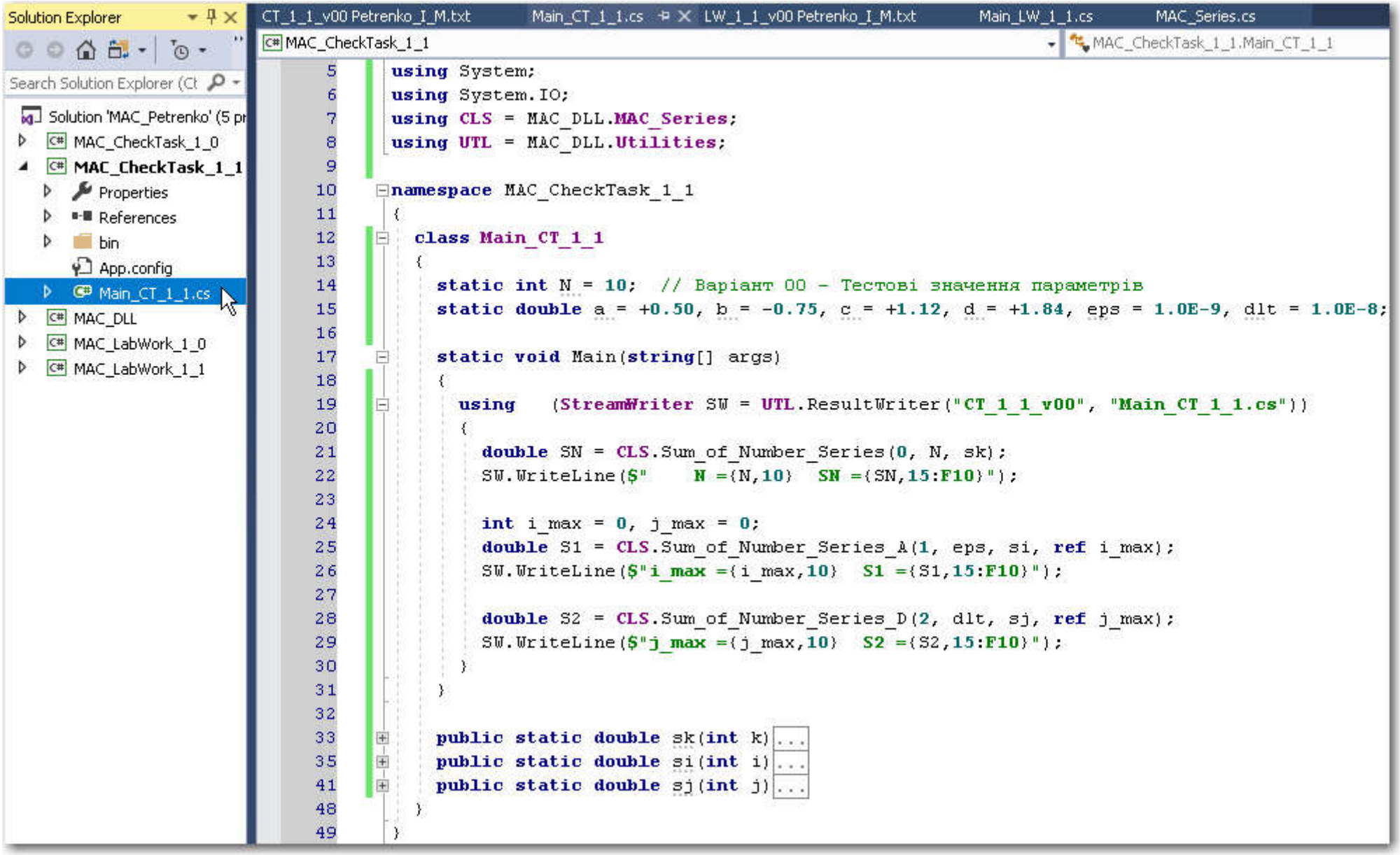
Отже, розглянемо наступні етапи виконання цього завдання на самостійну роботу.

В класі **Main_CT_1_1** консольного додатку **MAC_CheckTask_1_1** створимо коди статичних методів, які будуть реалізовувати обчислення членів заданих числових рядів.

Одразу визначаємо статичні параметри завдання та задаємо їх числові значення (ініціалізація параметрів).

```
10  namespace MAC_CheckTask_1_1
11  {
12    class Main_CT_1_1
13    {
14      static int N = 10;  // Варіант 00 - Тестові значення параметрів
15      static double a = +0.50, b = -0.75, c = +1.12, d = +1.84, eps = 1.0E-9, dlt = 1.0E-8;
16
17      static void Main(string[] args)...
32
33      public static double sk(int k) =>
34        Math.Exp(-k / 3.0) * Math.Sqrt(1.0 + Math.Sqrt(k)) / (9.0 + k * k) / (Math.PI + k);
35
36      public static double si(int i)
37      {
38        double x1 = 3.0 * i - Math.PI + 1.0;       x1 = Math.Sin(a * i + b) / x1 / x1;
39        double x2 = i + Math.PI - 1;               x2 = Math.Cos(b * i - a) / x2 / x2 / x2;
40        return x1 + x2;
41      }
42      public static double sj(int j)
43      {
44        double j3 = 1.0 * j * j * j, j5 = j3 * j * j;
45        double x1 = Math.Pow((d + c / j), -Math.Log(7.0));
46        double x2 = c * Math.Sqrt(d + j3) + d * Math.Sqrt(c + j5);
47        if ((j % 2) == 0) return -x1 / x2; else return x1 / x2;
48      }
49    }
50  }
```

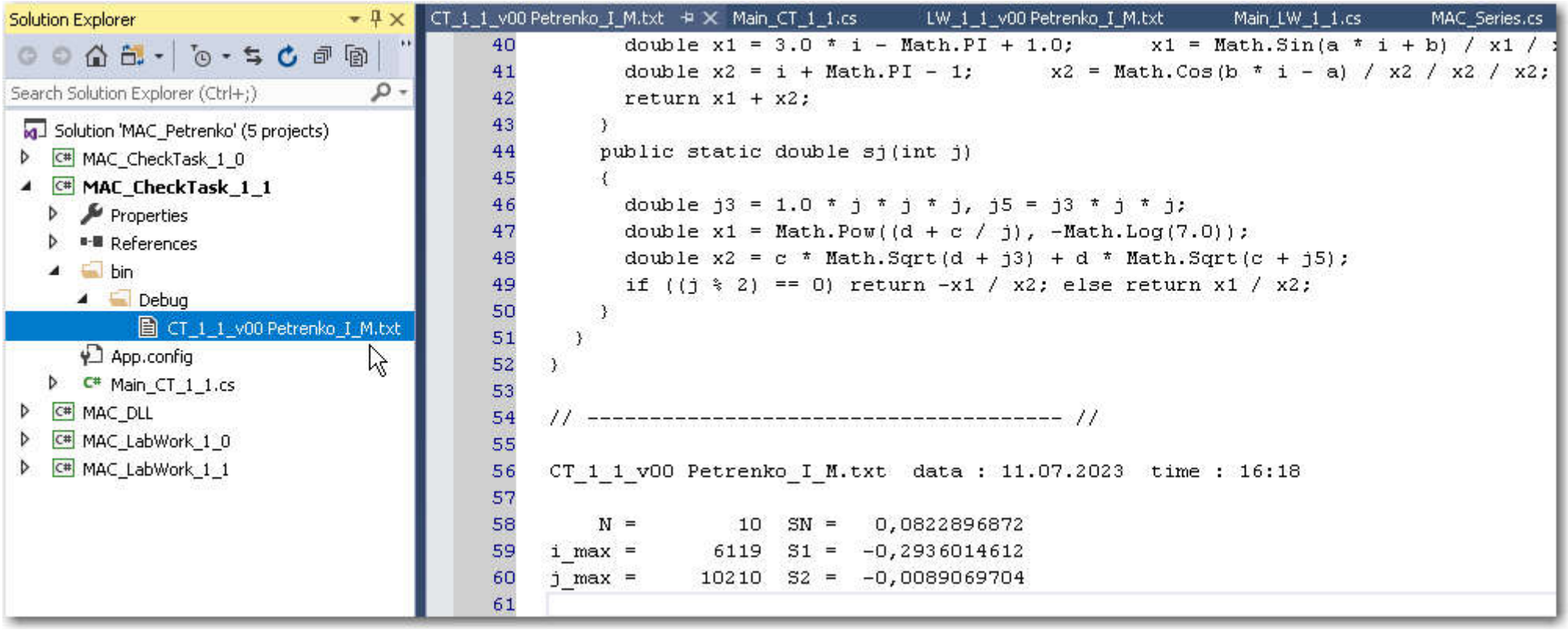
Тепер створюємо код основної функції **`Main()`** консольного додатку:



Нагадуємо Вам, що файл результатів обов’язково повинен містити код **`Main_CT_1_1.cs`** самого додатка.

Після здійснення обчислень треба включити файл результатів до складу даного проекту.

Виконання даної програми дає нам наступні результати:



Ви повинні їх порівняти з тестовими даними (відповідна таблиця була вище за текстом).

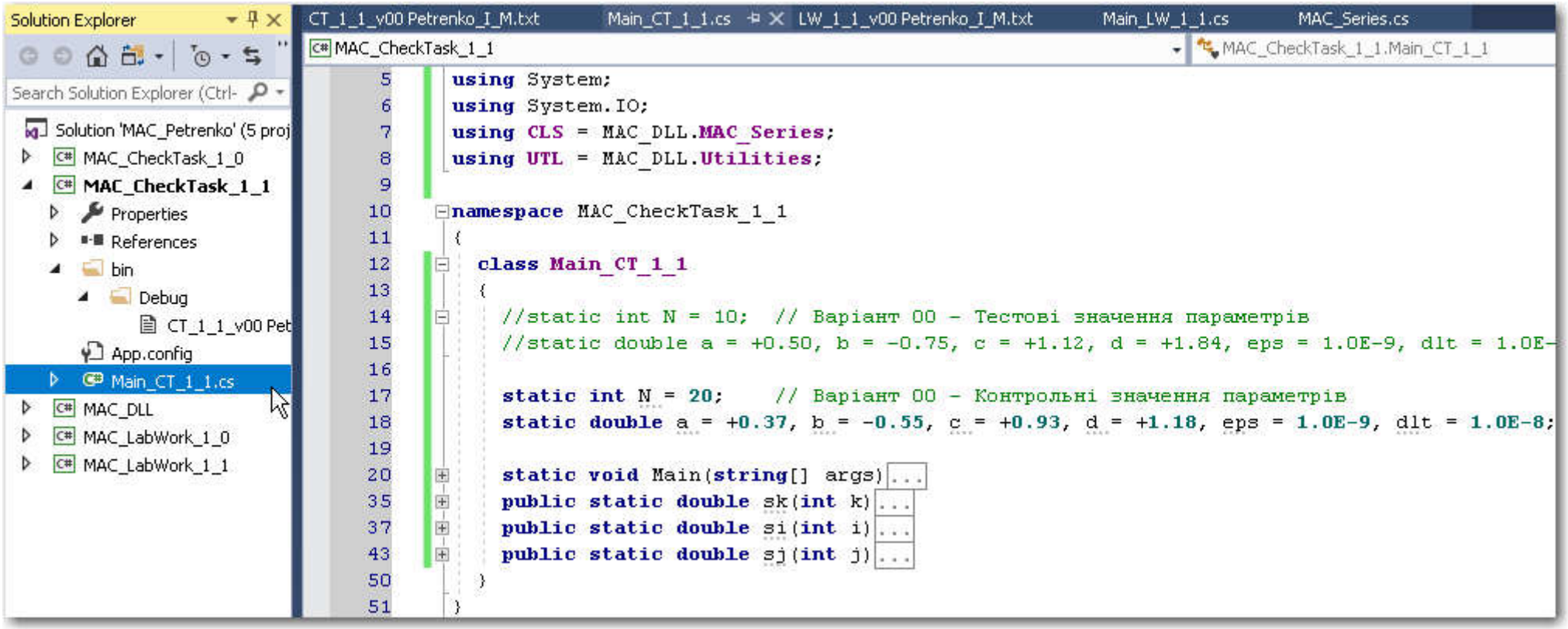
Якщо отримані Вами результати суттєво відрізняються від «тестових» – слід, насамперед, шукати помилки в коді Вашого додатку та перевірити числові значення основних параметрів (ініціалізація статичних змінних).

Якщо ж результати співпадають (із урахуванням правил округлення), Ви можете здійснити обчислення з контрольними значеннями параметрів завдання:

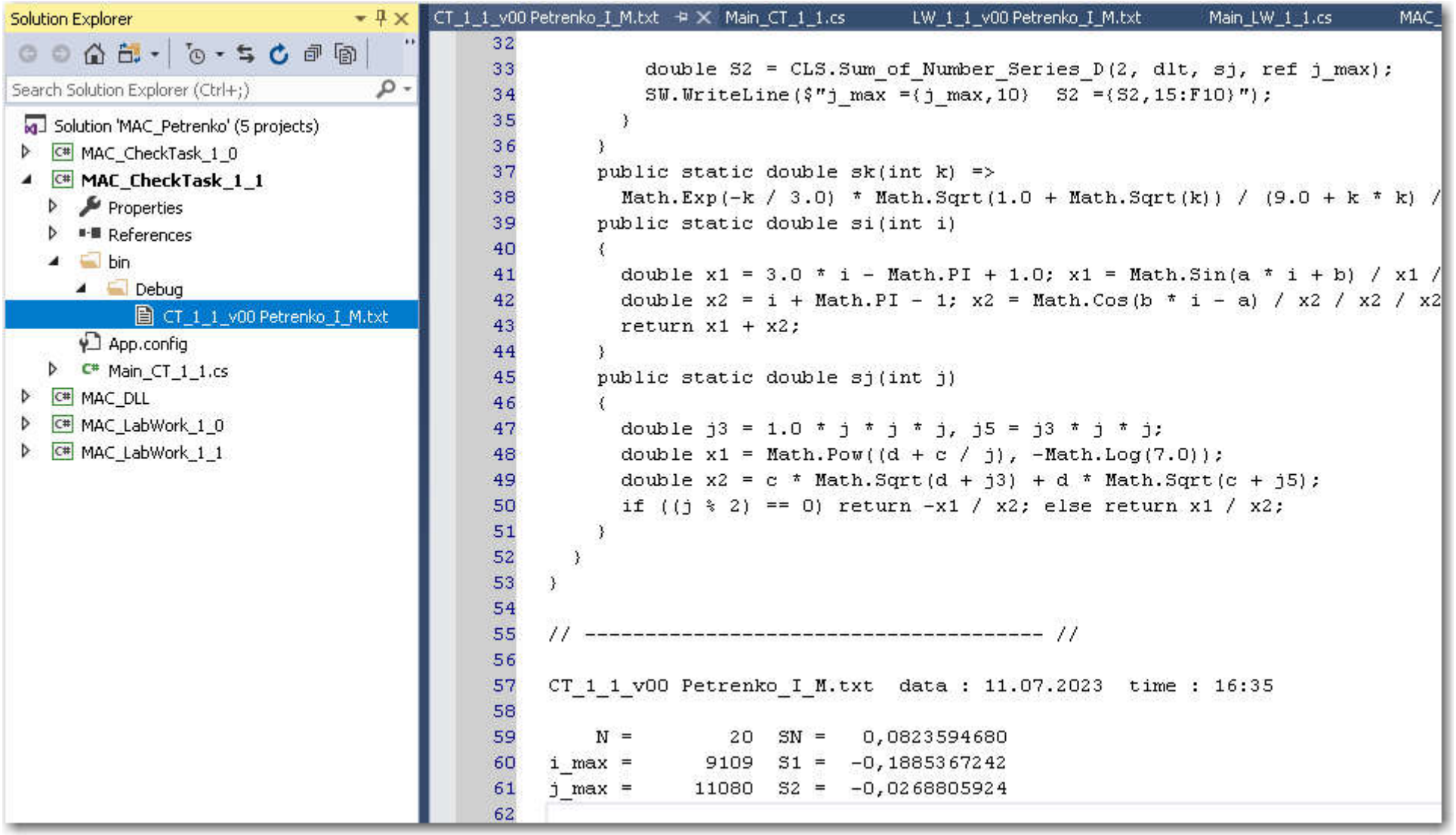
Таблиця *контрольних* значень параметрів завдання:

N	*a*	*b*	*c*	*d*
20	+0.37	-0.55	+0.93	+1.18

Для цього в коді існуючого додатка **Main_CT_1_1** просто слід замінити значення відповідних статичних змінних (параметрів завдання). Наприклад, в такий спосіб:



Файл **CT_1_1_v00 Petrenko_I_M.txt** з отриманими контрольними результатами висилається викладачу для перевірки на адресу його електронної пошти. Нижче наведено такий файл (частково) для нульового варіанту.



Варіант 1

$S_N = \sum_{k=-1}^{N} \frac{\sqrt{k+2}}{(11k-5)\cdot(7k+3)}$	$S_1 = \sum_{i=1}^{\infty} \frac{(a-bi-0.3)^{0.5}}{(2i+a)\cdot(5i+b)^2}$	$S_2 = \sum_{j=0}^{\infty} \frac{(-1)^j}{(3j+c)\cdot(2j+d)}$

Таблиця контрольних значень параметрів завдання:

варіант	N	a	b	c	d
v01.1	52350	+1.37	-2.41	-1.19	+3.05
v01.2	54500	+1.23	-2.87	-1.03	+0.60
v01.3	53150	+1.45	-2.58	-1.14	+2.63

Таблиця ***тестових*** значень параметрів завдання та ***тестових*** результатів обчислень:

N	a	b	c	d
51000	1.50	-2.00	-0.50	1.70

$S_N =$	-0.0245021946		
$S_1 =$	+0.0684380296	$i_{\max} =$	16285
$S_2 =$	-1.2624808397	$j_{\max} =$	7270

Варіант 2

$S_N = \sum_{k=0}^{N} \frac{\sqrt[4]{k+1}}{(3k^2-1)\cdot(\sqrt{k}+3)}$	$S_1 = \sum_{i=-1}^{\infty} \frac{\sqrt{a+i}+\sqrt{i-b}}{(5i+2a)^2\cdot(bi+1)}$	$S_2 = \sum_{j=1}^{\infty} \frac{(-1)^j}{(j^{1.5}+c)\cdot(2j-d+1)}$

Таблиця контрольних значень параметрів завдання:

варіант	N	a	b	c	d
v02.1	12800	+1.77	-2.82	+0.81	+2.42
v02.2	11350	+2.34	-1.57	+2.03	+0.91
v02.3	10950	+2.05	-1.92	+1.64	+1.88

Таблиця ***тестових*** значень параметрів завдання та ***тестових*** результатів обчислень:

N	a	b	c	d
10000	+1.20	-2.10	-0.50	+1.40

$S_N =$	-0.1183500708		
$S_1 =$	+0.4448402709	$i_{\max} =$	18374
$S_2 =$	-1.1572685438	$j_{\max} =$	2269

Варіант 3

$S_N = \sum_{k=2}^{N} \frac{(k^2-3)^{0.5}}{(k^2+1)\cdot(k-1)^2}$	$S_1 = \sum_{i=1}^{\infty}\left(\frac{i^{-1.5}}{2i+3b} + \frac{1}{(i+7)(i^2+a)}\right)$	$S_2 = \sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{(2j+9c)\cdot(cj-d\cdot j^{-2})}$

Таблиця контрольних значень параметрів завдання:

варіант	N	a	b	c	d
v03.1	2700	+2.51	+2.01	+4.46	+2.48
v03.2	1450	+1.74	+2.86	+3.13	+1.52
v03.3	1750	+2.34	+2.67	+3.95	+3.08

Таблиця ***тестових*** значень параметрів завдання та ***тестових*** результатів обчислень:

N	a	b	c	d
1000	+0.50	+1.50	+2.50	+3.50

$S_N =$	+0.3165014445		
$S_1 =$	+0.3872287737	$i_{max} =$	79585
$S_2 =$	-0.0469743374	$j_{max} =$	41259

Варіант 4

$S_N = \sum_{k=1}^{N} \frac{(k+5/k)^{0.25}}{(\sqrt{5k}-3)\cdot(2k-7)}$	$S_1 = \sum_{i=0}^{\infty}\left(\frac{a}{(i^2+9)(i-b)} - \frac{b}{(i+2a)^3}\right)$	$S_2 = \sum_{j=1}^{\infty} \frac{(-1)^{j+1}\cdot(d-2j^{-1})}{\sqrt{j+c}\cdot(j^2+9)}$

Таблиця контрольних значень параметрів завдання:

варіант	N	a	b	c	d
v04.1	26000	+0.85	-2.84	+0.37	-4.17
v04.2	19500	+1.25	-1.93	+1.68	-2.54
v04.3	22500	+1.13	-2.56	+0.84	-3.79

Таблиця ***тестових*** значень параметрів завдання та ***тестових*** результатів обчислень:

N	a	b	c	d
15000	+1.50	-1.50	+1.00	-3.00

$S_N =$	-2.0599960837		
$S_1 =$	+0.3742816489	$i_{max} =$	17314
$S_2 =$	-0.2395462438	$j_{max} =$	8715

Варіант 5

$S_N = \sum_{k=0}^{N} \frac{3k + \sqrt{k+1}}{(k^2 + 3k + 2) \cdot (\sqrt{k} - 2\pi)}$	$S_1 = \sum_{i=1}^{\infty} \frac{\sqrt{ai + b}}{(i + a) \cdot (3i + 1) \cdot (bi + 2)}$	$S_2 = \sum_{j=0}^{\infty} \frac{(-1)^j \sqrt{d + cj^3}}{(j^2 + \sqrt{dj} + cj + 1)^2}$

Таблиця контрольних значень параметрів завдання:

варіант	N	a	b	c	d
v05.1	1400	+2.82	+0.82	+4.24	+2.56
v05.2	1250	+0.91	+2.28	+3.67	+1.65
v05.3	1350	+1.94	+1.85	+3.78	+2.24

Таблиця ***тестових*** значень параметрів завдання та ***тестових*** результатів обчислень:

N	a	b	c	d
1000	+1.50	+0.50	+4.00	+3.00

$S_N =$	-1.3171015096		
$S_1 =$	+0.1367516285	$i_{\max} =$	102759
$S_2 =$	+1.7027013391	$j_{\max} =$	3378

Варіант 6

$S_N = \sum_{k=0}^{N} \left(\frac{k-1}{k^3-2} + \frac{\sqrt{k+1}}{k^2+4} \right)$	$S_1 = \sum_{i=1}^{\infty} \left(\frac{\sqrt{2i-a}}{(i+\sqrt{ai}+b)^2} \cdot \frac{\sqrt{2i-b}}{i^2+a} \right)$	$S_2 = \sum_{j=0}^{\infty} \left(\frac{(-1)^{j+1}}{cj^2+d\sqrt{j}+7} \cdot \frac{d}{\sqrt{dj+11}} \right)$

Таблиця контрольних значень параметрів завдання:

варіант	N	a	b	c	d
v06.1	1600	+0.16	-0.32	+0.26	+2.24
v06.2	1350	+1.21	-0.77	+0.85	+3.69
v06.3	1250	+0.75	-0.26	+0.51	+3.13

Таблиця ***тестових*** значень параметрів завдання та ***тестових*** результатів обчислень:

N	a	b	c	d
1000	+1.00	-0.10	+0.40	+1.00

$S_N =$	+2.8830721450		
$S_1 =$	+0.3282050320	$i_{\max} =$	14855
$S_2 =$	-0.0241886739	$j_{\max} =$	20264

Варіант 7

$S_N = \sum_{k=0}^{N} \left(\frac{k^{1.5}+3}{k^{2.5}-9} - \frac{\sqrt{k+2}}{5k^2+4} \right)$	$S_1 = \sum_{i=0}^{\infty} \left(\frac{(i+a)^{-0.25}}{i^{2.5}+bi+a} - \frac{b}{(ai+b)^3} \right)$	$S_2 = \sum_{j=1}^{\infty} \left(\frac{(-1)^{j+1}}{(d+1)j+c} \cdot \frac{d+\sqrt{j}}{(cj-0.5)^{1.5}} \right)$

Таблиця контрольних значень параметрів завдання:

варіант	N	a	b	c	d
v07.1	23000	+1.13	+1.07	+2.91	-0.71
v07.2	18200	+2.26	+1.53	+3.74	-0.36
v07.3	21800	+1.85	+1.71	+3.46	-0.59

Таблиця ***тестових*** значень параметрів завдання та ***тестових*** результатів обчислень:

N	a	b	c	d
10000	+1.50	+2.00	+2.50	-1.00

$S_N =$	+6.2678845027		
$S_1 =$	+0.5923378257	$i_{\max} =$	30819
$S_2 =$	-0.0088688722	$j_{\max} =$	33641

Варіант 8

$S_N = \sum_{k=1}^{N} \left(\frac{\sqrt[4]{k+3}}{k^{5/2}+6} - \frac{\sqrt{k}+\sqrt[3]{\pi}}{7k^{3/2}+2k} \right)$	$S_1 = \sum_{i=0}^{\infty} \left(\frac{(bi+a)^{0.25}}{(i^2+b)(i+a)} - \frac{a+b\sqrt{i}}{(ai^2+b)^2} \right)$	$S_2 = \sum_{j=1}^{\infty} \left(\frac{(-1)^j}{c \cdot \sqrt[2]{j} + d \cdot \sqrt[3]{j}} \cdot \frac{j+c}{(dj+1)^{\pi}} \right)$

Таблиця контрольних значень параметрів завдання:

варіант	N	a	b	c	d
v08.1	21000	+0.76	+1.17	+5.00	+1.34
v08.2	13400	+1.45	+0.82	+3.96	+1.77
v08.3	19100	+1.22	+0.95	+4.53	+1.46

Таблиця ***тестових*** значень параметрів завдання та ***тестових*** результатів обчислень:

N	a	b	c	d
18000	+0.80	+1.00	+2.00	+1.00

$S_N =$	-1.2842212941		
$S_1 =$	+0.1370239682	$i_{\max} =$	31822
$S_2 =$	-0.0910373125	$j_{\max} =$	3871

Варіант 9

$S_N = \sum_{k=0}^{N} \left(\frac{(k+1)^{-1.5}}{5k^2 + \ln(\pi + k)} - \frac{\pi}{2 + k^3} \right)$	$S_1 = \sum_{i=2}^{\infty} \left(\frac{\cos(ai + b)}{(2i - \pi)^2} - \frac{\sin(bi - a)}{(i + \pi)^3} \right)$	$S_2 = \sum_{j=1}^{\infty} \left(\frac{(-1)^{j+1} \cdot (dj + c)^{-\pi/2}}{c\sqrt{1 + j} + j\sqrt{d + 5}} \right)$

Таблиця контрольних значень параметрів завдання:

варіант	N	a	b	c	d
v09.1	21000	-0.23	-2.36	+0.44	+0.69
v09.2	17300	-1.28	-1.45	+2.63	+1.37
v09.3	18200	-0.54	-1.97	+1.82	+1.31

Таблиця ***тестових*** значень параметрів завдання та ***тестових*** результатів обчислень:

N	a	b	c	d
15000	-0.70	-2.50	+1.00	+1.50

$S_N =$	-13.2169584290		
$S_1 =$	-0.9524433465	$i_{\max} =$	7524
$S_2 =$	+0.0479576160	$j_{\max} =$	4561

Варіант 10

$S_N = \sum_{k=0}^{N} \left(\frac{\ln(k^2+\pi)}{k+2} \cdot \frac{\sqrt{2k+1}}{k+\pi^2} \right)$	$S_1 = \sum_{i=1}^{\infty} \left(\frac{\sin(ai+b)}{(3i-\pi+1)^2} + \frac{\cos(bi-a)}{(i+\pi-1)^3} \right)$	$S_2 = \sum_{j=2}^{\infty} \left(\frac{(-1)^j \cdot (cj-d)^{-\ln 5}}{d\sqrt{c+j} + j\sqrt{d+\pi}} \right)$

Таблиця контрольних значень параметрів завдання:

варіант	N	a	b	c	d
v10.1	1250	+0.75	+1.08	+1.47	+0.31
v10.2	1190	+1.48	+0.73	+1.82	+1.19
v10.3	1280	+1.54	+1.07	+1.26	+0.89

Таблиця ***тестових*** значень параметрів завдання та ***тестових*** результатів обчислень:

N	a	b	c	d
1000	+1.50	+0.50	+1.00	+1.50

$S_N =$	+5.7293678410		
$S_1 =$	+1.2362987982	$i_{\max} =$	4871
$S_2 =$	+0.3999099155	$j_{\max} =$	2448

Лабораторна робота 1.2

«Алгоритми обчислення значень степеневих рядів (спецфункцій)»

В лабораторній роботі 1.1 ми вивчили та реалізували алгоритми обчислення кінцевих $S = \sum^{N} a_k$ та нескінченних $S = \sum^{\infty} a_k$ сум для числових рядів.

В багатьох задачах прикладної математики і фізики для обчислень значень вищих трансцендентних функцій доводиться користуватися їх зображеннями в формі нескінченних степеневих рядів, наприклад, $f(x) = \sum_{k=k_o}^{\infty} \alpha_k x^k$, де $\alpha_k = \alpha_k(k)$ – деякі числові коефіцієнти, k_o – початкове значення індексу додавання.

Вочевидь, що при безпосередніх комп'ютерних обчисленнях кількість K членів степеневого ряду, які дійсно беруть участь у додаванні, обмежена та обирається програмним шляхом з урахуванням вимог до точності кінцевого результату, тобто ми будемо мати наступне підсумкове наближене значення $f(x) \approx \sum_{k=k_o}^{K} \alpha_k x^k$.

Якщо ж нам задано абсолютну похибку обчислень ε, то в якості критерію вибору найбільшого значення K, зокрема, можна прийняти наступну умову $\left|\alpha_{K+1} x^{K+1}\right| < \varepsilon$.

Однак виконання цієї нерівності, або будь-якої іншої аналогічної умови, аж ніяк не гарантує нам остаточну якість результату, і потребує додаткових тестових обчислень для з'ясування границь застосування розробленого алгоритму.

Далі продемонструємо сказане на прикладі достатньо простих трансцендентних функцій: $\sin(x)$, $\cos(x)$, e^x, $\sinh(x)$ і $\cosh(x)$, для яких відомі наступні зображення в виді степеневих рядів:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots, \qquad \cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots, \tag{л.1.2.1}$$

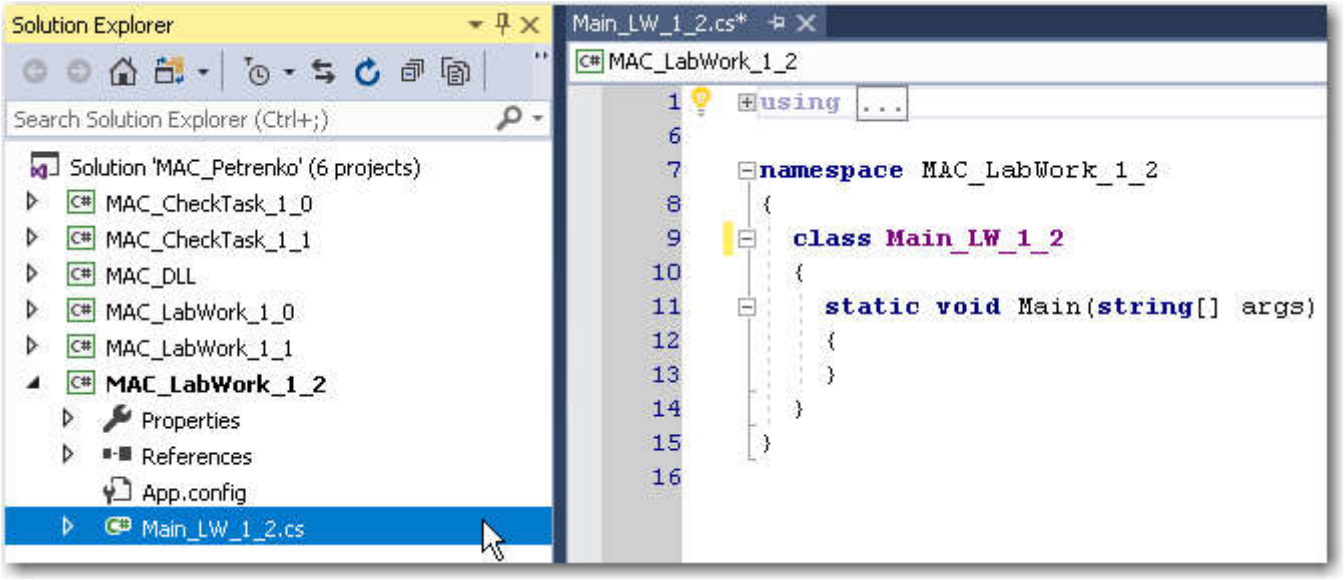
$$e^x = \exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots, \tag{л.1.2.2}$$

$$\sinh(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots, \qquad \cosh(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots \tag{л.1.2.3}$$

Одночасно ми розберемо «технологію» складання алгоритму обчислення значень для степеневих рядів (степеневих зображень трансцендентних функцій). Ця технологія Вам також знадобиться при виконанні завдання на самостійну роботу.

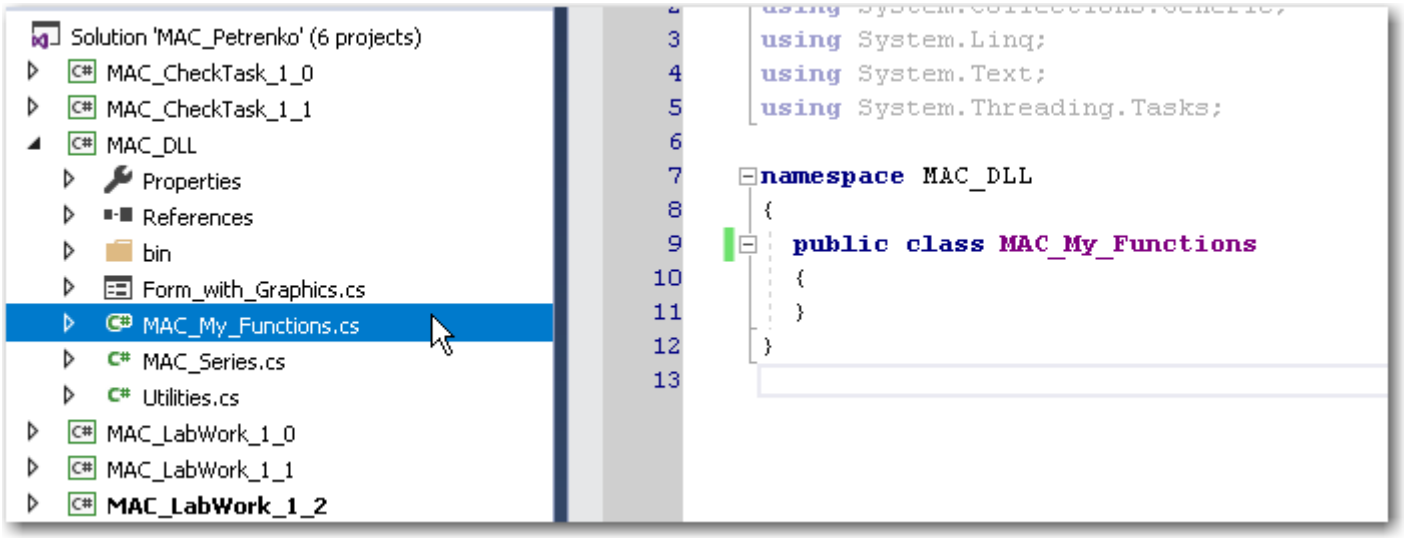
Отже, в робочому просторі **`MAC_Petrenko`** генеруємо новий проект **`MAC_LabWork_1_2`** консольного додатку.

Задаємо його основному класу нове м.'я – **`Main_LW_1_2`**.



Розробку алгоритмів обчислення функцій (л.1.2.1)–(л.1.2.3) будемо виконувати в новому окремому класі **MAC_My_Functions** бібліотеки класів **MAC_DLL** (проект **MAC_DLL**).

Додамо новий клас **MAC_My_Functions** в бібліотеку **MAC_DLL**:



Стандартним шляхом додаємо бібліотеку **MAC_DLL** до переліку **References** нового проекту **MAC_LabWork_1_2**.

Це дозволить нам вказувати бібліотеку **MAC_DLL** в директиві **using** консольного додатку **MAC_LabWork_1_2**, а також визивати її різні методи.

Перш ніж безпосередньо програмувати алгоритм обчислення тригонометричного синуса, запишемо вираз для загального члена ряду (л.1.2.1) та умовимося про нумерацію послідовності членів цього степеневого ряду:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + (-1)^{k+1}\frac{x^{2k-1}}{(2k-1)!} + \ldots = \sum_{k=1}^{\infty}(-1)^{k+1}\frac{x^{2k-1}}{(2k-1)!}. \qquad \text{(л.1.2.4)}$$

В зображенні (л.1.2.4) мається на увазі, що індекс k має інкремент, який дорівнює 1.

Тепер запишемо аналогічне зображення для тригонометричного косинуса:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots + (-1)^k \frac{x^{2k}}{(2k)!} + \ldots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}. \qquad \text{(л.1.2.5)}$$

Для зручності обчислення членів степеневих рядів (л.1.2.4) і (л.1.2.5) можна побудувати відповідні рекурентні формули.

Якщо для (л.1.2.4) ввести позначення:

$$p_1 = x, \quad p_2 = -\frac{x^3}{3!}, \quad p_3 = \frac{x^5}{5!}, \quad p_4 = -\frac{x^7}{7!}, \quad \ldots, \quad p_k = (-1)^{k+1} \frac{x^{2k-1}}{(2k-1)!},$$

то рекурентна послідовність буде наступною:

$$p_1 = x, \qquad p_k = -p_{k-1} \times \frac{x^2}{(2k-2)\cdot(2k-1)}, \qquad k > 1. \qquad \text{(л.1.2.6)}$$

Перевага такого способу обчислення значень для членів $(-1)^{k+1} \frac{x^{2k-1}}{(2k-1)!}$ степеневого ряду (л.1.2.4) безумовна.

По-перше, ми уникаємо явного зведення аргументу x у великі степені $(2k-1)$.

По-друге, нам не треба явним чином обчислювати значення факторіала для великих значень індексу k.

Крім цього, якщо ввести позначення $\tilde{x} = x / 2$, то можна ще підвищити якість обчислень за алгоритмом (л.1.2.6):

$$p_1 = x, \qquad p_k = -p_{k-1} \times \left(\frac{\tilde{x}}{k-1.0}\right) \times \left(\frac{\tilde{x}}{k-0.5}\right), \qquad k > 1. \qquad \text{(л.1.2.7)}$$

Побудуємо тепер аналогічні рекурентні формули для тригонометричного косинуса.

З формули (л.1.2.5) маємо:

$$p_0 = 1, \quad p_1 = -\frac{x^2}{2!}, \quad p_2 = \frac{x^4}{4!}, \quad p_3 = -\frac{x^6}{6!}, \quad \ldots, \quad p_k = (-1)^k \frac{x^{2k}}{(2k)!}.$$

Тоді рекурентна послідовність для функції $\cos(x)$ буде наступною:

$$p_0 = 1.0, \quad p_k = -p_{k-1} \times \frac{x^2}{(2k-1)\cdot(2k)}, \quad k > 0, \qquad \text{(л.1.2.8)}$$

або із використанням $\tilde{x} = x / 2$ $\quad p_0 = 1.0, \quad p_k = -p_{k-1} \times \left(\frac{\tilde{x}}{k-0.5}\right) \times \left(\frac{\tilde{x}}{k}\right), \quad k > 0.$ (л.1.2.9)

Таким чином, для обчислення тригонометричного синуса і косинуса у нас підготовлені наступні алгоритми:

$\sin(0) = 0$ або $\sin(x) = p_1 + \sum_{k=2}^{K} p_k$, де для значення індексу K виконується умова $|p_K| < \varepsilon$, (л.1.2.10)

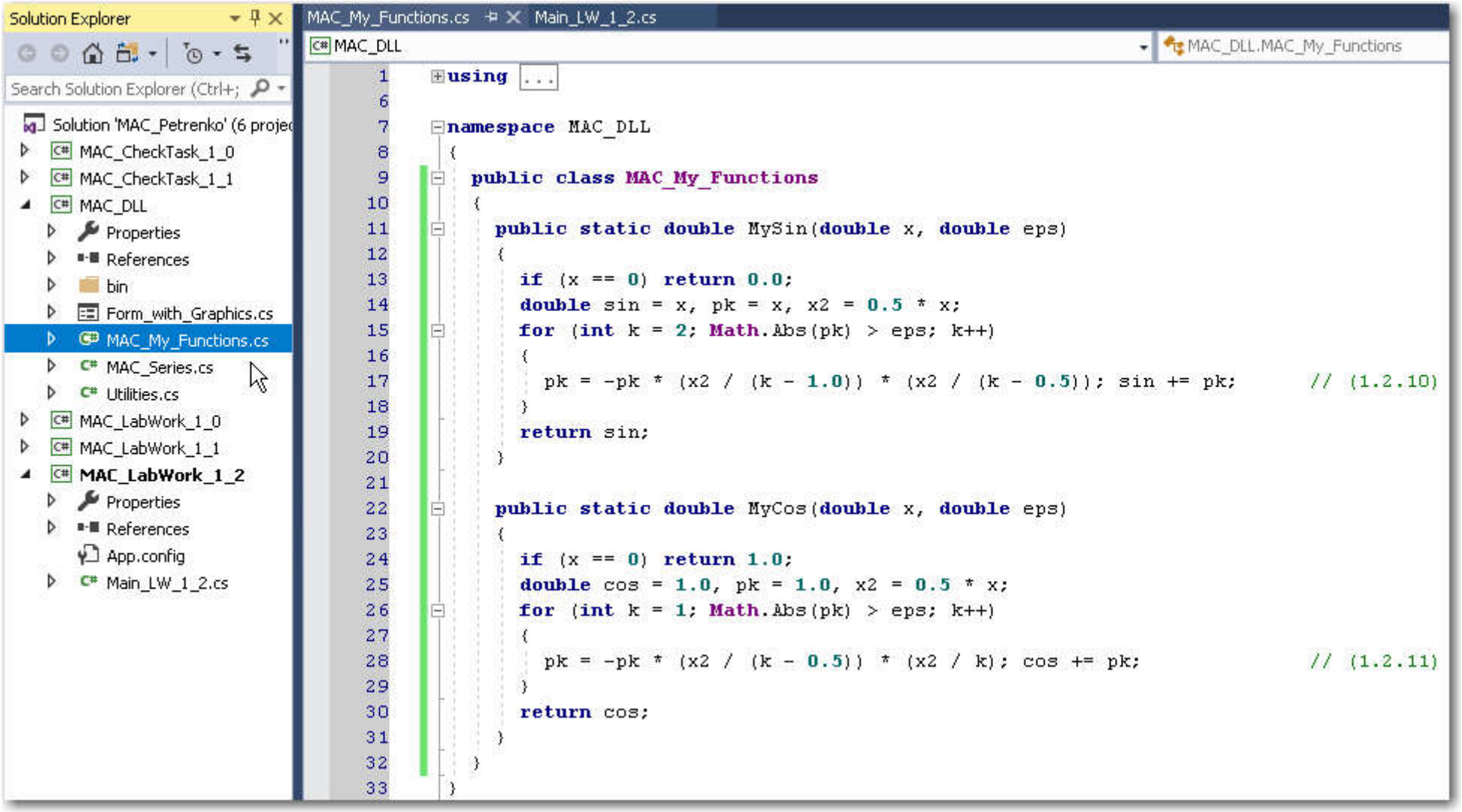
та $\quad p_1 = x, \quad p_k = -p_{k-1} \times \left(\frac{\tilde{x}}{k-1.0}\right) \times \left(\frac{\tilde{x}}{k-0.5}\right), \quad k > 1, \quad \tilde{x} = x / 2.$

$\cos(0) = 1$ або $\cos(x) = p_0 + \sum_{k=1}^{K} p_k$, де для значення індексу K виконується умова $|p_K| < \varepsilon$, (л.1.2.11)

та $\quad p_0 = 1.0, \quad p_k = -p_{k-1} \times \left(\frac{\tilde{x}}{k-0.5}\right) \times \left(\frac{\tilde{x}}{k}\right), \quad k > 0, \quad \tilde{x} = x / 2.$

Абсолютна похибка є обчислень буде другою змінною у всіх функціях (методах), які ми будемо розробляти та розміщувати в класі **MAC_My_Functions**. Таким чином (якщо виникне така потреба) ми зможемо змінити це значення.

Все вище сказане реалізуємо наступним чином:



Тепер нам слід протестувати розроблені функції **`MySin()`** і **`MyCos()`** із використанням відомої Вам тотожності:

$$\forall x \in (-\infty, +\infty) \text{ має місце } \cos^2(x) + \sin^2(x) \equiv 1. \qquad \text{(л.1.2.12)}$$

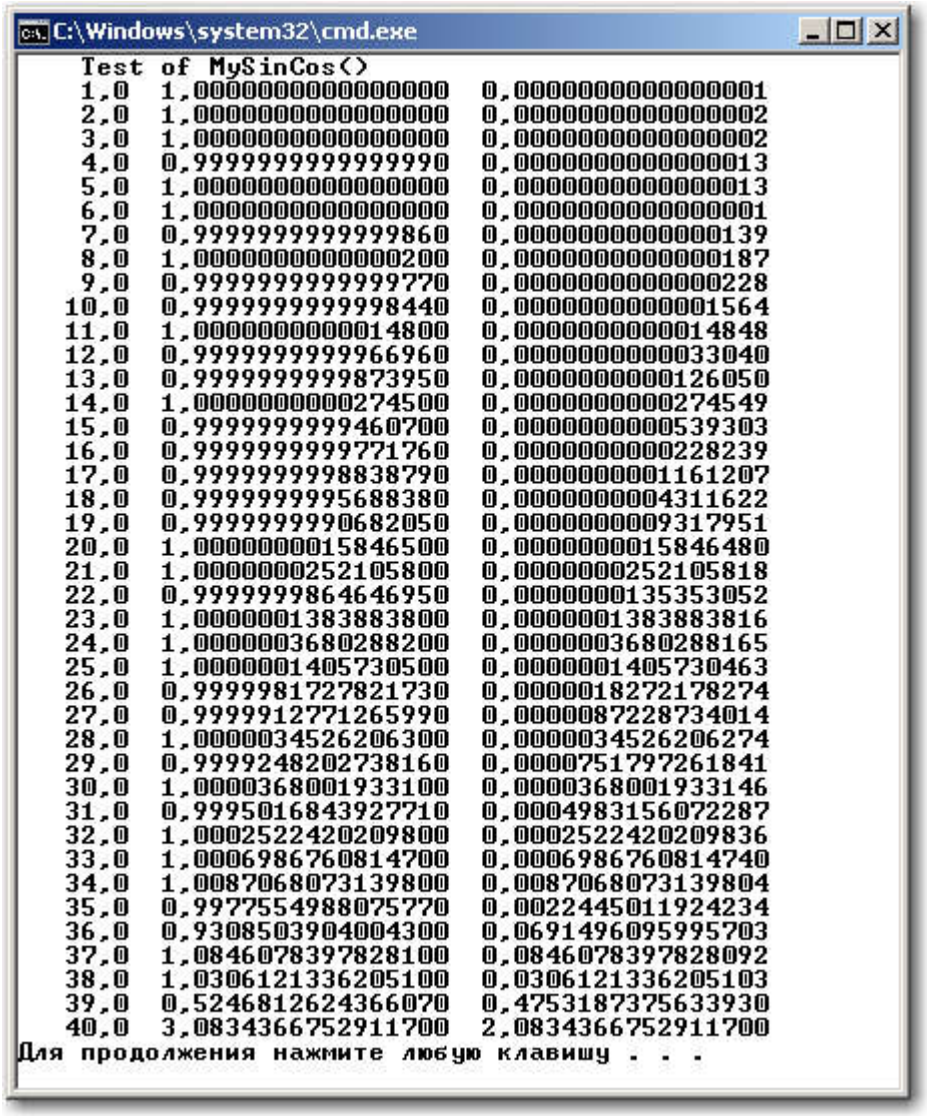
З цією метою створимо окремий допоміжний метод – функцію **`TestMySinCos()`** в класі **`Main_LW_1_2`**:

```
8   using MyF = MAC_DLL.MAC_My_Functions;
9
10  namespace MAC_LabWork_1_2
11  {
12      class Main_LW_1_2
13      {
14          static void Main(string[] args) { Console.Write(TestMySinCos()); }
15
16          static string TestMySinCos()
17          {
18              double unit, error, e = 1.0E-20;
19              string txt = "     Test of MySinCos() \r\n";
20              for (double x = 1.0; x <= 40.0; x += 1.0)
21              {
22                  unit = MyF.MyCos(x, e) * MyF.MyCos(x, e) + MyF.MySin(x, e) * MyF.MySin(x, e);  // (1.2.12)
23                  error = Math.Abs(1.0 - unit);
24                  txt += $"{x,7:F1}{unit,20:F16}{error,20:F16}\r\n";
25              }
26              return txt;
27          }
28      }
29  }
```

Ця функція повертає таблицю значень виразу (л.1.2.12) для послідовності значень аргументу $x = 1, 2, \ldots, 40$.

Крім цього, визначається помилка обчислень – змінна **`error`**, яка характеризує абсолютну похибку обчислень значення **`unit`** у порівнянні з її «еталонним» значенням 1.0.

Маємо наступні результати:



Таблиця, яку наведено вище, яскраво демонструє область застосування розроблених алгоритмів для тригонометричних функцій $\sin(x)$ і $\cos(x)$, коли вони обчислюються в вигляді степеневих рядів.

При зростанні абсолютного значення аргументу x похибка обчислень швидко зростає і при $x \approx 20$ складає $\sim 10^{-9}$.

І це не вважаючи на те, що в методи **`MySin()`** і **`MyCos()`** передавалося значення $\varepsilon = 10^{-20}$.

При значеннях аргументу $x > 30$ результати обчислень можна взагалі вважати неприйнятними з математичної точки зору.

Аналіз даної ситуації та виявлення її причин ми проведемо дедалі пізніше.

За аналогією з тригонометричними функціями, виконаємо алгоритмізацію гіперболічних функцій $\sinh(x)$ і $\cosh(x)$, а також експоненціальної функції e^x.

Що стосується гіперболічних функцій $\sinh(x)$ і $\cosh(x)$ – то тут все очевидно, так як їх зображення в виді степеневих рядів схожі з відповідними зображеннями для тригонометричних функцій $\sin(x)$ і $\cos(x)$, з тією лише різницею, що члени цих рядів мають сталий додатний знак.

Тому алгоритми для $\sinh(x)$ і $\cosh(x)$ отримуємо одразу з (л.1.2.10)–(л.1.2.11):

$$\sinh(0) = 0, \qquad \sinh(x) = p_1 + \sum_{k=2}^{K} p_k, \quad \text{де для індексу } K \text{ виконується умова } |p_K| < \varepsilon, \tag{л.1.2.13}$$

$$\text{і} \qquad p_1 = x, \quad p_k = p_{k-1} \times \left(\frac{\tilde{x}}{k - 1.0}\right) \times \left(\frac{\tilde{x}}{k - 0.5}\right), \quad k > 1, \ \tilde{x} = x / 2 .$$

$$\cosh(0) = 1, \qquad \cosh(x) = p_0 + \sum_{k=1}^{K} p_k, \quad \text{де для індексу } K \text{ виконується умова } |p_K| < \varepsilon, \tag{л.1.2.14}$$

$$\text{і} \qquad p_0 = 1.0, \quad p_k = p_{k-1} \times \left(\frac{\tilde{x}}{k - 0.5}\right) \times \left(\frac{\tilde{x}}{k}\right), \quad k > 0, \ \tilde{x} = x / 2.$$

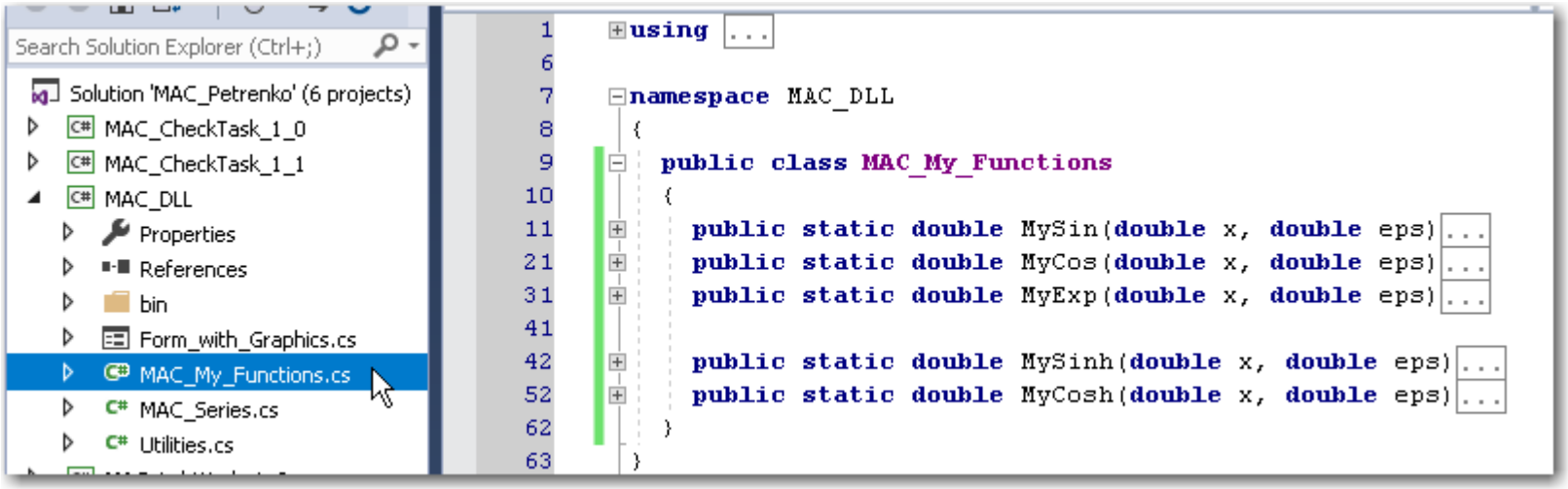
Алгоритм для експоненціальної функції буде наступним:

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^k}{k!} + \ldots = 1 + \sum_{k=1}^{\infty} \frac{x^k}{k!},$$ звідки

$$\exp(0) = 1, \qquad \exp(x) = p_0 + \sum_{k=1}^{K} p_k,$$ де для індексу K виконується умова $|p_K| < \varepsilon,$ (л.1.2.15)

$$\text{і} \quad p_0 = 1.0, \quad p_k = p_{k-1} \times \frac{x}{k}, \quad k > 0.$$

Самостійно доповніть клас **MAC_My_Functions** бібліотеки **MAC_DLL** відповідними методами (функціями) **MySinh()**, **MyCosh()** і **MyExp()**:



Для тестування алгоритмів (1.2.13)–(1.2.14) гіперболічних функцій $\sinh(x)$ і $\cosh(x)$ скористуємося відомою тотожністю:

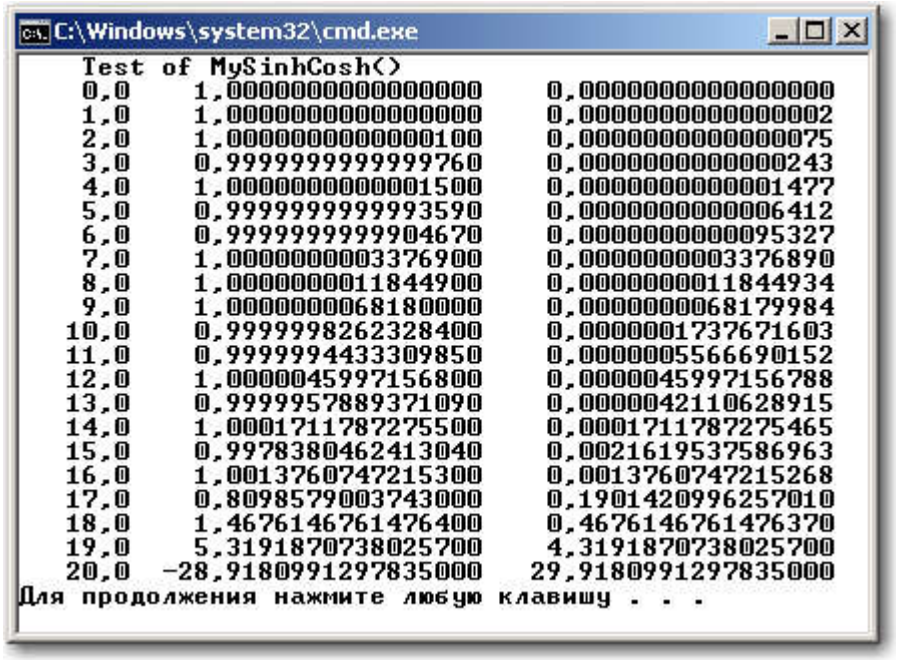
$$\forall x \in (-\infty, +\infty) \text{ має місце } \cosh^2(x) - \sinh^2(x) \equiv 1.$$ (л.1.2.16)

За аналогією із попередньою тестовою функцією **`TestMySinCos()`** основного додатку **`MAC_LabWork_1_2`** створимо окрему функцію **`TestMySinhCosh()`**, виклик якої будемо здійснювати в основній функції **`Main()`**:

```
8   using MyF = MAC_DLL.MAC_My_Functions;
9
10  namespace MAC_LabWork_1_2
11  {
12      class Main_LW_1_2
13      {
14          static void Main(string[] args)
15          {
16              Console.Write(TestMySinhCosh()); // Console.Write(TestMySinCos());
17          }
18          static string TestMySinhCosh()
19          {
20              string txt = "     Test of MySinhCosh() \r\n";
21              double unit, error, e = 1.0E-20;
22              for (double x = 0.0; x <= 20.0; x += 1.0)
23              {
24                  unit = (MyF.MyCosh(x, e) - MyF.MySinh(x, e)) * (MyF.MyCosh(x, e) + MyF.MySinh(x, e)); // (1.2.16)
25                  error = Math.Abs(1.0 - unit);
26                  txt += $"{x,7:F1}{unit,22:F16}{error,22:F16}\r\n";
27              }
28              return txt;
29          }
30          static string TestMySinCos()...
42      }
43  }
```

Функція **`TestMySinhCosh()`** будує таблицю значений виразу (л.1.2.16) для послідовності значень аргументу $x = 1, 2, \ldots, 20$. Тут також визначається помилка обчислень – змінна **`error`**, яка характеризує абсолютну похибку обчислень значення **`unit`** у порівнянні з її «еталонним» значенням 1.0.

Маємо наступні результати:



Таблиця, яка наведена вище, демонструє нам область застосування розроблених алгоритмів для гіперболічних функцій $\sinh(x)$ і $\cosh(x)$, коли вони обчислюються в вигляді степеневих рядів.

Ситуація тут значно гірша, ніж для тригонометричних функцій.

Похибка обчислень робить дані алгоритми марними вже при $x > 10$.

Причина такого падіння точності обчислювального алгоритму криється в сталих знаках членів степеневих рядів для функцій $\sinh(x)$ і $\cosh(x)$.

Нічим іншим (в даному контексті) тригонометричні функції $\sin(x)$ і $\cos(x)$ не відрізняються від гіперболічних функцій $\sinh(x)$ і $\cosh(x)$.

Зазначимо дві основні причини даної обчислювальної проблеми.

По-перше, це обмеженість представлення дійсних чисел типом **`double`**, яке забезпечує 16 цифр в мантисі числа із плаваючою точкою.

По-друге, це «катастрофічна» втрата значущих цифр результату, коли з одного великого дійсного числа віднімається інше велике дійсне число, яке є приблизно рівним першому за абсолютною величиною.

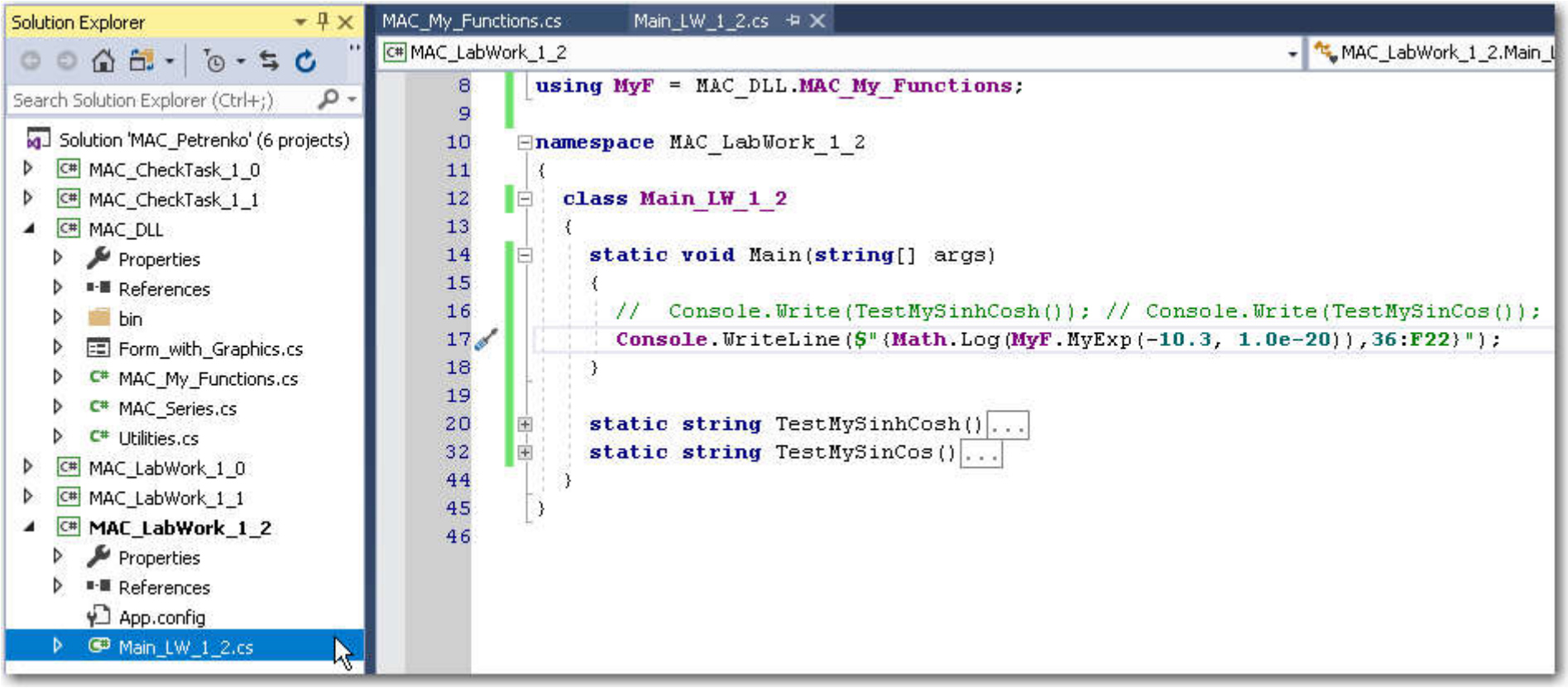
Сказане вище можна прослідкувати на наступному наочному прикладі.

«Перевіримо» очевидну рівність: $\ln(\exp(-10.3)) = -10.3$.

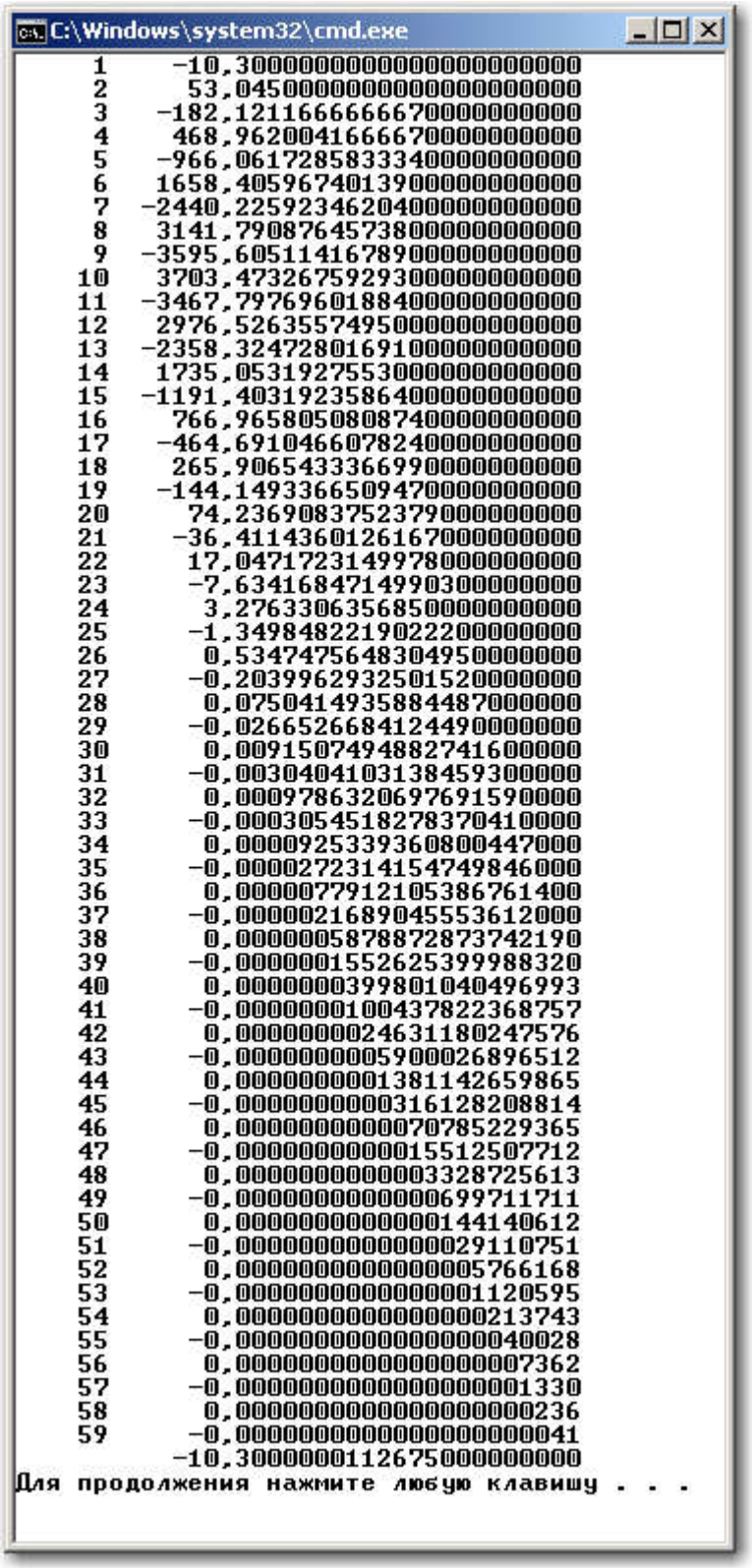
При цьому для обчислення експоненти будемо використовувати власну функцію **`MyExp()`**, в яку додатково внесемо оператор виводу значення члену степеневого ряду:

```
 9   public class MAC_My_Functions
10   {
11     public static double MySin(double x, double eps)...
21     public static double MyCos(double x, double eps)...
31
32     public static double MyExp(double x, double eps)
33     {
34       if (x == 0) return 1.0;
35       double exp = 1.0, pk = 1.0;
36       for (int k = 1; Math.Abs(pk) > eps; k++)
37       {
38         pk = pk * (x / k); exp += pk;  Console.WriteLine($"{k,6}{pk,30:F22}");
39       }
40       return exp;
41     }
42
43     public static double MySinh(double x, double eps)...
53     public static double MyCosh(double x, double eps)...
63   }
```

В основну програму додаємо відповідний рядок коду:



В якості результату будемо мати наступну таблицю числових даних:



Аналіз таблиці свідчить про те, що для аргументу $x = -10.3$ абсолютна похибка обчислення експоненціальної функції за алгоритмом (л.1.2.15) (із використанням $\varepsilon \sim 10^{-20}$) є числом `0,0000000112675`, тобто величиною $\sim 10^{-8}$.

Тепер нам треба перевірити якість розроблених алгоритмів для тригонометричних функцій за допомогою відомих аналітичних формул та бібліотечних математичних функцій мови програмування **C#**.

Наприклад, з математичного довідника беремо трипараметричну формулу для функції синус:

$$\sin(A + B + C) = \sin A \cdot \cos B \cdot \cos C + \cos A \cdot \sin B \cdot \cos C + \cos A \cdot \cos B \cdot \sin C - \sin A \cdot \sin B \cdot \sin C, \quad \text{(л.1.2.17)}$$

або в іншій формі $\Phi_1(A, B, C) = \Phi_2(A, B, C)$,

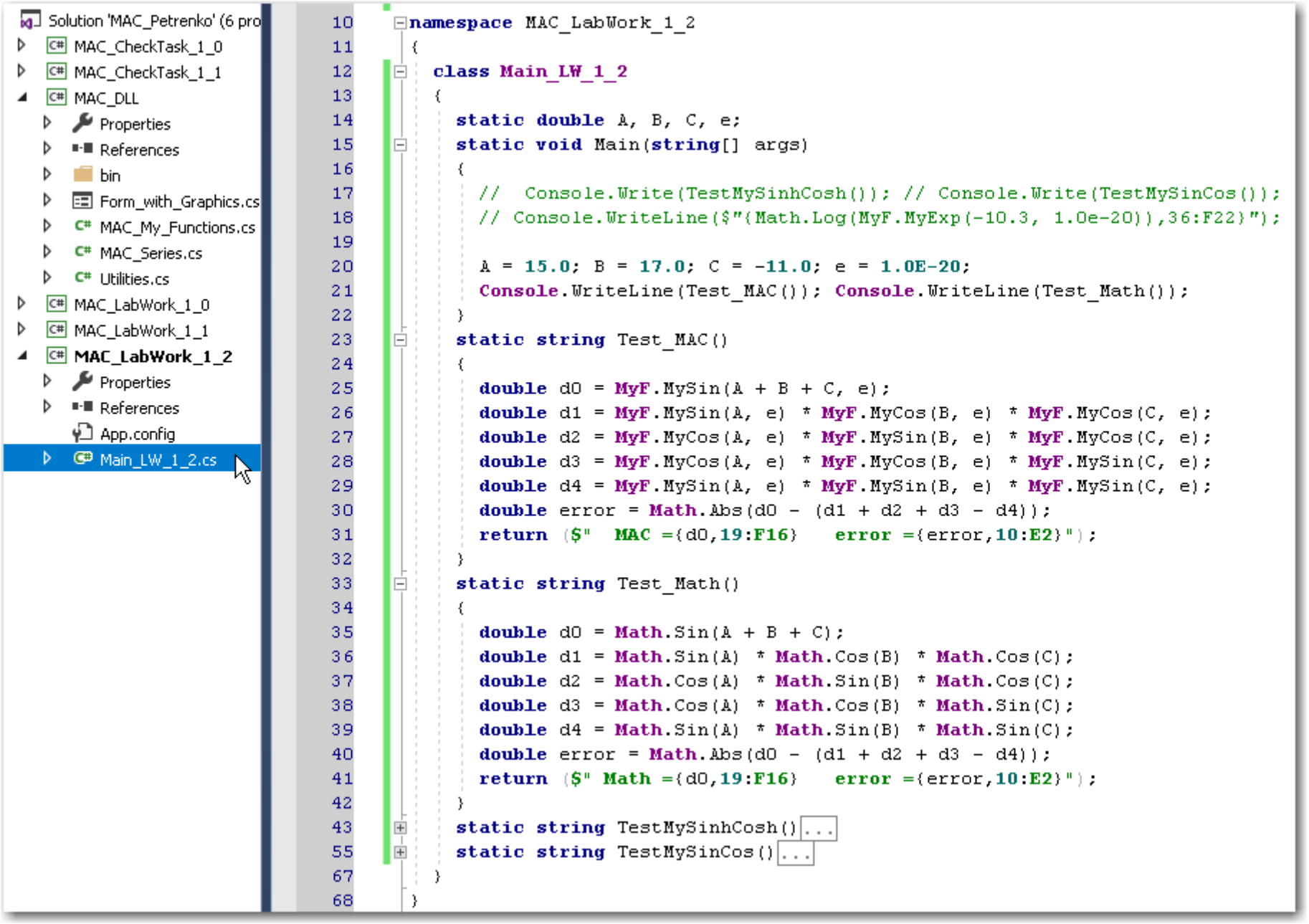
де
$$\Phi_1(A, B, C) = \sin(A + B + C),$$
$$\Phi_2(A, B, C) = \sin A \cdot \cos B \cdot \cos C + \cos A \cdot \sin B \cdot \cos C + \cos A \cdot \cos B \cdot \sin C - \sin A \cdot \sin B \cdot \sin C.$$

Визначимо абсолютну похибку $error = \left| \Phi_1(A, B, C) - \Phi_2(A, B, C) \right|$ обчислення даних виразів для деяких заданих значень параметрів A, B і C, коли в обчисленнях беруть участь:

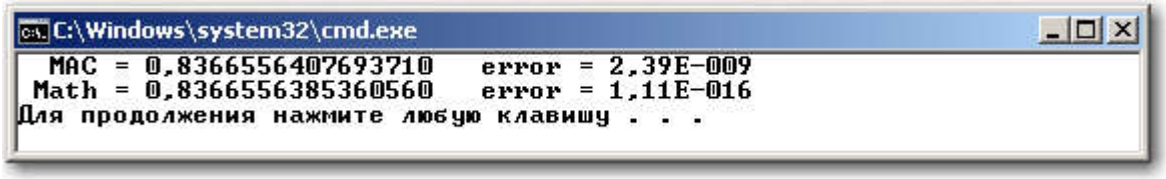
або математичні функції з класу **Math** мови програмування **C#**,

або методи, які були розроблені нами під час виконання даної лабораторної роботи, і які входять до складу класу **MAC_My_Functions** бібліотеки **MAC_DLL**.

Роботу виконуємо в рамках наявного консольного додатку **MAC_LabWork_1_2**.




```
10  namespace MAC_LabWork_1_2
11  {
12      class Main_LW_1_2
13      {
14          static double A, B, C, e;
15          static void Main(string[] args)
16          {
17              //  Console.Write(TestMySinhCosh()); // Console.Write(TestMySinCos());
18              // Console.WriteLine($"{Math.Log(MyF.MyExp(-10.3, 1.0e-20)),36:F22}");
19
20              A = 15.0; B = 17.0; C = -11.0; e = 1.0E-20;
21              Console.WriteLine(Test_MAC()); Console.WriteLine(Test_Math());
22          }
23          static string Test_MAC()
24          {
25              double d0 = MyF.MySin(A + B + C, e);
26              double d1 = MyF.MySin(A, e) * MyF.MyCos(B, e) * MyF.MyCos(C, e);
27              double d2 = MyF.MyCos(A, e) * MyF.MySin(B, e) * MyF.MyCos(C, e);
28              double d3 = MyF.MyCos(A, e) * MyF.MyCos(B, e) * MyF.MySin(C, e);
29              double d4 = MyF.MySin(A, e) * MyF.MySin(B, e) * MyF.MySin(C, e);
30              double error = Math.Abs(d0 - (d1 + d2 + d3 - d4));
31              return ($"  MAC ={d0,19:F16}    error ={error,10:E2}");
32          }
33          static string Test_Math()
34          {
35              double d0 = Math.Sin(A + B + C);
36              double d1 = Math.Sin(A) * Math.Cos(B) * Math.Cos(C);
37              double d2 = Math.Cos(A) * Math.Sin(B) * Math.Cos(C);
38              double d3 = Math.Cos(A) * Math.Cos(B) * Math.Sin(C);
39              double d4 = Math.Sin(A) * Math.Sin(B) * Math.Sin(C);
40              double error = Math.Abs(d0 - (d1 + d2 + d3 - d4));
41              return ($" Math ={d0,19:F16}    error ={error,10:E2}");
42          }
43          static string TestMySinhCosh()...
55          static string TestMySinCos()...
67      }
68  }
```

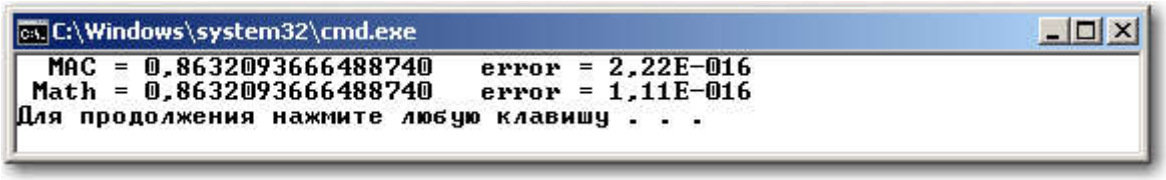


Отримуємо результат:

Вочевидь, що для великих значень аргументів тригонометричних функцій «якість» наших алгоритмів поступається якості алгоритмів фірми **`Microsoft`**.

Однак, якщо зменшити значення вхідних параметрів, наприклад, у 10 разів, то результат буде суттєво кращий:

```
12  class Main_LW_1_2
13  {
14      static double A, B, C, e;
15      static void Main(string[] args)
16      {
17          // Console.Write(TestMySinhCosh()); // Console.Write(TestMySinCos());
18          // Console.WriteLine($"{Math.Log(MyF.MyExp(-10.3, 1.0e-20)),36:F22}");
19
20          // A = 15.0; B = 17.0; C = -11.0; e = 1.0E-20;
21          A = 1.50; B = 1.70; C = -1.10; e = 1.0E-20;
22          Console.WriteLine(Test_MAC()); Console.WriteLine(Test_Math());
23      }
24      static string Test_MAC()...
34      static string Test_Math()...
44      static string TestMySinhCosh()...
56      static string TestMySinCos()...
68  }
```



На цьому загальну частину лабораторної роботи 1.2 завершено.

Далі Вам слід самостійно виконати наступний етап, який полягає в проведенні подібних тестових обчислень з іншою відомою тригонометричною формулою (при цьому можна використовувати проект **MAC_LabWork_1_2**).

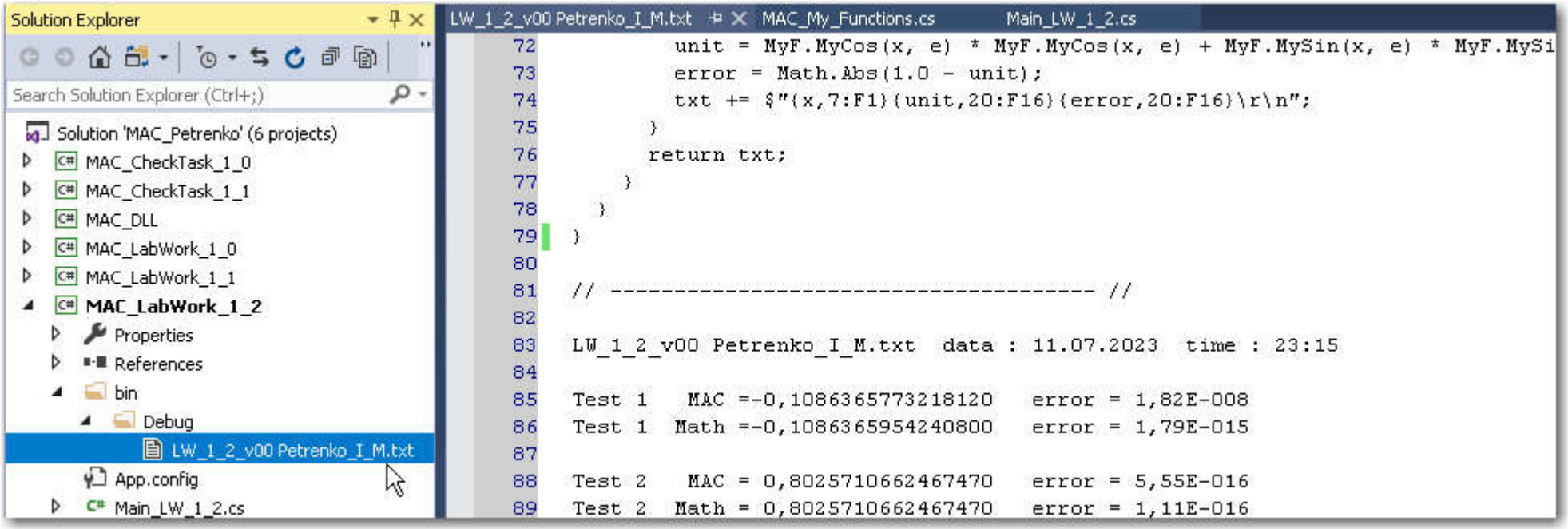
Результати цих обчислень записуються у відповідний текстовий файл та висилаються на перевірку.

Наведені вище обчислення можна вважати «зразком виконання» за індивідуальним варіантом лабораторної роботи 1.2 для «віртуального» студента.

Для «автоматичного» формування повного звіту (в формі файлу ***.txt**) за індивідуальним варіантом лабораторної роботи слід в існуючий проект консольного додатку додати відповідний метод, дивись наступну сторінку.

Не забудьте правильно сформувати м.'я файлу результатів, в якому необхідно обов'язково вказати номер вашого варіанту після літери **v**.

Також не забудьте включити цей файл до складу робочого проекту за даною лабораторною роботою:



```
1   using System;
2   using System.IO;
3   using UTL = MAC_DLL.Utilities;
4   using MyF = MAC_DLL.MAC_My_Functions;
5
6   namespace MAC_LabWork_1_2
7   {
8     class Main_LW_1_2
9     {
10      static double A, B, C, e;
11      static void Main(string[] args)
12      {
13        // Console.Write(TestMySinhCosh()); // Console.Write(TestMySinCos());
14        // Console.WriteLine($"{Math.Log(MyF.MyExp(-10.3, 1.0e-20)),36:F22}");
15        // A = 15.0; B = 17.0; C = -11.0; e = 1.0E-20;
16        // A = 1.50; B = 1.70; C = -1.10; e = 1.0E-20;
17        // Console.WriteLine(Test_MAC()); Console.WriteLine(Test_Math());
18
19        MyVariant("LW_1_2_v00");
20      }
21
22      static void MyVariant(string file)
23      {
24        using (StreamWriter SW = UTL.ResultWriter(file, "Main_LW_1_2.cs"))
25        {
26          A = 15.7; B = 18.3; C = -11.9; e = 1.0E-19;
27          SW.WriteLine("Test 1 " + Test_MAC()); SW.WriteLine("Test 1 " + Test_Math() + "\r\n");
28          A = 1.57; B = 1.83; C = -1.19;
29          SW.WriteLine("Test 2 " + Test_MAC()); SW.WriteLine("Test 2 " + Test_Math());
30        }
31      }
32      static string Test_MAC()...
42      static string Test_Math()...
52      static string TestMySinhCosh()...
64      static string TestMySinCos()...
76    }
77  }
```

Таблиця даних за варіантами

V	$\Phi_1(A,B)$	$\Phi_2(A,B)$	V.n	Тест 1		Тест 2	
				A	B	A	B
01	$2\sin(A)\cdot\cos(B)$	$\sin(A+B)+\sin(A-B)$	01.1	-14.50	-11.70	-6.58	+4.77
			01.2	-13.70	-10.90	-7.84	+5.16
			01.3	-15.60	-11.20	-6.91	+4.82
02	$\cos(A)+\cos(B)$	$2\cos\left(\frac{A+B}{2}\right)\cdot\cos\left(\frac{A-B}{2}\right)$	02.1	+15.20	+22.70	+2.75	-5.47
			02.2	+12.30	+22.40	+3.57	+1.98
			02.3	+14.90	+22.50	+3.23	-5.06
03	$2\cos(A)\cdot\cos(B)$	$\cos(A+B)+\cos(A-B)$	03.1	+15.70	+9.90	+4.25	-7.36
			03.2	+14.60	+10.80	-3.45	+6.29
			03.3	+15.30	+10.10	+4.06	-6.98
04	$\cos(A)-\cos(B)$	$2\sin\left(\frac{A+B}{2}\right)\cdot\sin\left(\frac{B-A}{2}\right)$	04.1	-13.60	+24.70	+3.35	-7.91
			04.2	+11.80	-23.10	-2.75	+3.41
			04.3	-13.70	+23.50	-2.39	+4.68

V	$\Phi_1(A,B)$	$\Phi_2(A,B)$	V.n	Тест 1		Тест 2	
				A	B	A	B
05	$2\sin(A)\cdot\sin(B)$	$\cos(A-B)-\cos(A+B)$	05.1	+17.20	+12.70	-3.42	-5.27
			05.2	+15.90	+13.20	-1.86	-2.44
			05.3	+16.40	+12.90	-2.57	-3.89
06	$\sin^2(A)-\sin^2(B)$	$\sin(A+B)\cdot\sin(A-B)$	06.1	-14.30	-15.20	+5.19	+2.34
			06.2	-13.40	-12.50	+4.92	+1.81
			06.3	-15.60	-14.50	+5.38	+2.29
07	$\sin(A)+\sin(B)$	$2\sin\left(\frac{A+B}{2}\right)\cdot\cos\left(\frac{A-B}{2}\right)$	07.1	+28.10	+15.30	-4.21	+5.63
			07.2	+24.80	+14.70	-2.31	+4.57
			07.3	+25.90	+15.10	-3.64	+5.05
08	$\cos^2(A)-\cos^2(B)$	$\sin(A+B)\cdot\sin(B-A)$	08.1	-14.60	-11.20	+6.15	+2.87
			08.2	-11.60	-10.50	+3.84	+2.90
			08.3	-13.90	-12.60	+4.28	+2.64

V	$\Phi_1(A,B)$	$\Phi_2(A,B)$	V.n	Тест 1		Тест 2	
				A	B	A	B
09	$\sin(A)-\sin(B)$	$2\sin\left(\frac{A-B}{2}\right)\cdot\cos\left(\frac{A+B}{2}\right)$	09.1	-25.90	+17.20	-3.38	-5.79
			09.2	-16.20	+27.30	-1.58	-1.94
			09.3	-23.70	+11.60	-2.27	-4.03
10	$\cos^2(A)-\sin^2(B)$	$\cos(A+B)\cdot\cos(A-B)$	10.1	+17.70	-15.40	-4.41	+3.28
			10.2	+15.40	-16.90	+1.34	+2.86
			10.3	+13.60	-18.10	-2.55	+3.71

Методика виконання завдання на самостійну роботу 1.2

«Обчислення значень степеневих рядів (спецфункцій)»

Розглянемо «технологію» складання алгоритму, призначеного для обчислення значень «модельної» спецфункції, яка має наступне функціональне зображення в вигляді степеневого ряду:

$$f_0(x) = \sum_{k=0}^{\infty} \frac{(-1)^k \cdot \left[\cos\left(\frac{b}{k+1}x\right) + \sin\left(\frac{b}{k+a}x\right)\right]}{4 \cdot (2k+a) \cdot (k+1)!} \left(\frac{11x}{7}\right)^{k+1}$$, де a і b – деякі додаткові числові параметри. (с.1.2.1)

Функція $f_0(x)$ має більш складне математичне зображення, ніж у елементарних трансцендентних функцій, які були розглянуті під час виконання лабораторної роботи 1.2.

Однак це не завадить нам скористатися для функції $f_0(x)$ вже відомим для нас підходом при складанні обчислювального алгоритму і відповідного до нього коду. Виконаємо попередні перетворення.

По-перше, винесемо за знак суми множники, які є спільними для всіх доданків:

$$f_0(x) = \left(\frac{1}{4} \cdot \frac{11x}{7}\right) \times \sum_{k=0}^{\infty} \frac{(-1)^k \cdot \left[\cos\left(\frac{b}{k+1}x\right) + \sin\left(\frac{b}{k+a}x\right)\right]}{(2k+a) \cdot (k+1)!} \left(\frac{11x}{7}\right)^{k}.$$

По-друге, виділимо перший доданок (при $k = 0$) із загальної суми:

$$f_0(x) = \frac{11x}{28} \times \left\{ \frac{\cos(bx) + \sin\left(\frac{b}{a}x\right)}{a} + \sum_{k=1}^{\infty} \frac{(-1)^k \cdot \left[\cos\left(\frac{b}{k+1}x\right) + \sin\left(\frac{b}{k+a}x\right)\right]}{(2k+a) \cdot (k+1)!} \left(\frac{11x}{7}\right)^{k} \right\}.$$

Звернемо увагу на той факт, що ***частина*** виразу, яка стоїть під знаком суми, може бути обчислена рекурентним шляхом, тобто через відповідне йому «попереднє» значення. Тому вводимо вже «традиційне» позначення:

$$p_k = \frac{(-1)^k}{(k+1)!} \cdot \left(\frac{11x}{7}\right)^k,$$

тоді $p_0 = 1$ при $k = 0$, і $p_k = -p_{k-1} \cdot \frac{\tilde{x}}{k+1}$, при $k > 0$, де $\tilde{x} = \frac{11x}{7}$. (с.1.2.2)

Таким чином, k – й доданок, який стоїть під знаком суми, ми можемо представити в виді

$$\mathrm{A}_k = \frac{p_k}{2k+a} \cdot \left[\cos\left(\frac{bx}{k+1}\right) + \sin\left(\frac{bx}{k+a}\right)\right] \quad \text{та} \quad \mathrm{A}_0 = \frac{1}{a} \cdot \left[\cos(bx) + \sin\left(\frac{bx}{a}\right)\right]. \qquad \text{(с.1.2.3)}$$

Тоді, з урахуванням всіх позначень, задана функція (с.1.2.1) може бути обчислена за наступною компактною формулою:

$$f_0(x) = \frac{\tilde{x}}{4}\left\{\mathrm{A}_0 + \sum_{k=1}^{\infty} \mathrm{A}_k\right\}. \qquad \text{(с.1.2.4)}$$

Приступаємо до безпосереднього програмування формул (с.1.2.2)–(с.1.2.4), тобто складання коду методу, який обчислюватиме значення заданої функції $f_0(x)$.

Відразу відзначимо для себе, що функція (с.1.2.1) повинна приймати у якості вхідних параметрів чотири величини, які мають тип **`double`**.

Перша з них – безпосередньо змінна x, дві наступні – a і b – математичні параметри функції $f_0(x)$, остання – eps – абсолютна похибка обчислень.

Розміщувати код цієї функції будемо в класі **MAC_My_Functions** нашої бібліотеки **MAC_DLL**.

```
7   namespace MAC_DLL
8   {
9     public class MAC_My_Functions
10    {
11      public static double f0(double x, double a, double b, double eps)
12      {
13        if (x == 0) return 0.0;
14        double xz = 11.0 * x / 7.0, bx = b * x;
15        double pk = 1.0, Ak = 1.0, summa = 0.0, Ao = (Math.Cos(bx) + Math.Sin(bx / a)) / a;
16
17        for (int k = 1; Math.Abs(Ak) > eps; k++)
18        {
19          pk = -pk * xz / (k + 1.0);
20          Ak = pk * (Math.Cos(bx / (k + 1.0)) + Math.Sin(bx / (k + a))) / (2.0 * k + a);
21          summa += Ak;
22        }
23        return 0.25 * xz * (Ao + summa);
24      }
25
26      public static double MySin(double x, double eps)...
36      public static double MyCos(double x, double eps)...
46      public static double MyExp(double x, double eps)...
56      public static double MySinh(double x, double eps)...
66      public static double MyCosh(double x, double eps)...
76    }
77  }
```

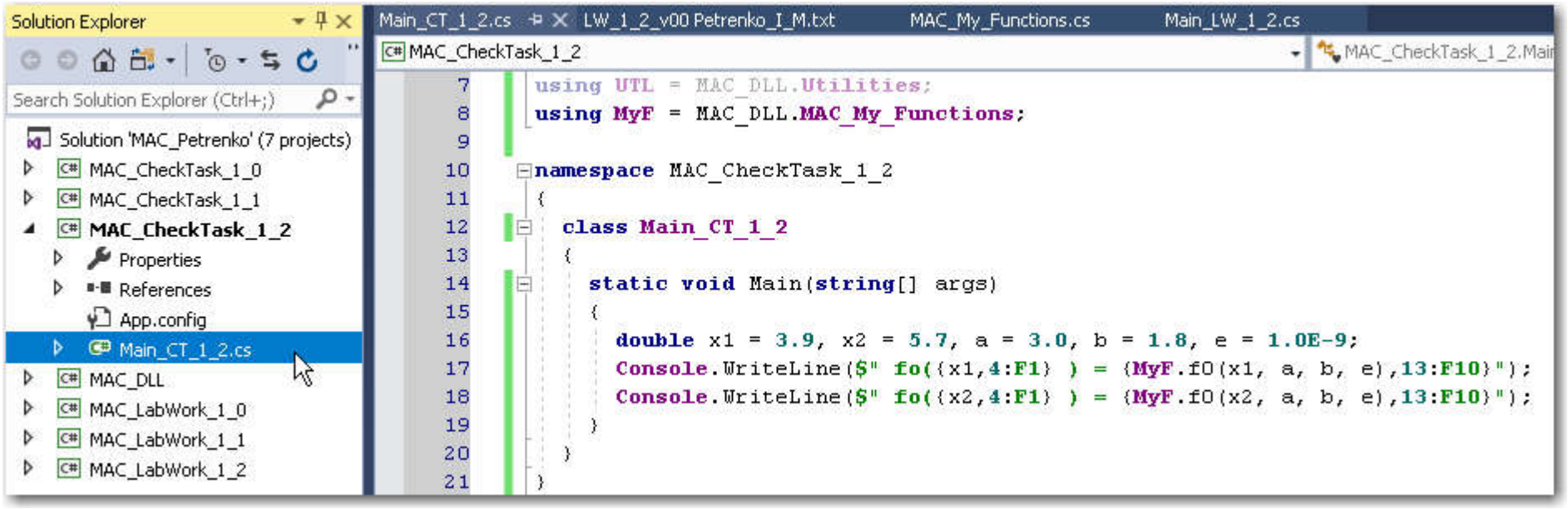
Обчислення значень заданої функції (с.1.2.1) будемо виконувати в рамках окремого проекту консольного додатку **MAC_CheckTask_1_2**, який (як ми і домовлялися раніше) додамо до існуючого робочого простору **MAC_Petrenko**.

Для зручності одразу перейменуємо клас **Program** в клас **Main_CT_1_2** і стандартним шляхом добавимо бібліотеку **MAC_DLL** до переліку **References** нового проекту.

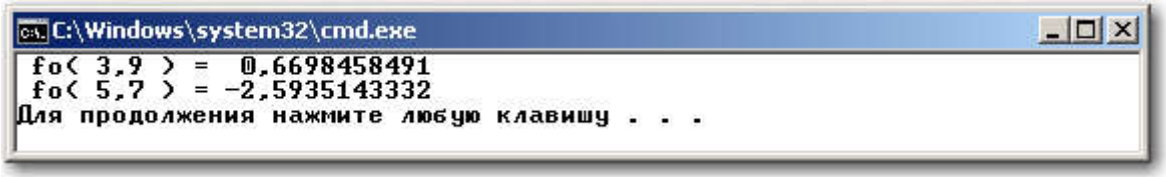
Нехай нам потрібно обчислити два значення функції $f_0(x)$, які відповідають значенням $x_1 = 3.9$ і $x_2 = 5.7$ її аргументу, при заданих значеннях $a = 3.0$ і $b = 1.8$ її параметрів.

При цьому абсолютна похибка обчислень значень заданої функції $f_0(x)$ повинна бути не гіршою, ніж 10^{-9}.

Остаточний вид додатку **`MAC_CheckTask_1_2`** може бути наступним:



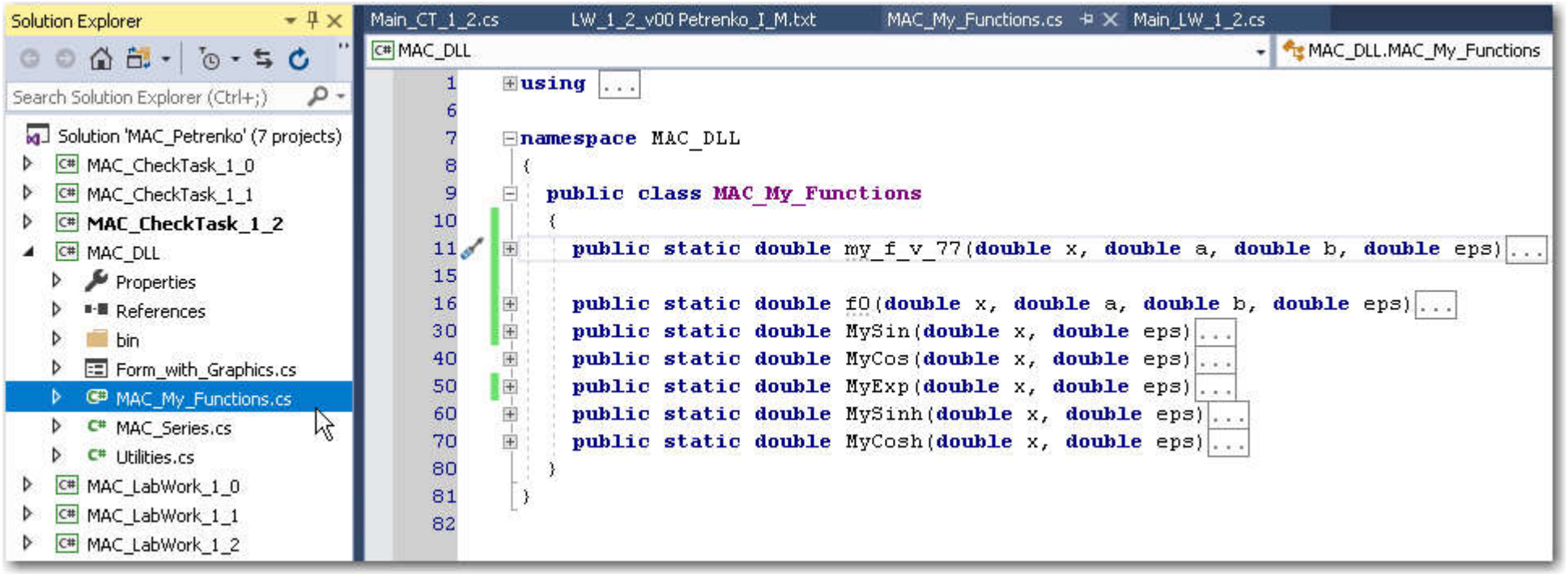
Маємо шуканий результат виконання основної програми:



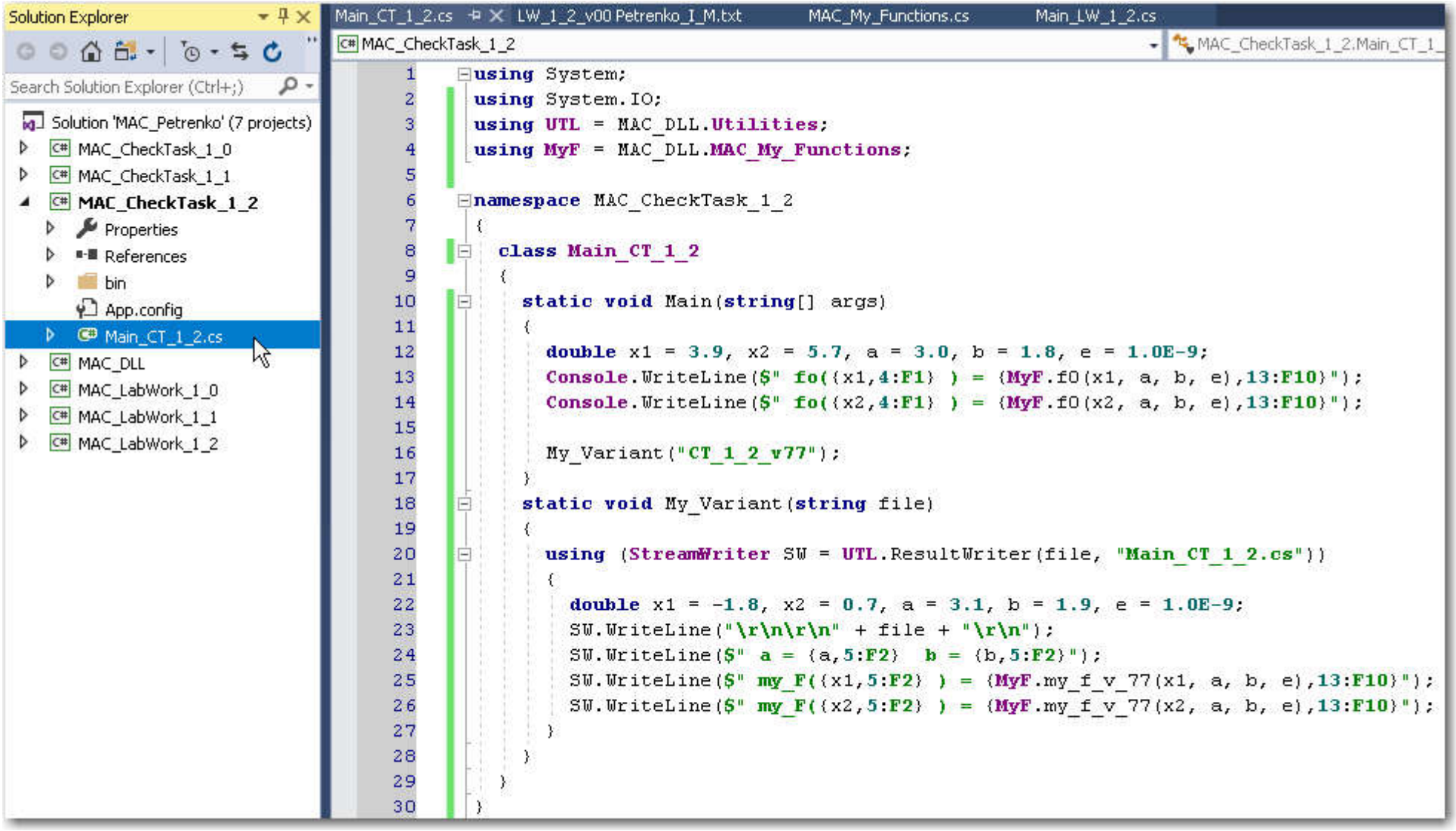
На цьому загальна методична частина вказівок закінчується.

Тепер Ви повинні виконати аналогічне проектування для індивідуального варіанта функції та значень її параметрів.

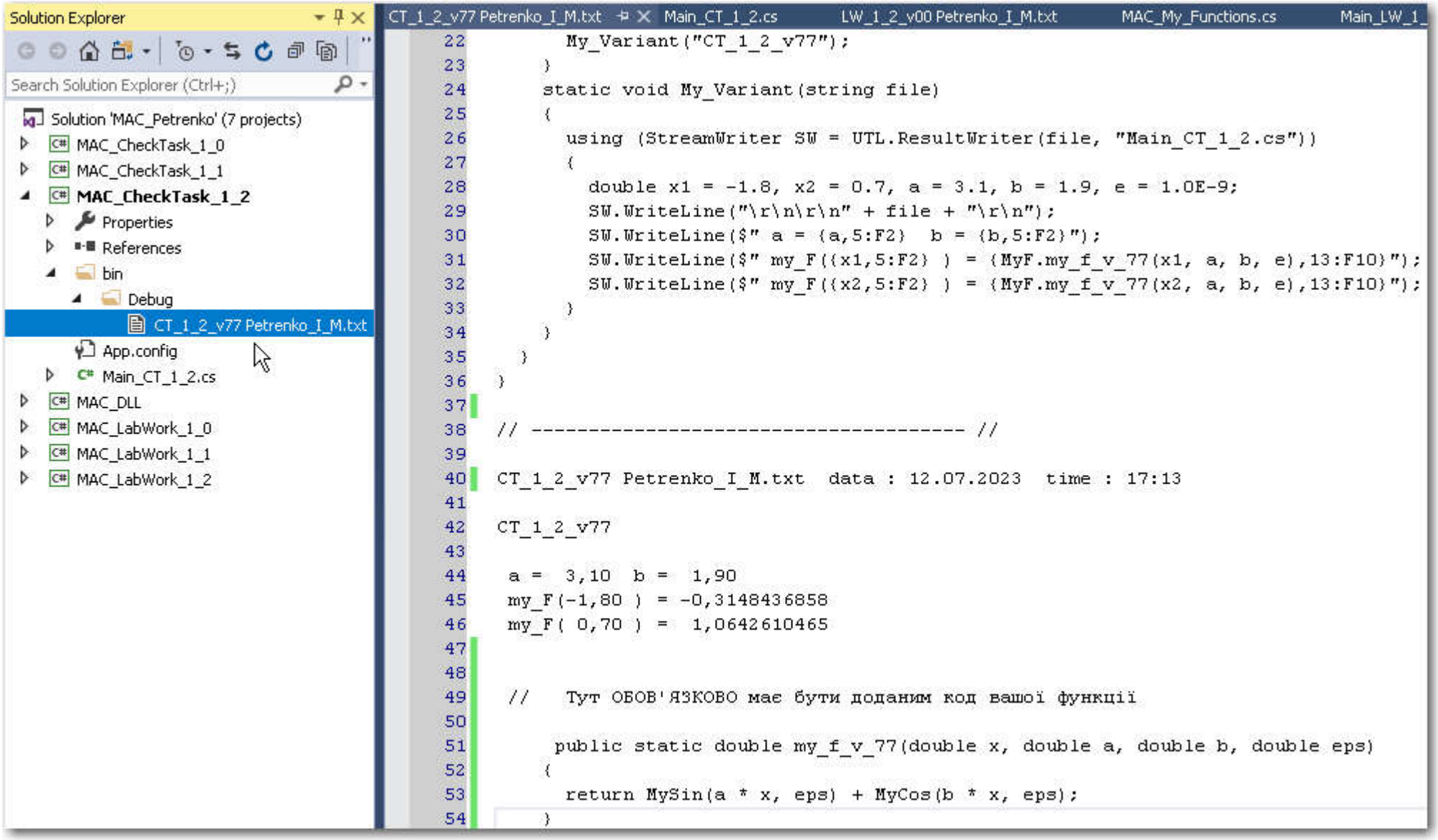
Функцію (метод), яку Вам треба розробити самостійно, слід розміщувати в класі **MAC_My_Functions** бібліотеки **MAC_DLL** та іменувати, наприклад, **my_f_v_77**, де **77** – номер вашого варіанта.



Для програмування обчислень слід використовувати існуючий проект **`MAC_CheckTask_1_2`**.



Результати обчислень – текстовий файл, який слід надсилати викладачу, обов'язково повинен містити код основної програми та код функції **`my_f_v_77`** (одразу після числових результатів).



Завдання на самостійну роботу 1.2 за варіантами

Тема: **«Обчислення значень степеневих рядів (спецфункцій)»**

Завдання:

Розробити алгоритм обчислення значень функції $f(x)$, яку задано в виді степеневого ряду (відповідну методику відпрацьовано під час виконання лабораторної роботи 1.2 і у вказівках до виконання самостійної роботи 1.2).

Реалізувати цей алгоритм в виді окремого статичного метода в складі класу **`MAC_My_Functions`** існуючої бібліотеки класів **`MAC_DLL`**.

Налагодити консольний **`Windows`**–додаток, призначений для обчислення значень заданої функції $f(x)$ з абсолютною похибкою $\varepsilon \sim 10^{-9}$ при заданих значеннях параметрів a і b.

Рекомендовано скористатися вже існуючим проектом **`MAC_CheckTask_1_2`**.

В рамках цього додатку здійснити розрахунок двох контрольних значень $f(x_1)$ і $f(x_2)$ заданої функції.

Таблиця даних за варіантами

$f_V(x)$	V.n	a	b	x_1	x_2
$f_1(x) = \sqrt{\frac{x}{2}} \cdot \sum_{k=0}^{\infty} \left[\frac{(-x^2/4)^k}{(k)! \cdot (k+1)!} \cdot \sin\left(\sqrt{\frac{a}{k+1}} \cdot x + b \right) \right]$	01.1	+1.97	-2.24	+2.35	+3.76
	01.2	+1.63	-3.27	+1.41	+4.53
	01.3	+1.72	-2.43	+1.89	+4.14
$f_2(x) = \sum_{k=0}^{\infty} \left[\frac{(-1)^k x^{2k+1}}{(2k+b) \cdot (2k+1)!} \cdot \cos\left(\frac{\pi}{2} \frac{ax^2}{kx+1} \right) \right]$	02.1	+0.49	+1.76	+2.51	+3.83
	02.2	+0.87	+1.33	+2.36	+3.71
	02.3	+0.75	+1.64	+2.43	+3.59
$f_3(x) = \sum_{k=1}^{\infty} \left[\frac{(-1)^k x^{2k}}{2k \cdot (2k)!} \cdot \left(\cos\left(\frac{a\sqrt{x+\pi}}{k} \right) + \sin\left(\frac{bx^{3/2}+\pi}{k} \right) \right) \right]$	03.1	+0.89	-1.73	+1.82	+4.45
	03.2	+0.53	-0.91	+2.94	+4.68
	03.3	+0.64	-1.35	+2.28	+4.89
$f_4(x) = \sum_{k=0}^{\infty} \left[\frac{(-1)^k x^{2k}}{(k)! \cdot (2k+1)} \cdot \left(\frac{\cos(ax)}{kx^2+\sqrt{\pi}} - \frac{1}{\sqrt{2}} \cdot \frac{\sin(bx)}{kx+b} \right) \right]$	04.1	+2.81	+4.03	+2.36	+4.97
	04.2	+2.34	+3.68	+2.11	+5.27
	04.3	+2.53	+3.81	+2.19	+5.36
$f_5(x) = \sin(ax-b) \cdot \sum_{k=0}^{\infty} \left[\frac{(-1)^k \cdot x^{4k+1}}{(2k)! \cdot (4k+3)} \cdot \left(1 + \mathrm{th}(kx^2+b) \right) \right]$	05.1	+5.81	-1.73	+2.38	+3.84
	05.2	+6.34	-2.51	+1.76	+4.28
	05.3	+6.26	-2.38	+1.47	+4.09

$f_V(x)$	V.n	a	b	x_1	x_2
$f_6(x) = \sum_{k=0}^{\infty}\left[\frac{(-1)^k \cdot (2x)^{k+1}}{(ak+b)\cdot(k!)^{1.5}} \cdot \cos\left(\pi x + \log_{10}(kx+b)\right)\right]$	06.1	+1.76	+2.39	+2.48	+4.16
	06.2	+2.13	+3.56	+2.64	+4.79
	06.3	+1.95	+2.76	+2.31	+4.58
$f_7(x) = \left(\cos(ax) + \sin(bx)\right) \cdot \sum_{k=0}^{\infty}\left[\frac{(-1)^k \cdot (x/2)^{2k}}{(bk+a)\cdot(2k+1)!}\right]$	07.1	+2.47	+2.93	+3.42	+5.21
	07.2	+3.05	+4.68	+2.63	+6.09
	07.3	+2.71	+3.64	+2.25	+5.78
$f_8(x) = \left(\sin\left(ax^2\right) + \sin(bx)\right) \cdot \sum_{k=1}^{\infty}\left[\frac{(-1)^k \cdot (x/3)^{2k}}{\sqrt{ak+b}\cdot(2k-1)!}\right]$	08.1	+0.62	+1.19	+2.65	+4.83
	08.2	+0.72	+2.36	+1.93	+5.24
	08.3	+0.58	+1.77	+2.34	+4.96
$f_9(x) = \ln(x+a) \cdot \cos(bx) \cdot \sum_{k=1}^{\infty}\left[\frac{(-1)^{k+1} \cdot (x/b)^k}{(a+b\sqrt{k})\cdot(2k)!}\right]$	09.1	+1.35	+2.47	+3.52	+5.86
	09.2	+0.84	+1.76	+2.15	+6.18
	09.3	+1.06	+1.93	+2.48	+5.72
$f_{10}(x) = \sum_{k=1}^{\infty}\left[\frac{(-1)^{k-1} x^{k-1}}{(ak-b)\cdot(k+1)!} \cdot \sin\left(ax + \log_{10}(k+x)\right)\right]$	10.1	+1.38	-2.42	+3.27	+5.35
	10.2	+1.23	-3.41	+2.38	+6.26
	10.3	+1.28	-2.73	+2.91	+5.84

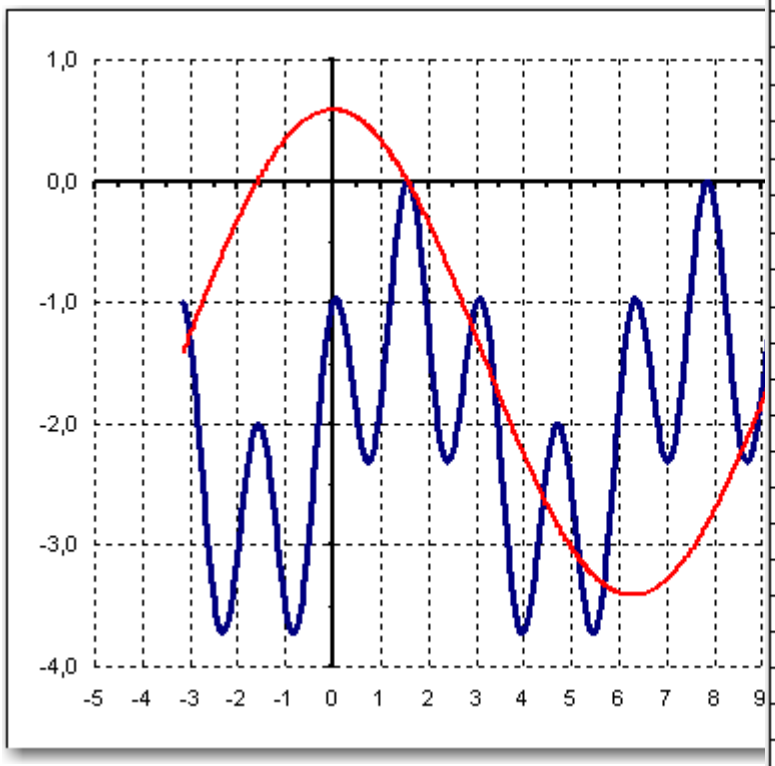
Таблиця ***тестових*** результатів за варіантами самостійної роботи 1.2

V.n	a	b	x_1	$f_V(x_1)$	x_2	$f_V(x_2)$
01.0	+1.50	-2.00	+2.50	+0.7925680161	+3.50	-0.3069116197
02.0	+0.50	+1.50	+2.50	+0.2890879467	+3.20	+0.3050709488
03.0	+1.00	-0.50	+2.50	+0.0098364490	+5.00	+5.4693535330
04.0	+2.00	+4.00	+2.00	-0.3767224509	+5.00	-0.4664821840
05.0	+5.50	-1.50	+2.00	+0.1206466332	+3.50	-2.2252029495
06.0	+1.50	+2.50	+3.00	+0.0580223995	+5.00	+0.1243954257
07.0	+2.30	+3.10	+3.00	+0.3479710586	+4.50	+0.1148569555
08.0	+0.50	+1.50	+2.00	-0.3086823606	+4.00	-0.6743965722
09.0	+1.00	+3.00	+3.00	-0.1481269063	+5.50	-0.2679400356
10.0	+1.00	-2.00	+3.00	-0.0361313697	+4.50	-0.0644695471

Лабораторна робота 1.3

«Формування наборів числових даних в виді таблиці функції»

Аналіз фізичних процесів та їх математичних моделей нерозривно пов’язаний з такими відомими поняттями як функція, графік функції, таблиця функції.



x	y = f1(x)	y = f2(x)
-3,1416	-1,0000	-1,4142
-3,1227	-1,0217	-1,3953
-3,1039	-1,0491	-1,3765
-3,0850	-1,0821	-1,3576
-3,0661	-1,1206	-1,3388
-3,0473	-1,1646	-1,3199
-3,0284	-1,2138	-1,3011
-3,0095	-1,2680	-1,2822
-2,9906	-1,3272	-1,2634
-2,9718	-1,3910	-1,2446
-2,9529	-1,4591	-1,2258
-2,9340	-1,5313	-1,2070
-2,9152	-1,6073	-1,1883
-2,8963	-1,6868	-1,1695
-2,8774	-1,7693	-1,1508
-2,8586	-1,8545	-1,1321
-2,8397	-1,9420	-1,1135
-2,8208	-2,0315	-1,0948
-2,8020	-2,1225	-1,0762
-2,7831	-2,2145	-1,0576
-2,7642	-2,3072	-1,0391

Теоретичні дослідження, як правило, оперують з функціями, для яких відоме (або задається) аналітичне зображення. Наприклад, закон гармонічного коливання матеріальної точки зображує наступна функція часу: $x(t) = a \cdot \sin(\omega t + \alpha)$.

Інший приклад – теоретична залежність в'язкості μ моторного мастила від його температури θ зображується теоретичною формулою (кінетикою) $\mu(\theta) = \mu_o \cdot \exp\left(-\delta_o \cdot \frac{\theta}{1+\theta}\right)$.

На основі цих аналітичних зображень виконується послідовний математичний аналіз фізичних моделей, та робляться ті або інші теоретичні висновки і рекомендації.

Інша ситуація виникає, коли фізичне явище (процес) або його математична модель досліджуються за допомогою натурного (або комп'ютерного) експерименту (досліду).

В цьому випадку в розпорядженні науковця маються лише тільки дискретні числові дані (результати експерименту), які описують явище, та саме ці дані йому слід систематизувати та обробити задля досягнення кінцевої мети дослідження.

Як правило, такі числові дані зберігаються в виді таблиць, які в дискретній формі визначають ті або інші фізичні залежності.

Методи наближених обчислень повинні забезпечити нас можливістю «обробки» функцій в обох зазначених формах: в виді аналітичної залежності $f = f(x)$, і в формі дискретно заданої функції, записаної в виді впорядкованої таблиці даних.

Ми постараємося для цього використовувати всі переваги **ООП**, які надає нам сучасна система розробки застосувань (додатків) **MS Visual Studio** та мова програмування **C#**.

Вочевидь, що нам треба буде виконати певну підготовчу роботу, яка безпосередньо не пов'язана з чисельними методами, але її результати забезпечать нам подальше комфортне програмування та прозорість в зрозумілості основ та самої суті обчислювальної математики.

Нові програмні компоненти (інтерфейси, делегати, класи, тощо) будемо розміщувати в робочому просторі **MAC_Petrenko** в проекті **MAC_DLL** бібліотеки класів **(.dll)** у відповідних файлах.

Клас `MAC_Table_Node`

Приступимо до формування першого власного типу даних (класу).

Будемо спиратися на те, що основним об'єктом чисельного аналізу, який досліджується, є задана таблиця функції $f = f(x)$.

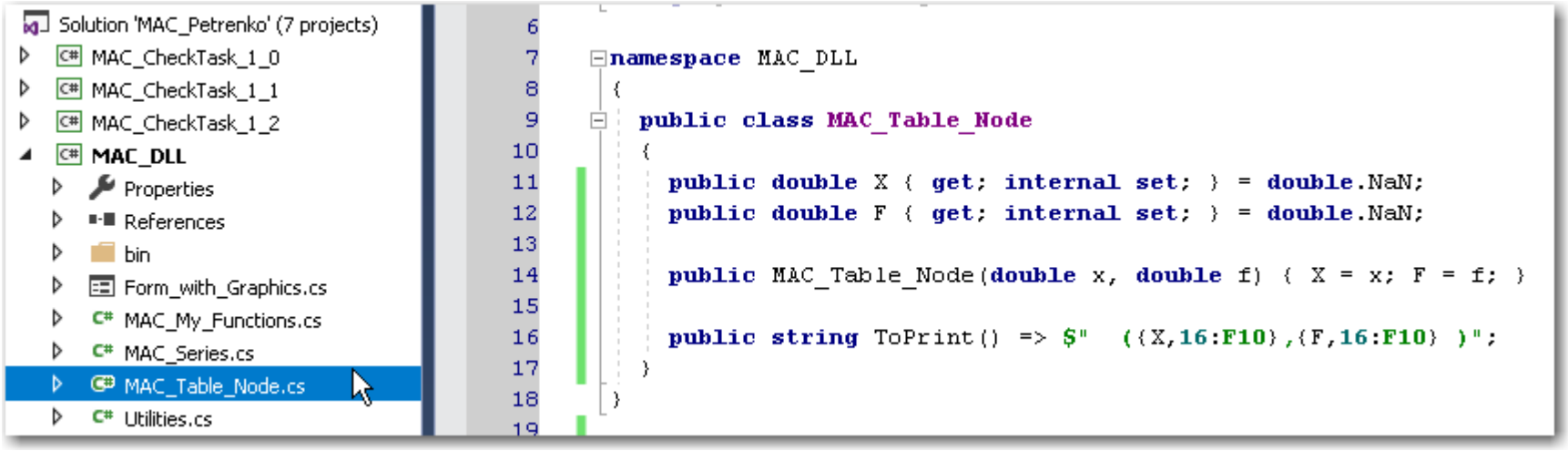
Під таблицею функції ми будемо розуміти впорядковану кінцеву множину пар (x, f) дійсних чисел, де x – аргумент, а f – відповідне до нього значення функції.

Кожну таку пару чисел, яка «займає» один рядок в таблиці функції, будемо називати вузлом таблиці.

Рядки таблиці нумеруються від нуля і слідують один за одним у відповідності до правила зростання значень аргументу x.

Таким чином, нульовий рядок таблиці містить найменше значення аргументу x, а останній рядок таблиці – найбільше значення аргументу x (та відповідні їм значення функції).

Пару дійсних чисел (x, f), яка є вузлом таблиці, ми «поєднаємо» в рамках нового класу `MAC_Table_Node`, який розмістимо в щойно створеному файлі:

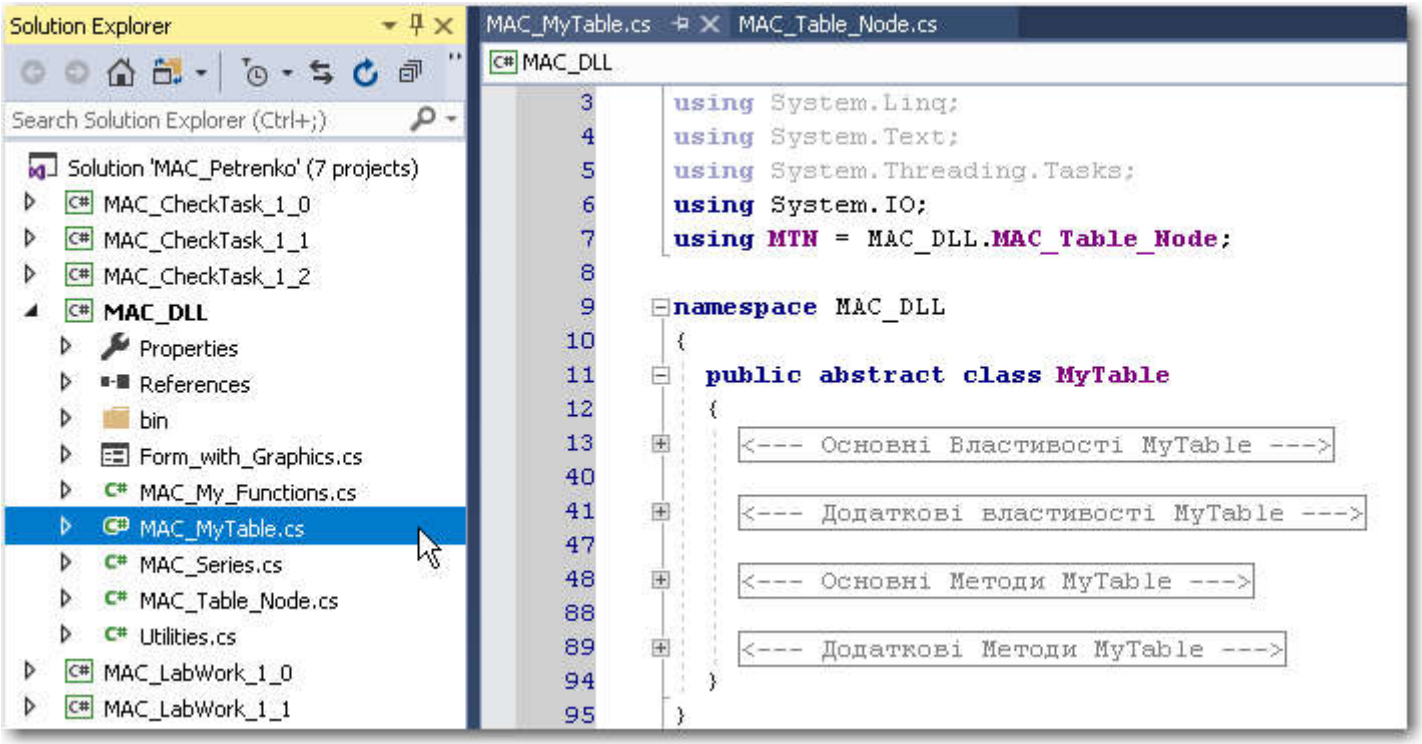


Клас **MAC_Table_Node** має конструктор і метод виводу даних **ToPrint()**.

Метод **ToPrint()** призначений для візуального зображення числових даних, які в інтуїтивно зрозумілій текстовій формі виражають зміст екземпляра даного класу.

Для самої таблиці функції ми підготуємо новий абстрактний клас **MyTable**, який потім будуть успадковувати інші класи, які описуватимуть конкретні таблиці, в основі генерації яких будуть або аналітичні вирази $f = f(x)$, або певні набори числових даних, отриманих з експерименту.

Цей клас розташуємо в окремому файлі **MAC_MyTable.cs** в складі бібліотеки класів **MAC_DLL**.



Абстрактний клас `MyTable`.

Розглянемо розділ основних властивостей абстрактного класу **`MyTable`**:

```
9   namespace MAC_DLL
10  {
11    public abstract class MyTable
12    {
13      #region    <--- Основні Властивості MyTable --->
14
15      // Масив вузлів (x,f) даної таблиці
16      public MTN[] Nodes { get; protected set; } = null;
17
18      // Повертає кількість вузлів в даній таблиці
19      public int Length { get { return Nodes == null ? 0 : Nodes.Length; } }
20
21      // Повертає посилання на вузол з найбільшим значенням функції
22      public MTN Maximum { get; protected set; } = null;
23
24      // Повертає посилання на вузол з найменшим значенням функції
25      public MTN Minimum { get; protected set; } = null;
26
27      // Повертає довжину області визначенняя для даної функції
28      public double Region_X { get { return Nodes == null ? double.NaN : this[Length - 1].X - this[0].X; } }
29
30      // Повертає довжину області значень для даної функції
31      public double Region_F { get { return Nodes == null ? double.NaN : Maximum.F - Minimum.F; } }
32
33      // Властивість, яка визначає похибку математичних операцій
34      public double Epsilon { get; set; }
35
36      // Властивість, яка визначає заголовок таблиці
37      public string Title { get; protected set; } = null;
38
39      #endregion <--- Основні Властивості MyTable --->
40
41      <--- Додаткові властивості MyTable --->
```

Властивість **Nodes[]** в даному класі власне і буде носієм основної числової інформації – а саме таблиці деякої функціональної залежності $f = f(x)$.

Вочевидь, що всі числові дані, які однозначно визначають математичну суть таблиці, після її безпосереднього формування (або генерації) не повинні змінюватися. Тому до відповідних властивостей ми додали такий атрибут, як **protected**.

Ми можемо змінювати ці дані тільки в методах класів-нащадків. Сенс властивостей, які були представлені вище, можна зрозуміти з коментарів в коді.

Далі – розділ основних методів абстрактного класу **MyTable**:

```
48      #region    <--- Основні Методи MyTable --->
49
50      // Індексатор MyTable
51      public MTN this[int index] { get { return (0 <= index) && (index < Length) ? Nodes[index] : null; } }
52
53      // Повертає значення x для рядка таблиці з номером index
54      public double X(int index) => (0 <= index) && (index < Length) ? this[index].X : double.NaN;
55
56      // Повертає значення f для рядка таблиці з номером index
57      public double F(int index) => (0 <= index) && (index < Length) ? this[index].F : double.NaN;
58
59      // Повертає два масиви із значеннями аргументу X і функції F
60      public void ToArrays(out double[] x, out double[] f)
61      {
62          x = new double[Length]; f = new double[Length];
63          for (int i = 0; i < Length; i++) { x[i] = this[i].X; f[i] = this[i].F; }
64      }
65
```

```
66      // Створює таблицю функції в текстовому форматі
67      public virtual string Table_of_Function()
68      {
69        string txt = "\r\n  Таблиця функції " + Title + " : \r\n";
70        for (int i = 0; i < Length; i++) { txt += $"{i,4}" + Nodes[i].ToPrint() + "\r\n"; }
71        txt += $"\r\n  x = [{this[0].X,17:F12} :{this[Length - 1].X,17:F12} ]";
72        txt += $"    x_Reg = {Region_X,16:F12}\r\n";
73        txt += $"\r\n Min -> {Minimum.ToPrint()} \r\n Max -> {Maximum.ToPrint()}";
74        txt += $"    f_Reg = {Region_F,16:F12}\r\n"; return txt;
75      }
76
77      // Віртуальний метод, призначений для збереження таблиці у текстовому файлі path із коментарем comment
78      public virtual void To_txt_File(string path, string comment)
79      {
80        if (path == "") path = "My_Table.txt";
81        FileInfo file = new FileInfo(path); if (file.Exists) file.Delete();
82        StreamWriter SW = new StreamWriter(file.OpenWrite());
83        SW.Write(comment + "\r\n" + Table_of_Function()); SW.Close();
84      }
85      #endregion <--- Основні Методи MyTable --->
86
87      #region    <--- Додаткові Методи MyTable --->
88      // Додатковий віртуальний метод, презначений для операцій виводу
89      //    в текстовій формі різноманітної інформації по даній таблиці
90      public virtual string ToPrint(string comment) { return comment; }
91      #endregion <--- Додаткові Методи MyTable --->
92    }
```

Сенс вище перелічених методів Вам потрібно розібрати самостійно, користуючись кодом та коментарями до нього.

Тепер ми маємо основу абстрактного класу **MyTable**, ґрунтуючись на якій ми зможемо надалі проектувати нові класи-нащадки, які вже будуть враховувати особливості формування конкретних видів таблиць функцій.

По мірі розширення кола задач, які розв'язуватимуться методами обчислювальної математики, ми будемо доповнювати новими властивостями і методами як сам абстрактний клас **MyTable**, так і його класи-нащадки.

Клас `MyTableOfData`

Отже, нехай результати деякого чисельного експерименту, реалізованого в виді самостійного **`Windows`**–додатку, зберігаються в формі іменованого файлу із невпорядкованими числовими даними (наприклад, на жорсткому диску комп'ютера).

Для зберігання великих об'ємів числових даних, як правило, використовуються текстові форматні файли або файли бінарного типу.

Перші зручні тим, що дозволяють досліднику швидко переглядати результати і робити попередні висновки на основі їх безпосереднього візуального аналізу.

Однак такі файли не є зручними для подальшого використання в рамках інших додатків, оскільки потребують додаткових зусиль по організації зчитування текстової інформації з подальшою її конвертацією в набори даних числового типу, над якими слід виконувати певні математичні дії та перетворення.

Бінарні файли зберігають в собі інформацію у так званій «внутрішній» формі, яка визначається архітектурою даного конкретного процесора (та операційної системи).

Тому без будь-яких додаткових конвертацій дані з цих файлів можна «завантажувати» безпосередньо в оперативну пам'ять комп'ютера (для їх подальшої обробки іншими додатками).

Крім того, при одному і тому ж об'ємі корисної математичної інформації бінарні файли мають розміри в декілька разів менші, ніж файли текстові.

Оскільки, як вже зазначалося вище, файл результатів формується окремим самостійним **`Windows`**–додатком, в програмі, яку ми будемо далі розробляти, слід передбачити способи «відкривання» файлів обох указаних типів.

Введемо наступне правило ідентифікації файлів результатів чисельного експерименту, яке (якщо виникне така потреба) можна буде легко модифікувати або доповнити.

Текстовий файл даних повинен мати розширення «**.txt**».

При цьому пара числових даних (x, f) повинна розділятися між собою в рядку символами `'пробіл'` або `';'`.

Таке розширення дозволяє відкривати та переглядати зазначений файл за допомогою будь-якої стандартного **Windows**–додатку, наприклад, «**Блокнот**» або **MS Word**.

Файл бінарного типу повинен мати розширення «**.bin**».

На даному етапі розробки будемо вимагати, щоб один запис в файлі результатів містив в якості інформації два дійсні числа (типу **double**): значення незалежної змінної x та відповідне до нього значення функції f.

Наприклад, це може бути момент часу t і координата матеріальної точки x, або ж x – координата та y – координата точки, яка здійснює рух на площині Oxy.

Алгоритм, який ми розроблятимемо, повинен виконувати наступні послідовні дії:

- відкривати заданий файл (який попередньо був розміщений в директорії додатку);
- виділяти пам'ять під список значень «пара дійсних чисел» (x, f) – для змінної x та функції $f = f(x)$;
- читати всі дані з файлу в зазначений список;
- впорядковувати отриманий список за зростанням аргументу x та перетворювати його на відповідний масив вузлів таблиці;
- визначати додаткові характеристики впорядкованої таблиці;
- зберігати впорядковану таблицю даних в виді текстової інформації, яка є зручною для подальшого візуального аналізу.

Підіб’ємо підсумок вищесказаного.

Числові дані, які представляють собою шукану таблицю функції, розміщені в певному файлі (змінна **`path`**) та повинні бути нами перетворені у об’єкт певного типу, який успадковує основні елементи класу **`MyTable`**.

Алгоритм безпосереднього перетворення інформації з заданого файлу в об’єкт «таблиця даних» буде нами реалізований в конструкторі нового класу, який ми назвемо **`MyTableOfData`**.

Цей алгоритм включатиме декілька етапів.

Перший етап. Читання майбутньої таблиці з файлу у «тимчасовий список» її вузлів.

Після досягнення кінця відповідного файлу ми будемо знати загальну кількість вузлів таблиці і зможемо виділити пам’ять під масив вузлів **`Nodes[]`**.

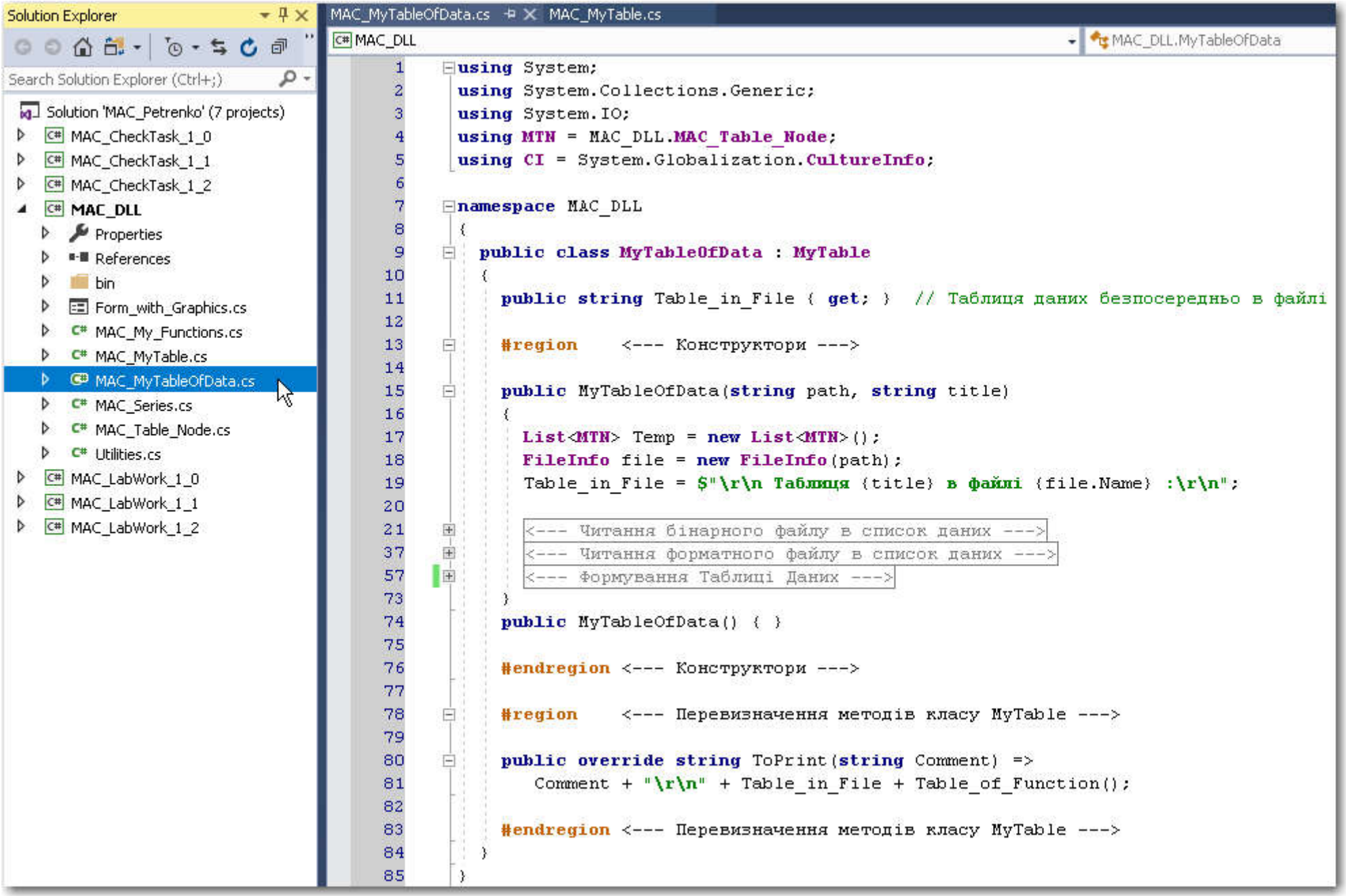
Другий етап. Нема ніякої гарантії, що у «тимчасовому списку» вузли є впорядкованими за зростанням аргументу.

Тому слід виконати відповідне сортування елементів «тимчасового списку», після чого перетворити його в необхідний масив вузлів **`Nodes[]`**.

Після того, як таблиця вузлів впорядкована, слід визначити ті її вузли, які містять найбільше та найменше значення функції на всьому інтервалі зміни аргументу.

Крім цього, нам дуже важливо мати уявлення про числову інформацію, яка записана безпосередньо в початковому файлі даних. Для цього ми додамо в клас **`MyTableOfData`** властивість **`Table_in_File`**.

Загальна структура класу **`MyTableOfData`** наводиться далі:



Алгоритми етапів конструктора Вам слід «розібрати» самостійно:

```
#region    <--- Читання бінарного файлу в список даних --->
if (file.Extension == ".bin")
{
  BinaryReader bin_rdr = new BinaryReader(file.OpenRead()); int i = -1;
  try
  {
    while (true)
    {
      Temp.Add(new MTN(bin_rdr.ReadDouble(), bin_rdr.ReadDouble()));
      Table_in_File += $"{++i,4}" + Temp[i].ToPrint() + "\r\n";
    }
  }
  catch (IOException) { }
  bin_rdr.Close();
}
#endregion <--- Читання бінарного файлу в список даних --->
#region    <--- Читання форматного файлу в список даних --->
if (file.Extension == ".txt")
{
  bool dot_or_comma = CI.CurrentCulture.NumberFormat.NumberDecimalSeparator == ".";
  StreamReader txt_rdr = new StreamReader(file.OpenRead());

  // Читаемо записи з текстового файлу та редагуємо їх
  string[] txt; string line; int i = -1;
  while (!txt_rdr.EndOfStream)
  {
    i++; line = txt_rdr.ReadLine();
    line = dot_or_comma ? line.Replace(",", ".").Trim() : line.Replace(".", ",").Trim();

    txt = line.Split(new char[] { ' ', ';' }, StringSplitOptions.RemoveEmptyEntries);
    Temp.Add(new MTN(Convert.ToDouble(txt[0]), Convert.ToDouble(txt[1])));
    Table_in_File += $"{i,4}" + Temp[i].ToPrint() + "\r\n";
  }
  txt_rdr.Close();
}
#endregion <--- Читання форматного файлу в список даних --->
```

```
57      #region    <--- Формування Таблиці Даних --->
58      if (Temp != null)
59      {
60        Temp.Sort((node_i, node_j) => node_i.X.CompareTo(node_j.X));
61        Nodes = Temp.ToArray();
62        Minimum = new MTN(double.NaN, double.MaxValue);
63        Maximum = new MTN(double.NaN, double.MinValue);
64        for (int i = 0; i < Length; i++)
65        {
66          if (Minimum.F > Nodes[i].F) { Minimum = Nodes[i]; }
67          if (Maximum.F < Nodes[i].F) { Maximum = Nodes[i]; }
68        }
69      }
70      else { Nodes = null; Minimum = null; Maximum = null; }
71      Title = title;
72      #endregion <--- Формування Таблиці Даних --->
```

Після того, як Ви завершили формування всіх вище перелічених програмних компонентів, слід виконати компіляцію проекту **MAC_DLL** та впевнитися у відсутності помилок (при необхідності – виправити помилки).

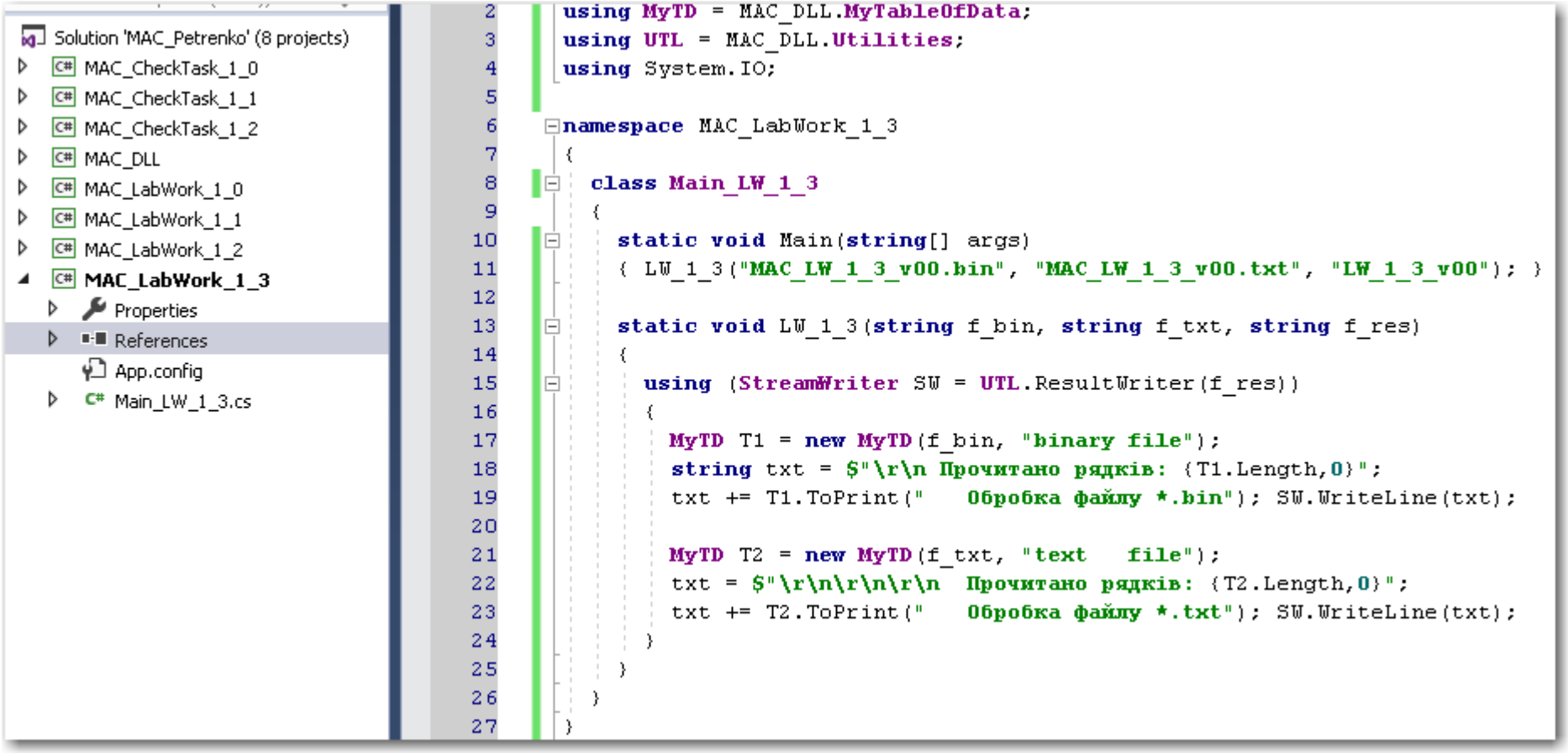
Зверніть увагу на те, що в заголовку файлу **MAC_MyTableOfData** (де директиви **using**) слід вказати клас **System.Globalization.CultureInfo**, за допомогою якого ми зможемо визначити для Вашого комп'ютера: який саме із символів «.» або «,» призначений бути роздільником цілої та дробової частини числа в його десятинній формі:

```
1   using System;
2   using System.Collections.Generic;
3   using System.IO;
4   using MTN = MAC_DLL.MAC_Table_Node;
5   using CI = System.Globalization.CultureInfo;
6
```

Тестування класів `MyTable` та `MyTableOfData`.

Тепер переходимо до проектування консольного додатку, в якому будуть використовуватися спроектовані нами нові програмні компоненти. В робочій простір **`MAC_Petrenko`** додаємо новий проект консольного додатку **`MAC_LabWork_1_3`**.

Налагодження його коду будемо здійснювати із використанням ***тестових*** файлів даних **`MAC_LW_1_3_v00.txt`** і **`MAC_LW_1_3_v00.bin`**. Ці файли повинні бути Вами попередньо скопійовані в папку **`bin\Debug\`** проекту консольного додатку **`MAC_LabWork_1_3`**. Код основної програми **`MAC_LabWork_1_3`** повинен бути таким:




```
2  using MyTD = MAC_DLL.MyTableOfData;
3  using UTL = MAC_DLL.Utilities;
4  using System.IO;
5
6  namespace MAC_LabWork_1_3
7  {
8    class Main_LW_1_3
9    {
10     static void Main(string[] args)
11     { LW_1_3("MAC_LW_1_3_v00.bin", "MAC_LW_1_3_v00.txt", "LW_1_3_v00"); }
12
13     static void LW_1_3(string f_bin, string f_txt, string f_res)
14     {
15       using (StreamWriter SW = UTL.ResultWriter(f_res))
16       {
17         MyTD T1 = new MyTD(f_bin, "binary file");
18         string txt = $"\r\n Прочитано рядків: {T1.Length,0}";
19         txt += T1.ToPrint("   Обробка файлу *.bin"); SW.WriteLine(txt);
20
21         MyTD T2 = new MyTD(f_txt, "text   file");
22         txt = $"\r\n\r\n\r\n  Прочитано рядків: {T2.Length,0}";
23         txt += T2.ToPrint("   Обробка файлу *.txt"); SW.WriteLine(txt);
24       }
25     }
26   }
27 }
```

Результати (файл **LW_1_3_v00 Petrenko_I_M.txt**):

```
 4    Прочитано рядків: 15   Обробка файлу *.bin
 5
 6    Таблиця binary file в файлі MAC_LW_1_3_v00.bin :
 7      0  (   -2,3428571429,    6,6862274052  )
 8      1  (    0,3071428571,   -1,7195455539  )
 9      2  (   -1,2071428571,    6,9410098397  )
10      3  (   -3,1000000000,   -1,5810000000  )
11      4  (   -0,0714285714,    0,4333090379  )
12      5  (    2,2000000000,    2,2880000000  )
13      6  (    1,8214285714,   -1,5681942420  )
14      7  (    1,0642857143,   -4,0474894315  )
15      8  (    0,6857142857,   -3,3216559767  )
16      9  (   -1,9642857143,    8,0650965743  )
17     10  (    1,4428571429,   -3,5715131195  )
18     11  (   -2,7214285714,    3,5793728134  )
19     12  (   -1,5857142857,    8,0415131195  )
20     13  (   -0,4500000000,    2,8113750000  )
21     14  (   -0,8285714286,    5,0891195335  )
22
23     Таблиця функції binary file :
24      0  (   -3,1000000000,   -1,5810000000  )
25      1  (   -2,7214285714,    3,5793728134  )
26      2  (   -2,3428571429,    6,6862274052  )
27      3  (   -1,9642857143,    8,0650965743  )
28      4  (   -1,5857142857,    8,0415131195  )
29      5  (   -1,2071428571,    6,9410098397  )
30      6  (   -0,8285714286,    5,0891195335  )
31      7  (   -0,4500000000,    2,8113750000  )
32      8  (   -0,0714285714,    0,4333090379  )
33      9  (    0,3071428571,   -1,7195455539  )
34     10  (    0,6857142857,   -3,3216559767  )
35     11  (    1,0642857143,   -4,0474894315  )
36     12  (    1,4428571429,   -3,5715131195  )
37     13  (    1,8214285714,   -1,5681942420  )
38     14  (    2,2000000000,    2,2880000000  )
39
40     x = [  -3,100000000000 :   2,200000000000 ]   x_Reg =   5,300000000000
41
42    Min  (    1,064285714286,   -4,047489431487 )
43    Max  (   -1,964285714286,    8,065096574344 )  f_Reg =  12,112586005831
44
45
46     Прочитано рядків: 25   Обробка файлу *.txt
47
48    Таблиця text   file в файлі MAC_LW_1_3_v00.txt :
49      0  (    1,5375000000,   -3,2265878906  )
50      1  (   -0,2291666667,    1,4154821325  )
51      2  (    1,9791666667,   -0,2053041811  )
52      3  (   -0,0083333333,    0,0500688657  )
53      4  (    0,2125000000,   -1,2202480469  )
54      5  (    0,8750000000,   -3,8144531250  )
```

```
 54      5   (    0,8750000000,   -3,8144531250  )
 55      6   (   -1,3333333333,    7,4074074074  )
 56      7   (   -3,1000000000,   -1,5810000000  )
 57      8   (    0,6541666667,   -3,2171257957  )
 58      9   (    0,4333333333,   -2,3308518519  )
 59     10   (   -2,4375000000,    6,0842285156  )
 60     11   (   -0,4500000000,    2,8113750000  )
 61     12   (   -2,8791666667,    1,6974586950  )
 62     13   (   -1,5541666667,    7,9864469763  )
 63     14   (   -0,6708333333,    4,1731307147  )
 64     15   (    1,3166666667,   -3,8838009259  )
 65     16   (   -0,8916666667,    5,4361325231  )
 66     17   (   -1,1125000000,    6,5357636719  )
 67     18   (    1,0958333333,   -4,0582170862  )
 68     19   (   -2,2166666667,    7,3217731481  )
 69     20   (    2,2000000000,    2,2880000000  )
 70     21   (   -1,7750000000,    8,2082656250  )
 71     22   (   -1,9958333333,    8,0082466001  )
 72     23   (   -2,6583333333,    4,2309959491  )
 73     24   (    1,7583333333,   -2,0219612269  )
 74
 75     Таблиця функції text   file :
 76      0   (   -3,1000000000,   -1,5810000000  )
 77      1   (   -2,8791666667,    1,6974586950  )
 78      2   (   -2,6583333333,    4,2309959491  )
 79      3   (   -2,4375000000,    6,0842285156  )
 80      4   (   -2,2166666667,    7,3217731481  )
 81      5   (   -1,9958333333,    8,0082466001  )
 82      6   (   -1,7750000000,    8,2082656250  )
 83      7   (   -1,5541666667,    7,9864469763  )
 84      8   (   -1,3333333333,    7,4074074074  )
 85      9   (   -1,1125000000,    6,5357636719  )
 86     10   (   -0,8916666667,    5,4361325231  )
 87     11   (   -0,6708333333,    4,1731307147  )
 88     12   (   -0,4500000000,    2,8113750000  )
 89     13   (   -0,2291666667,    1,4154821325  )
 90     14   (   -0,0083333333,    0,0500688657  )
 91     15   (    0,2125000000,   -1,2202480469  )
 92     16   (    0,4333333333,   -2,3308518519  )
 93     17   (    0,6541666667,   -3,2171257957  )
 94     18   (    0,8750000000,   -3,8144531250  )
 95     19   (    1,0958333333,   -4,0582170862  )
 96     20   (    1,3166666667,   -3,8838009259  )
 97     21   (    1,5375000000,   -3,2265878906  )
 98     22   (    1,7583333333,   -2,0219612269  )
 99     23   (    1,9791666667,   -0,2053041811  )
100     24   (    2,2000000000,    2,2880000000  )
101
102     x = [  -3,100000000000 :   2,200000000000 ]   x_Reg =   5,300000000000
103
104    Min   (    1,095833333333,   -4,058217086227 )
105    Max   (   -1,775000000000,    8,208265625000 )  f_Reg =  12,266482711227
```

Зверніть увагу на те, що бінарний і текстовий файли містять в собі числову інформацію, яка відповідає одній і тій ж самій математичній функції $f(x)$.

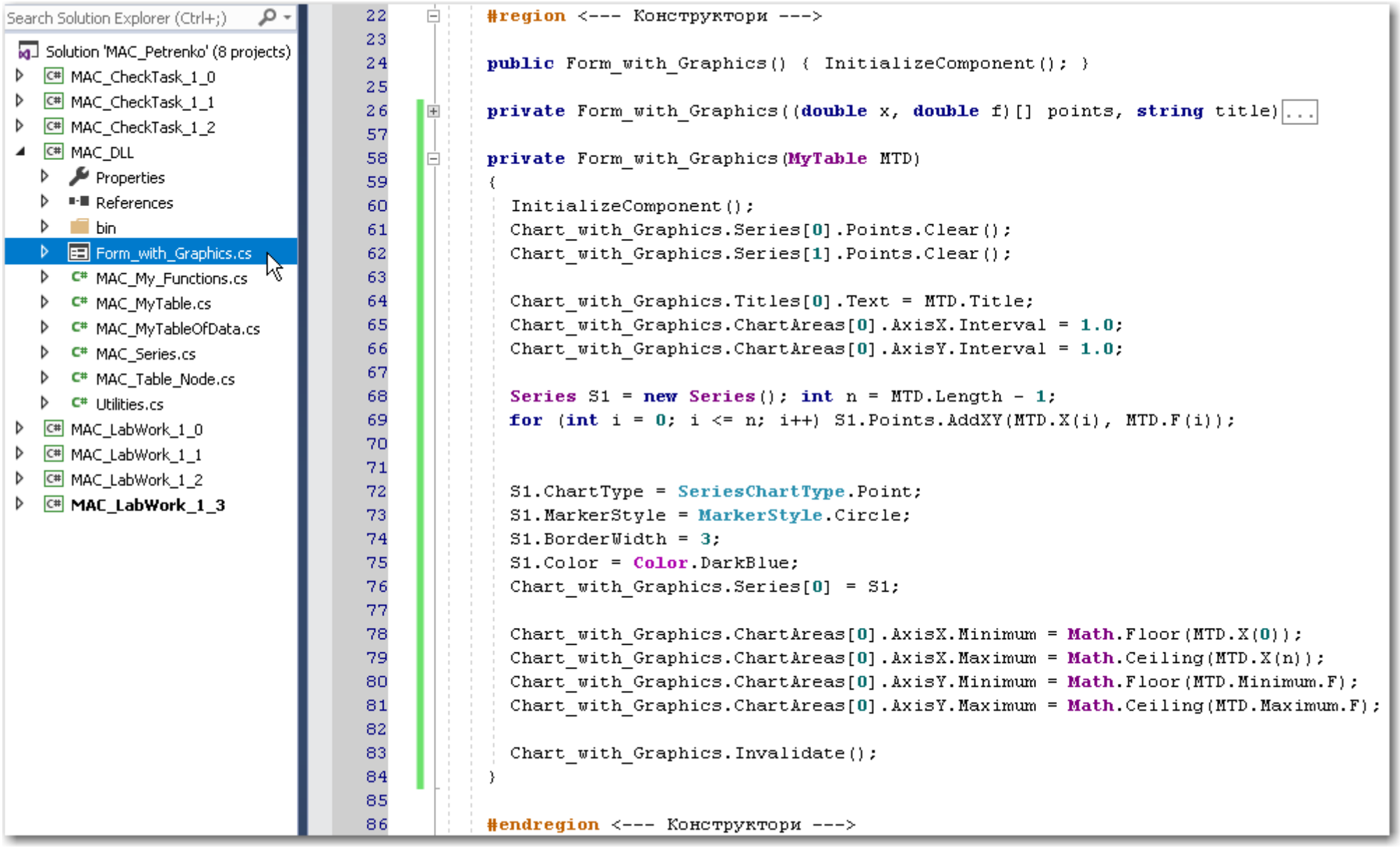
Інтервали зміни аргументу $x \in [-3.1, +2.2]$ у «Таблиць даних в файлі» також співпадають, але кількість рядків в цих таблицях різна, а самі вузли в них розташовані в довільному порядку.

Завершимо дану лабораторну роботу побудовою графіку тієї функції, яка представлена у вигляді таблиці даних.

Для цього, по-перше, нам слід розширити функціонал класу **`Form_with_Graphics`**.

Додамо новий метод та новий конструктор до цього класу (відповідні фрагменти коду вам будуть надані у директорії Кодів на Гугл-диску – файл **`Form_with_Graphics for LW_1_3.cs`**):

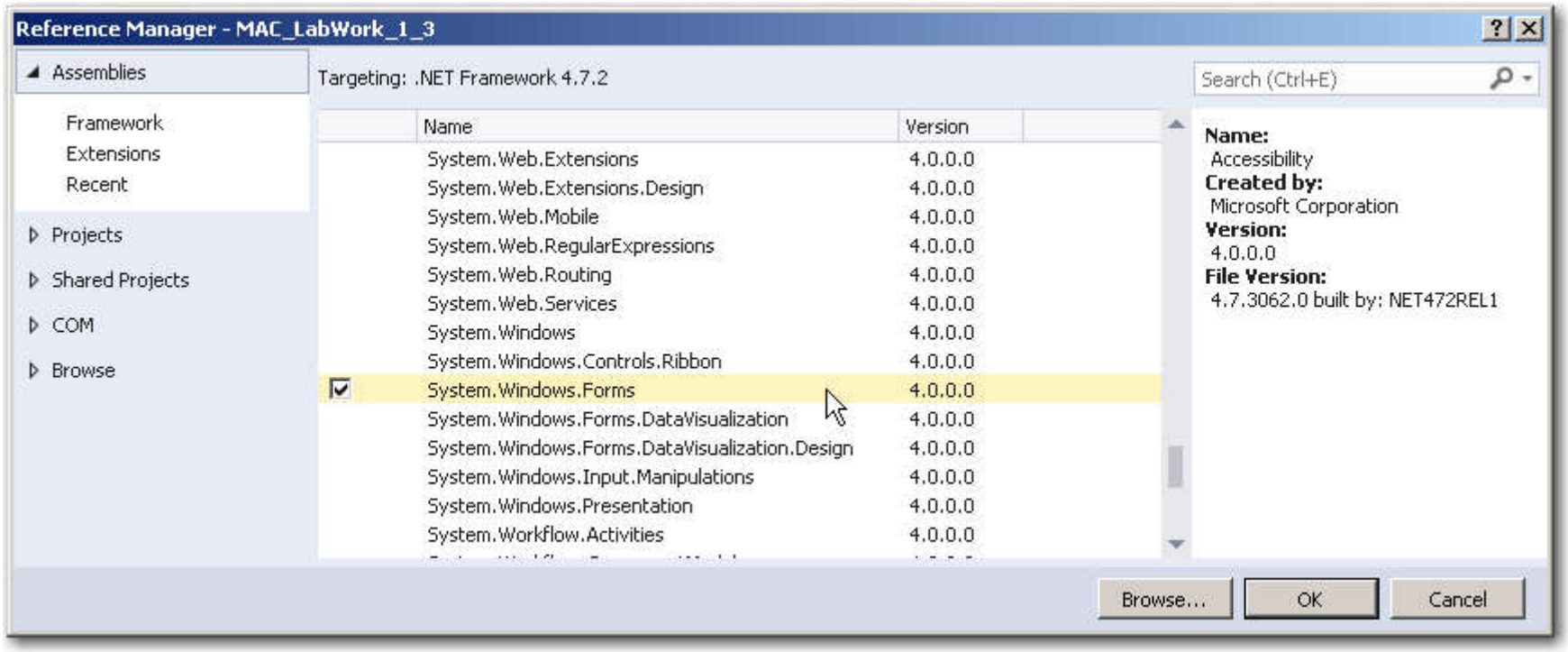
```
88      #region    <--- Методи --->
89
90      public static void SingleGraphic(double[] X, double[] Y, string Title, int Nx, int Ny)...
98
99      public static void SingleGraphic(MyTable table, int Nx, int Ny)
100     {
101       if (table == null) return;
102       Form_with_Graphics SingleG = new Form_with_Graphics(table);
103       SingleG.filename = table.Title;
104       SingleG.StartPosition = FormStartPosition.Manual;
105       SingleG.Location = new Point(Nx, Ny);
106       SingleG.ShowDialog();
107     }
108
109     #endregion <--- Методи --->
```


Search Solution Explorer (Ctrl+;)
Solution 'MAC_Petrenko' (8 projects)
MAC_CheckTask_1_0
MAC_CheckTask_1_1
MAC_CheckTask_1_2
MAC_DLL
Properties
References
bin
Form_with_Graphics.cs
MAC_My_Functions.cs
MAC_MyTable.cs
MAC_MyTableOfData.cs
MAC_Series.cs
MAC_Table_Node.cs
Utilities.cs
MAC_LabWork_1_0
MAC_LabWork_1_1
MAC_LabWork_1_2
MAC_LabWork_1_3
22 #region <--- Конструктори --->
23
24 public Form_with_Graphics() { InitializeComponent(); }
25
26 private Form_with_Graphics((double x, double f)[] points, string title)...
57
58 private Form_with_Graphics(MyTable MTD)
59 {
60 InitializeComponent();
61 Chart_with_Graphics.Series[0].Points.Clear();
62 Chart_with_Graphics.Series[1].Points.Clear();
63
64 Chart_with_Graphics.Titles[0].Text = MTD.Title;
65 Chart_with_Graphics.ChartAreas[0].AxisX.Interval = 1.0;
66 Chart_with_Graphics.ChartAreas[0].AxisY.Interval = 1.0;
67
68 Series S1 = new Series(); int n = MTD.Length - 1;
69 for (int i = 0; i <= n; i++) S1.Points.AddXY(MTD.X(i), MTD.F(i));
70
71
72 S1.ChartType = SeriesChartType.Point;
73 S1.MarkerStyle = MarkerStyle.Circle;
74 S1.BorderWidth = 3;
75 S1.Color = Color.DarkBlue;
76 Chart_with_Graphics.Series[0] = S1;
77
78 Chart_with_Graphics.ChartAreas[0].AxisX.Minimum = Math.Floor(MTD.X(0));
79 Chart_with_Graphics.ChartAreas[0].AxisX.Maximum = Math.Ceiling(MTD.X(n));
80 Chart_with_Graphics.ChartAreas[0].AxisY.Minimum = Math.Floor(MTD.Minimum.F);
81 Chart_with_Graphics.ChartAreas[0].AxisY.Maximum = Math.Ceiling(MTD.Maximum.F);
82
83 Chart_with_Graphics.Invalidate();
84 }
85
86 #endregion <--- Конструктори --->

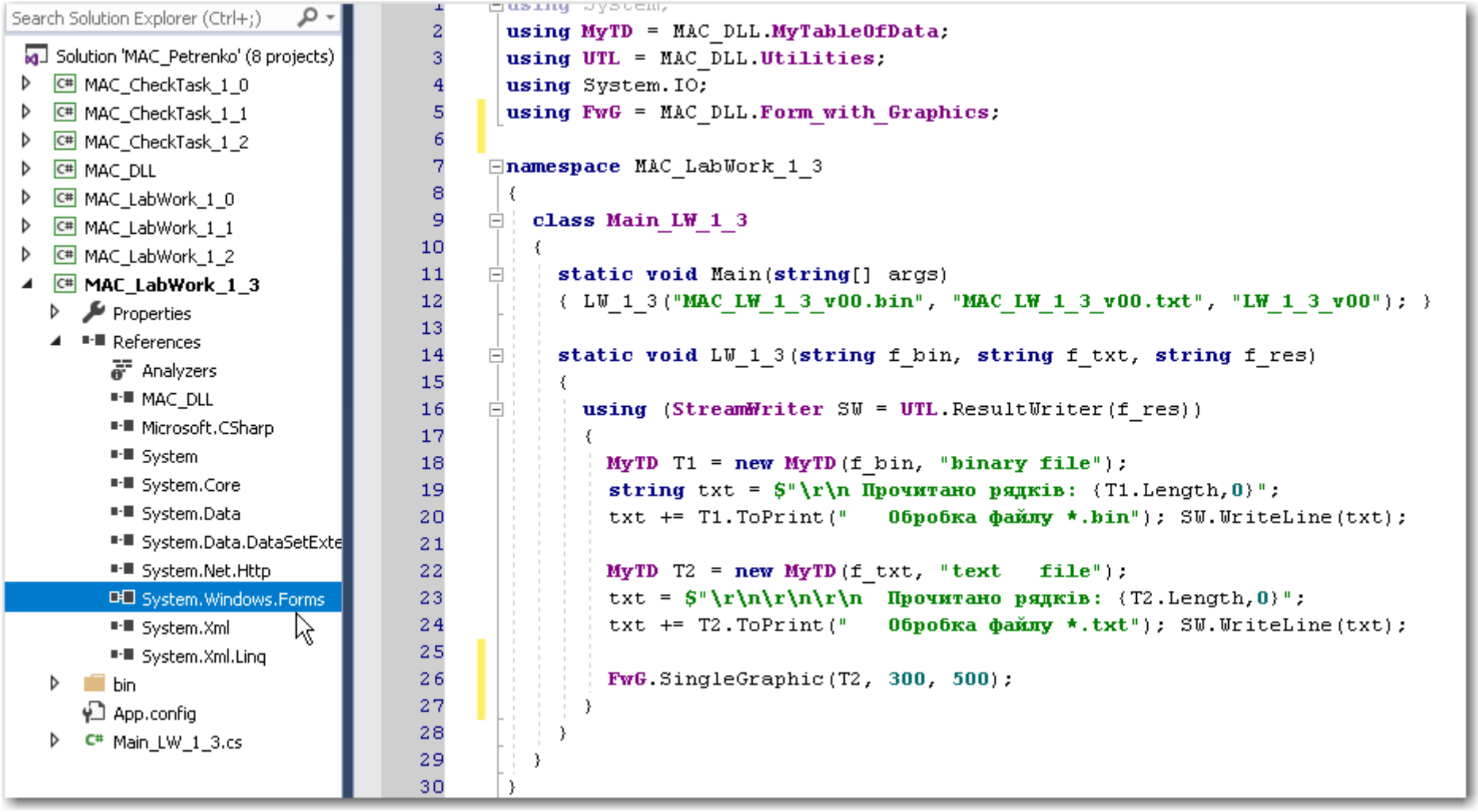
Тепер слід додати відповідну директиву до основної програми:

```
using FwG = MAC_DLL.Form_with_Graphics;
```

та приєднати до переліку посилань системну бібліотеку **`System.Windows.Forms`**:



Маємо оновлений стан основної програми:

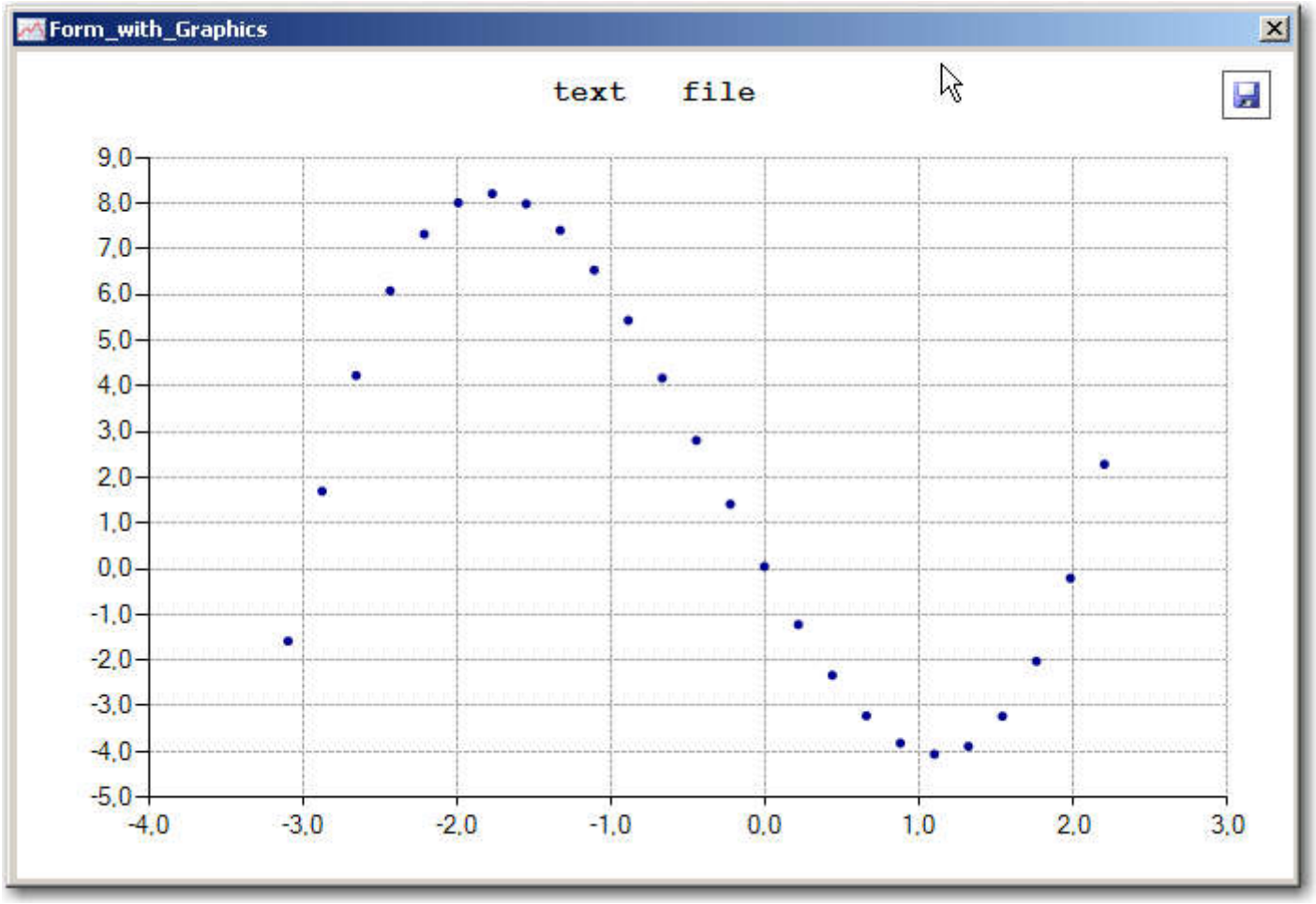


Ми спостерігаємо, що взагалі-то ми додали до основної програми лише три рядки коду.

Але отримали якісну підтримку щодо аналізу отриманих результатів – графік таблиці функції.

Спробуємо виконати програму.

Маємо наступний результат:



Тепер ми маємо можливість візуального аналізу функції та зберігання цього графічного об’єкту на диску з метою подальшого використання його у відповідних звітах.

Після успішного виконання даної лабораторної роботи із використанням тестових файлів Вам слід виконати заключний етап з індивідуальними наборами даних у відповідності із Вашим особистим Варіантом.

Контрольний прорахунок слід виконати з файлами даних **MAC_LW_1_3_vNN.txt** і **MAC_LW_1_3_vNN.bin**, де **NN** – номер Вашого **особистого** варіанта.

Ці файли даних слід скопіювати в директорію **bin\Debug**, в якій розташований відповідний файл **MAC_Lab_Work_1_3.exe**.

Не забудьте включити в файл результатів код основної програми лабораторної роботи, наприклад, в такий спосіб:

```
1   using ...
8
9   namespace MAC_LabWork_1_3
10  {
11      class Main_LW_1_3
12      {
13          static void Main(string[] args)
14          {
15              //LW_1_3("MAC_LW_1_3_v00.bin", "MAC_LW_1_3_v00.txt", "LW_1_3_v00");
16              LW_1_3("MAC_LW_1_3_v20.bin", "MAC_LW_1_3_v20.txt", "LW_1_3_v20");
17          }
18          static void LW_1_3(string f_bin, string f_txt, string f_res)
19          {
20              using (StreamWriter SW = UTL.ResultWriter(f_res, "Main_LW_1_3.cs"))...
32          }
33      }
34  }
```

Крім текстового файлу результатів на перевірку слід відправляти і файл із графічним зображенням функції.

Методика виконання завдання на самостійну роботу № 1.3

«Побудування таблиці функції із заданою точністю»

Постановка задачі. Побудувати алгоритм та налагодити програму обчислення таблиці функції, яку задано в виді аналітичного зображення (нескінченний степеневий ряд):

$$f_0(x) = \sum_{k=0}^{\infty} \frac{(-1)^k \cdot \left[\cos\left(\frac{b}{k+1}x\right) + \sin\left(\frac{b}{k+a}x\right)\right]}{4 \cdot (2k+a) \cdot (k+1)!} \left(\frac{11x}{7}\right)^{k+1}, \qquad \text{(с.1.3.1)}$$

на інтервалі $x \in [\, x_0, x_n \,]$ з заданою кількістю рядків $(n+1)$ та абсолютною похибкою ε, де a і b – деякі задані значення дійсних параметрів даної функції.

На основі даних, які розміщені в таблиці функції, встановити найбільше та найменше значення функції на всьому заданому інтервалі $[\, x_0, x_n \,]$ зміни аргументу x.

Етап 1.

Підготовчі дії. В існуючому робочому просторі **MAC_Petrenko** згенеруємо (додамо) новий проект **MAC_CheckTask_1_3** консольного додатку, який призначимо «стартовим» проектом. Додамо до переліку **References** проекту **MAC_CheckTask_1_3** посилання на проект динамічної бібліотеки **MAC_DLL**.

Повертаємось до проекту динамічної бібліотеки **MAC_DLL** і приступаємо до розробки програмної реалізації нового класу **MyTableOfFunction**, який будемо створювати у відповідному файлі **MAC_MyTableOfFunction.cs**.

Власно створення таблиці функції $f = f(x)$, повинно бути реалізоване саме в тілі конструктора нового класу – **MyTableOfFunction**, який є нащадком від класу **MyTable**.

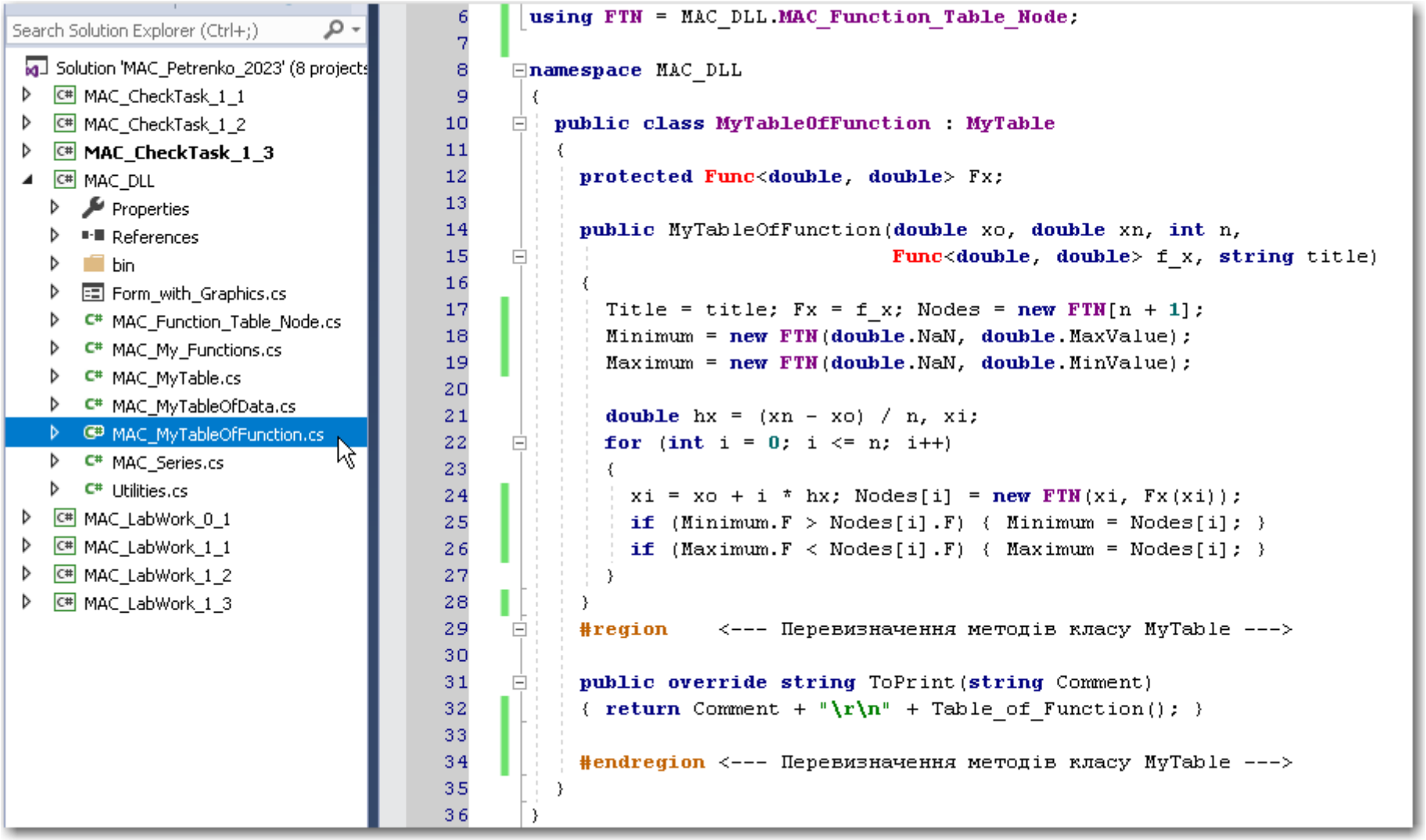
Вказаний конструктор повинен приймати конкретну функцію $f(x)$ в вигляді фактичного параметра.

Метод, який реалізовуватиме обчислення значень функцій $f(x)$ у відповідності до зображення (с.1.3.1), повинен бути розміщеним в окремому класі **MyFunctions** динамічної бібліотеки **MAC_DLL**.

І ми це вже зробили під час виконання лабораторної роботи 1.2.

Клас `MyTableOfFunction`. За умовами завдання конструктор `MyTableOfFunction()` повинен приймати в якості вхідного параметра метод з сигнатурою, яка відповідає функції (1) від однієї дійсної змінної, тобто $f = f(x)$.

Скористуємось для цього узагальненим делегатом. Пропонується наступна реалізація класу:




```
6   using FTN = MAC_DLL.MAC_Function_Table_Node;
7
8   namespace MAC_DLL
9   {
10    public class MyTableOfFunction : MyTable
11    {
12      protected Func<double, double> Fx;
13
14      public MyTableOfFunction(double xo, double xn, int n,
15                               Func<double, double> f_x, string title)
16      {
17        Title = title; Fx = f_x; Nodes = new FTN[n + 1];
18        Minimum = new FTN(double.NaN, double.MaxValue);
19        Maximum = new FTN(double.NaN, double.MinValue);
20
21        double hx = (xn - xo) / n, xi;
22        for (int i = 0; i <= n; i++)
23        {
24          xi = xo + i * hx; Nodes[i] = new FTN(xi, Fx(xi));
25          if (Minimum.F > Nodes[i].F) { Minimum = Nodes[i]; }
26          if (Maximum.F < Nodes[i].F) { Maximum = Nodes[i]; }
27        }
28      }
29      #region    <--- Перевизначення методів класу MyTable --->
30
31      public override string ToPrint(string Comment)
32      { return Comment + "\r\n" + Table_of_Function(); }
33
34      #endregion <--- Перевизначення методів класу MyTable --->
35    }
36  }
```

Алгоритм прорахунку таблиці функції, коли її задано аналітичним зображенням, достатньо простий.

Безпосередньо в тілі конструктора класу **MyTableOfFunction** виконується цикл за значеннями аргументу від лівої границі x_0 до правої границі x_n області визначення $x \in [\, x_0, x_n \,]$ із сталим шагом $h = (x_n - x_0) / n$.

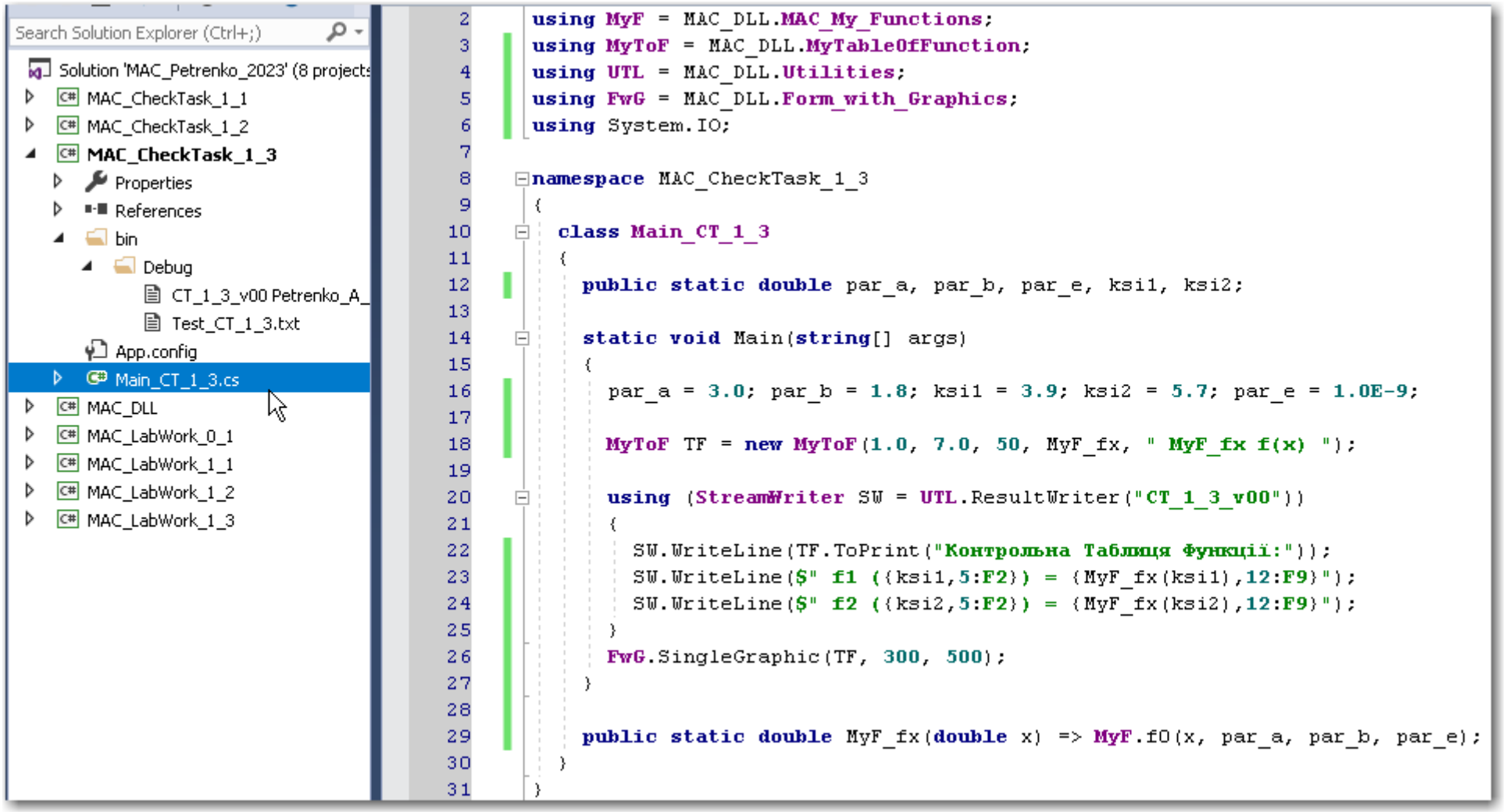
В цьому ж циклі одночасно виконується вибір вузлів таблиці з найменшим та найбільшим значенням функції.

Також перевизначимо віртуальний метод **ToPrint()**.

За умовою завдання, параметри конструктора **MyTableOfFunction()** мають зміст:

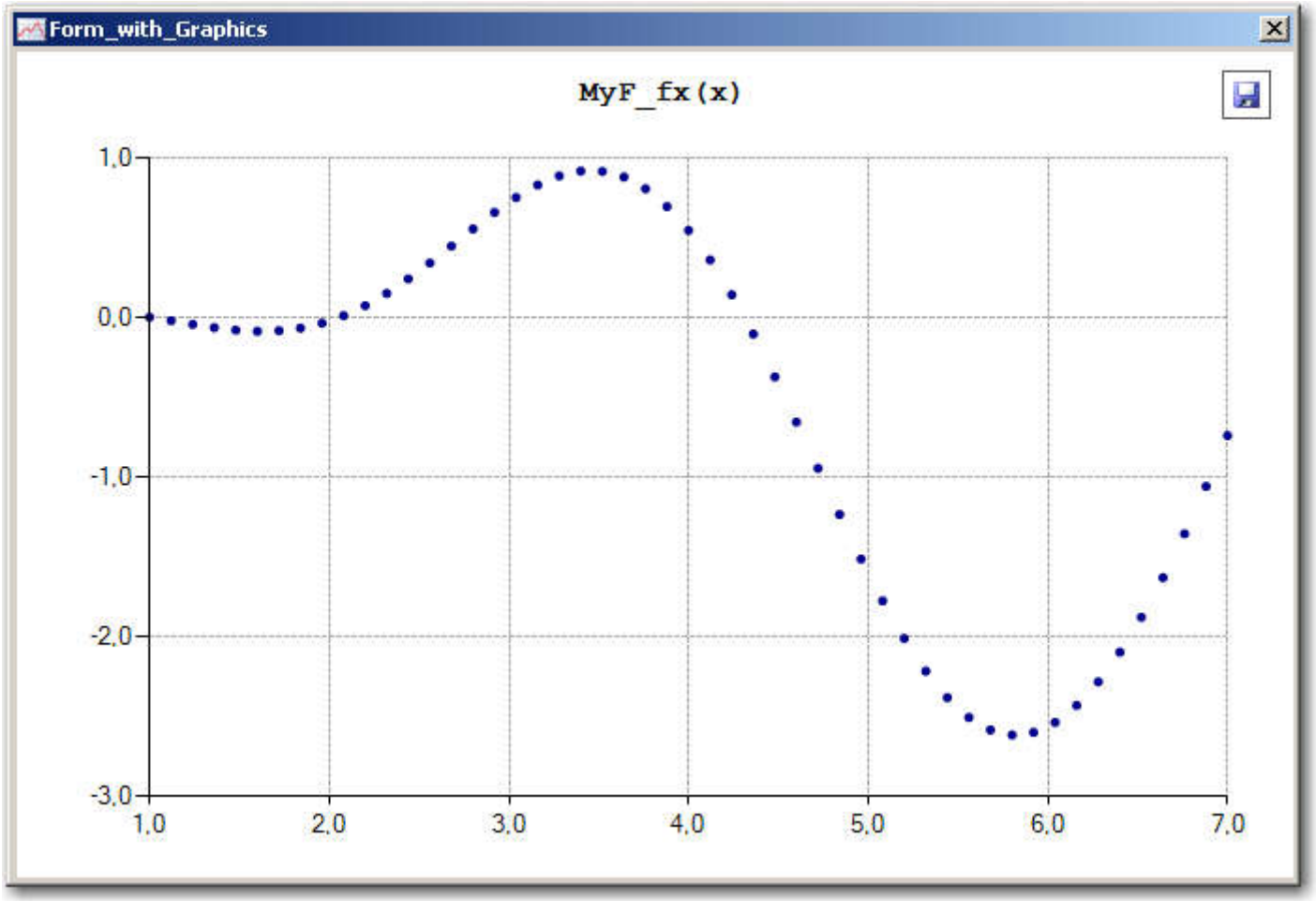
- **xo** – ліва границя інтервалу, на якому обчислюється таблиця функції;
- **xn** – права границя цього інтервалу;
- **n** – кількість рівних інтервалів, на які треба розбити загальний відрізок $[\, x_0, x_n \,]$;
- **f_x** – делегат, який визначає сигнатуру метода, що безпосередньо обчислює значення заданої функції;
- **title** – строкова змінна, яка містить «ідентифікатор» таблиці функції.

Здійснимо першу перевірку та обчислимо таблицю заданої функції (1) в основній програмі **MAC_CheckTask_1_3**:



Маємо наступний контрольний результат:

```
CT_1_3_v00 Petrenko_I_M.txt  data : 02.03.20

Контрольна Таблиця Функції:

  Таблиця функції  MyF_fx(x)  :
   0  (    1,0000000000,    -0,0005153151  )
   1  (    1,1200000000,    -0,0227539923  )
   2  (    1,2400000000,    -0,0454573978  )
   3  (    1,3600000000,    -0,0659410809  )
   4  (    1,4800000000,    -0,0813182888  )
   5  (    1,6000000000,    -0,0886920525  )
   6  (    1,7200000000,    -0,0853595300  )
   7  (    1,8400000000,    -0,0690167634  )
   8  (    1,9600000000,    -0,0379515945  )
   9  (    2,0800000000,     0,0087872304  )
  10  (    2,2000000000,     0,0712557579  )
  11  (    2,3200000000,     0,1485246995  )
  12  (    2,4400000000,     0,2386406411  )
  13  (    2,5600000000,     0,3386431091  )
  14  (    2,6800000000,     0,4446390331  )
  15  (    2,8000000000,     0,5519330086  )
  16  (    2,9200000000,     0,6552086377  )
  17  (    3,0400000000,     0,7487532819  )
  18  (    3,1600000000,     0,8267159654  )
  19  (    3,2800000000,     0,8833860648  )
  20  (    3,4000000000,     0,9134789409  )
  21  (    3,5200000000,     0,9124139102  )
  22  (    3,6400000000,     0,8765699601  )
  23  (    3,7600000000,     0,8035053962  )
  24  (    3,8800000000,     0,6921291774  )
  25  (    4,0000000000,     0,5428139298  )
  26  (    4,1200000000,     0,3574434628  )
  27  (    4,2400000000,     0,1393908791  )
  28  (    4,3600000000,    -0,1065730736  )
  29  (    4,4800000000,    -0,3744381223  )
  30  (    4,6000000000,    -0,6571722337  )
  31  (    4,7200000000,    -0,9469889508  )
  32  (    4,8400000000,    -1,2356514479  )
  33  (    4,9600000000,    -1,5147983084  )
  34  (    5,0800000000,    -1,7762746470  )
  35  (    5,2000000000,    -2,0124516605  )
  36  (    5,3200000000,    -2,2165180448  )
  37  (    5,4400000000,    -2,3827279704  )
  38  (    5,5600000000,    -2,5065924158  )
  39  (    5,6800000000,    -2,5850035091  )
  40  (    5,8000000000,    -2,6162850033  )
  41  (    5,9200000000,    -2,6001659301  )
  42  (    6,0400000000,    -2,5376786602  )
  43  (    6,1600000000,    -2,4309867997  )
  44  (    6,2800000000,    -2,2831523666  )
  45  (    6,4000000000,    -2,0978553044  )
  46  (    6,5200000000,    -1,8790813963  )
  47  (    6,6400000000,    -1,6307968639  )
  48  (    6,7600000000,    -1,3566292221  )
  49  (    6,8800000000,    -1,0595742484  )
  50  (    7,0000000000,    -0,7417481572  )

  x = [    1,000000000000 :    7,000000000000 ]
  x_Reg =    6,000000000000

 Min  (    5,800000000000,    -2,616285003267 )
 Max  (    3,400000000000,     0,913478940863 )
 f_Reg =    3,529763944130

 f1 ( 3,90) =   0,669845849
 f2 ( 5,70) = -2,593514333
```



Візуальний аналіз таблиці та її графіка підтверджує правильність програмного визначення вузлів з найбільшим та найменшим значеннями функції, а також довжин області визначення та області значень для даної функції.

Підіб’ємо попередні підсумки виконаної роботи.

Зараз ми маємо абстрактний клас **MyTable**, який описує узагальнену сукупність числових даних типу «Таблиця» і алгоритми (методи та властивості) їх обробки незалежно від способу формування самої таблиці.

На основі цього абстрактного класу ми створили два класи-нащадки **MyTableOfData** і **MyTableOfFunction**, які формують таблицю двома принципово різними способами.

При цьому класи-нащадки користуються загальними методами і властивостями класу **MyTable**, та за необхідністю здатні перевизначити їх.

Ми й в подальшому будемо притримуватися такої стратегії при формуванні нових властивостей та методів для класів, які зазначені вище.

Алгоритми, які є загальними для двох типів таблиць, ми будемо розміщувати в батьківському класі **MyTable**.

А всі специфічні алгоритми – в формі власних членів в класах-нащадках.

Продемонструємо сказане наступним чином.

Результати виконання двох останніх завдань **MAC_LabWork_1_3** і **MAC_CheckTask_1_3** містили достатньо великі об’єми числової інформації, яка виводилася безпосередньо у вікно консольного додатку.

Перегляд і аналіз результатів в такій формі неможна признати зручними, в особливості, коли її треба зберігати з метою співставлення з подібними результатами.

Очевидним (зручнішим) є спосіб збереження результатів в виді текстового іменованого файлу, який, окрім всього іншого, можна включити у відповідний проект робочого простору в якості його компонента, який постійно оновлюється.

Оскільки спосіб збереження результатів обробки таблиці не залежить від способу генерації її числових даних, метод, який буде нам забезпечувати виведення результатів в файл, ми помістимо саме в абстрактному класі та зробимо віртуальним, аби за необхідністю його можна було б і перевизначити.

Основним аргументом цього методу є повний шлях до файлу (параметр **path**), в якому планується зберігати результати обчислень. Якщо шлях до файлу містить тільки його м.'я, то файл буде розміщено в робочій директорії файлу консольного додатку даного проекту.

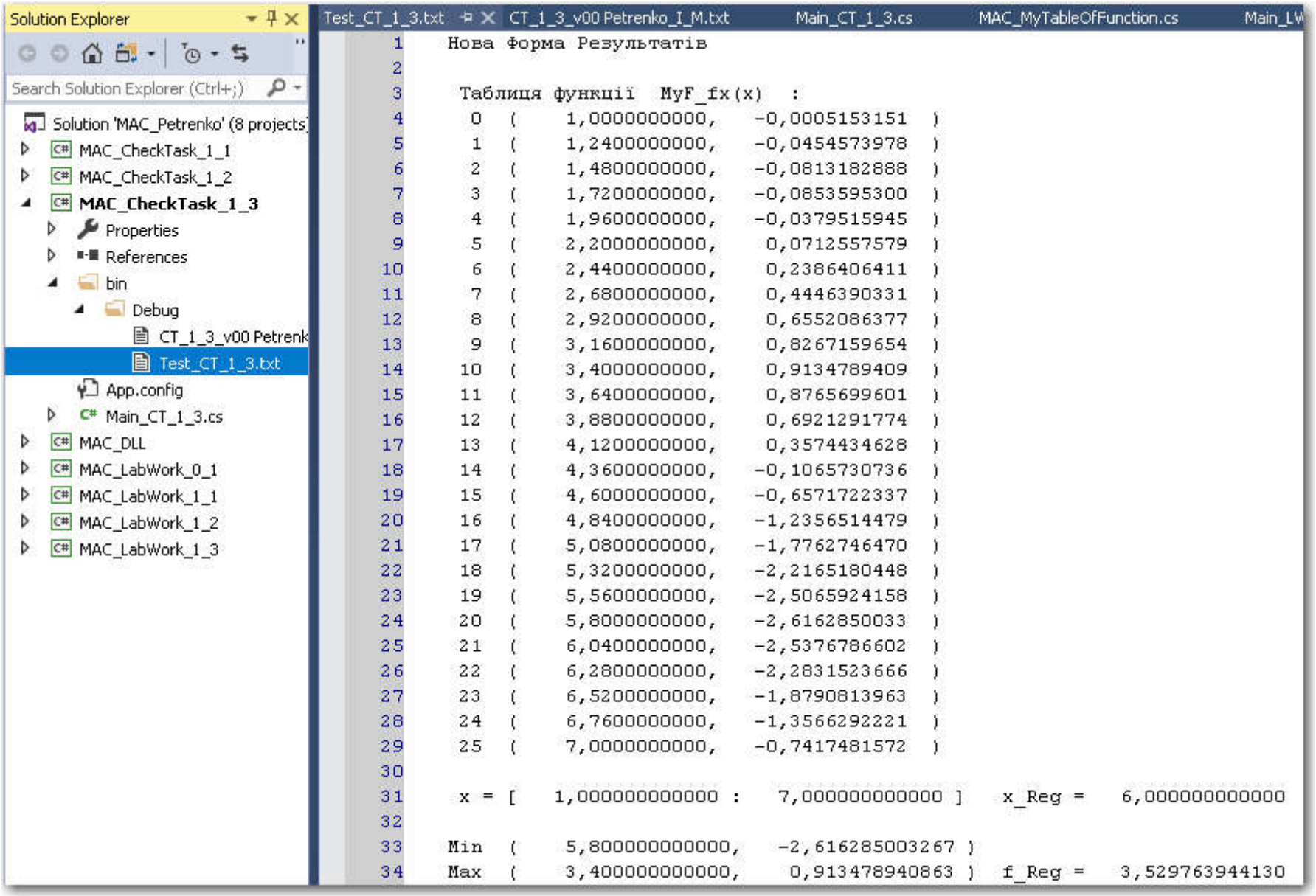
Якщо м.'я не зазначено взагалі – обирається м.'я «за умовчанням» – **My_Table.txt**.

Зменшимо кількість вузлів таблиці до **25**, аби скоротити загальний об'єм таблиці результатів (це зараз не суттєво).

Додаємо виклик метода **To_txt_File()** в код додатку **MAC_CheckTask_1_3**:

```
18      static void Main(string[] args)
19      {
20        par_a = 3.0; par_b = 1.8; ksi1 = 3.9; ksi2 = 5.7; par_e = 1.0E-9;
21
22        MyToF TF = new MyToF(1.0, 7.0, 25, MyF_fx, " MyF_fx(x) ");
23
24        using (StreamWriter SW = UTL.ResultWriter("CT_1_3_v00"))
25        {
26          SW.WriteLine(TF.ToPrint("Контрольна Таблиця Функції:"));
27          SW.WriteLine($" f1 ({ksi1,5:F2}) = {MyF_fx(ksi1),12:F9}");
28          SW.WriteLine($" f2 ({ksi2,5:F2}) = {MyF_fx(ksi2),12:F9}");
29        }
30        TF.To_txt_File("Test_CT_1_3.txt", " Нова Форма Результатів");
31        //FwG.SingleGraphic(TF, 300, 500);
32      }
```

Файл **`Test_CT_1_3.txt`** приєднуємо до проекту та відкриваємо для перегляду:



Етап 2. Переходимо до першої задачі з математичної обробки даних в таблиці функції.

З курсу математичного аналізу відомо, що якщо функція $f(x)$ є строго монотонною на інтервалі $[x_{i-1}, x_i]$ і виконується нерівність $f(x_{i-1}) \cdot f(x_i) < 0$, то на цьому інтервалі існує точка з координатою x_*, в якій функція стає нулем, тобто $f(x_*) = 0$.

Поставимо перед собою задачу програмним шляхом встановити для створеної вже таблиці всі ті інтервали $[x_{i-1}, x_i]$ змінення аргументу, на кінцях яких функція $f(x)$ має різні знаки, тобто виконується саме ця нерівність $f(x_{i-1}) \cdot f(x_i) < 0$.

Вочевидь, що ми не можемо наперед вгадати кількість таких інтервалів.

Ми будемо змушені послідовно «обробляти» вузли таблиці і визначати ті з них, для яких будуть виконуватися умови, які вказані вище. Як тільки такий інтервал $[x_{i-1}, x_i]$ буде знайдений – ми повинні зберегти інформацію про нього в деякій змінній (в об'єкті) та перейти до наступних вузлів таблиці.

Для подібного «збору» інформації підійде тип даних – список.

Але нам ще належить узагальнити поняття «інтервал, який містить нуль функції» в виді деякого власного типу даних (класу) і визначити кількість та зміст його членів.

Будемо виходити з того, що інтервал, який містить нуль функції, описується трьома дійсними числами: це координати кінців інтервалу $[x_{i-1}, x_i]$ і власно координата x_*, в якій функція стає нулем, тобто $f(x_*) = 0$.

В чисельних методах координата x_* обчислюється наближено (із застосуванням певного алгоритму) з деякою похибкою, яка характеризує «якість» значення x_*, яке було обчислене. Цю похибку можна ототожнити із значенням $\varepsilon = |f(x_*)|$.

Чим менша ця величина, тим «якісніше» значення x_*, яке було нами обчислене.

Ми додамо цю величину в якості члена в новий клас, який будемо зараз створювати.

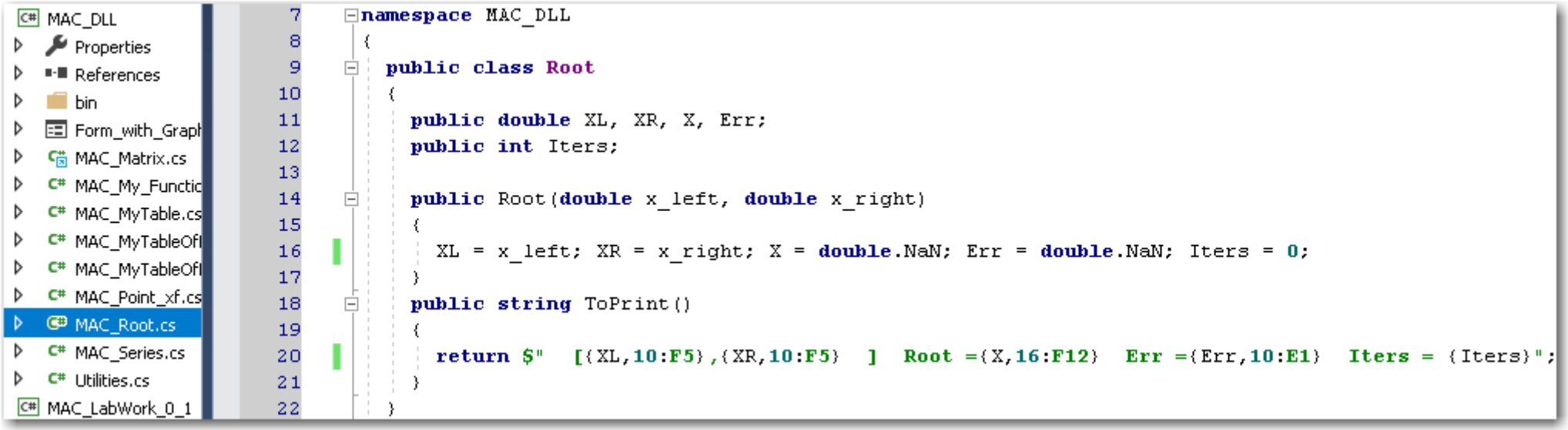
Крім цього, алгоритми уточнення нулів функції, як правило, здійснюються ітераційним шляхом.

Нас буде цікавити кількість ітерацій, які були витрачені тим або іншим чисельним методом на обчислення нуля функції із заданою похибкою. Вважається, що чим менше таких ітерацій – тим кращий чисельний метод (висока швидкість збігання).

Кількість ітерацій – яке є цілим числом – також буде членом нового класу.

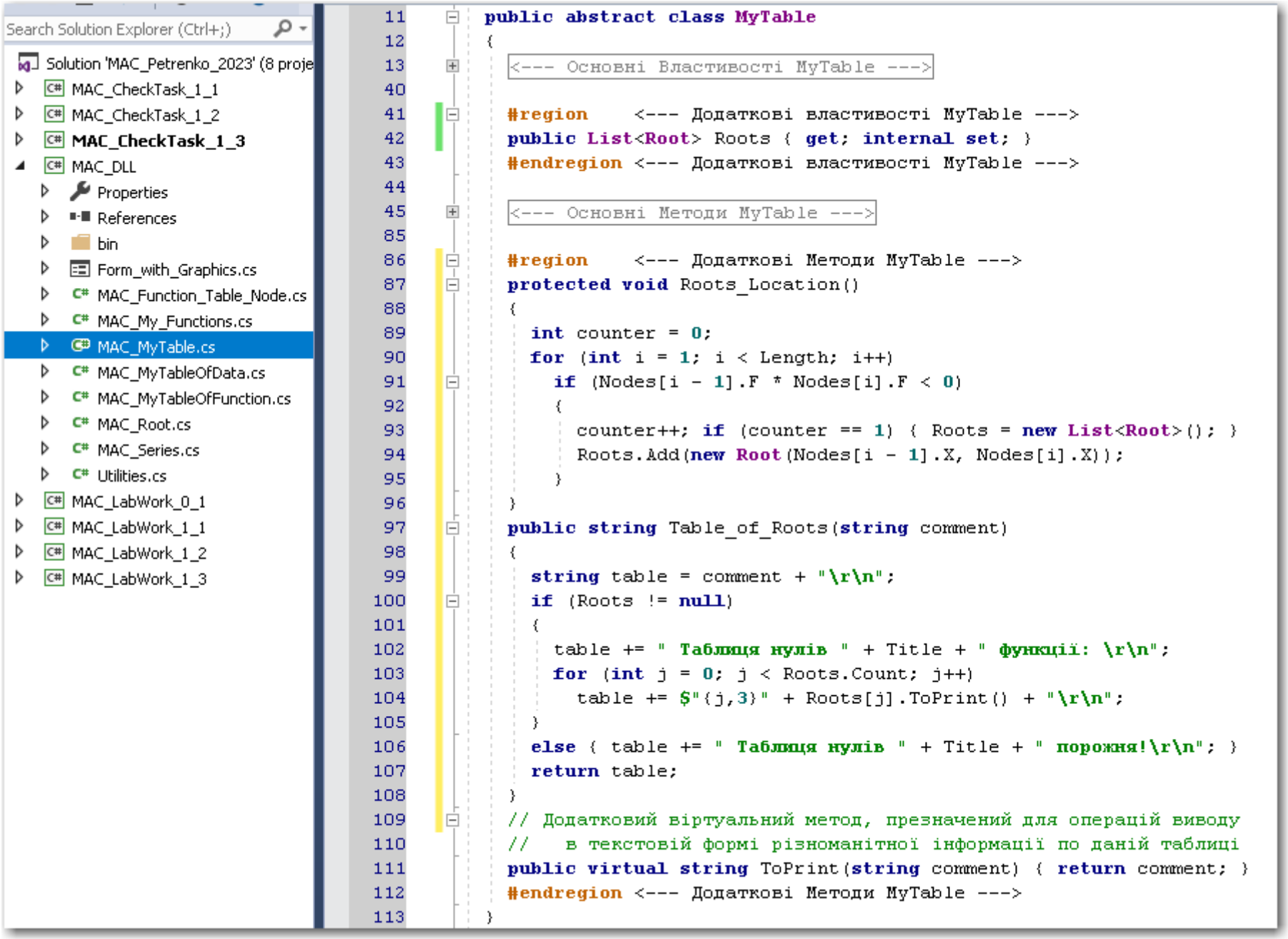
Клас `Root`.

Нижче пропонується код нового класу **`Root`**, який розмістимо в розділі загальних класів в файлі **`MAC_Root.cs`**:



Йдемо далі. Додамо тепер до сукупності членів абстрактного класу **`MyTable`** новий елемент (властивість) – **`List<Root>`** **`Roots`** – список об'єктів типа **`Root`**. Алгоритм ініціалізації цього списку і заповнення його відповідними екземплярами класу **`Root`** реалізуємо в виді закритого метода **`Roots_Location()`** абстрактного класу **`MyTable`**.

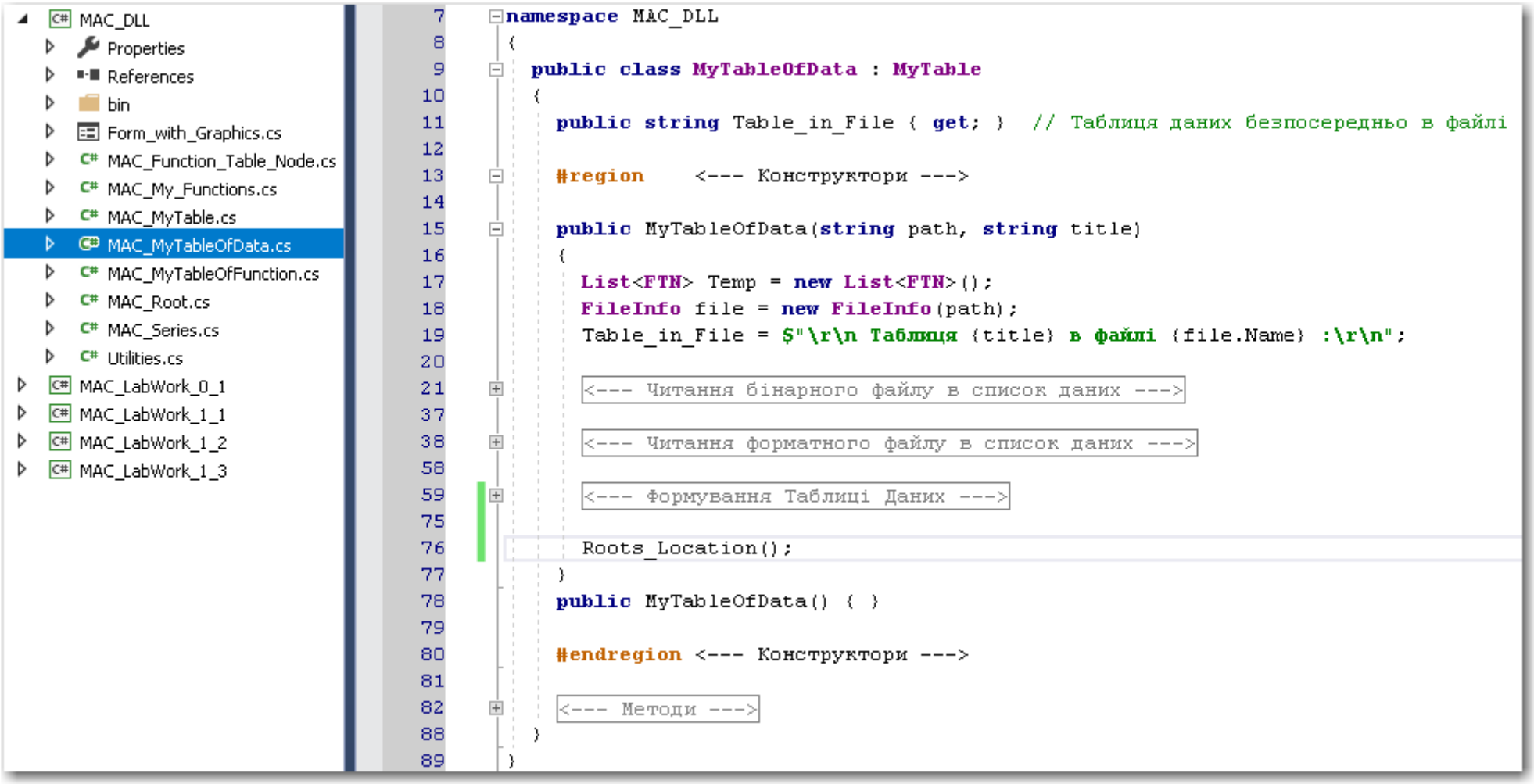
Одночасно з цим створимо допоміжний метод **`Table_of_Roots()`**, який формуватиме окрему таблицю шуканих інтервалів у текстовій формі. Обидва цих методи запишемо в окремому регіоні «Додаткові методи **`MyTable`**»:

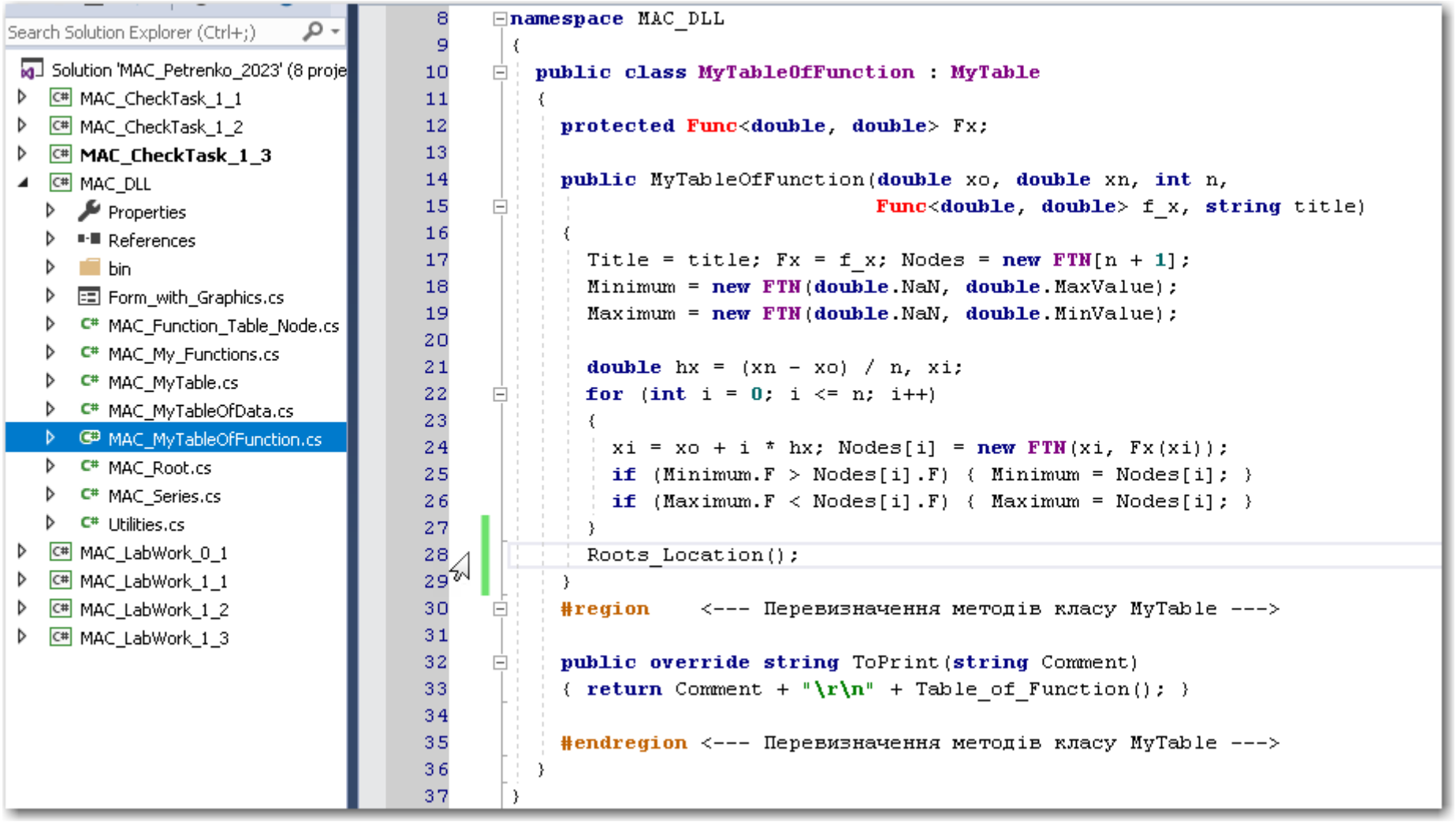

Search Solution Explorer (Ctrl+;)
Solution 'MAC_Petrenko_2023' (8 proje
MAC_CheckTask_1_1
MAC_CheckTask_1_2
MAC_CheckTask_1_3
MAC_DLL
Properties
References
bin
Form_with_Graphics.cs
MAC_Function_Table_Node.cs
MAC_My_Functions.cs
MAC_MyTable.cs
MAC_MyTableOfData.cs
MAC_MyTableOfFunction.cs
MAC_Root.cs
MAC_Series.cs
Utilities.cs
MAC_LabWork_0_1
MAC_LabWork_1_1
MAC_LabWork_1_2
MAC_LabWork_1_3
public abstract class MyTable
{
<--- Основні Властивості MyTable --->
#region <--- Додаткові властивості MyTable --->
public List<Root> Roots { get; internal set; }
#endregion <--- Додаткові властивості MyTable --->
<--- Основні Методи MyTable --->
#region <--- Додаткові Методи MyTable --->
protected void Roots_Location()
{
int counter = 0;
for (int i = 1; i < Length; i++)
if (Nodes[i - 1].F * Nodes[i].F < 0)
{
counter++; if (counter == 1) { Roots = new List<Root>(); }
Roots.Add(new Root(Nodes[i - 1].X, Nodes[i].X));
}
}
public string Table_of_Roots(string comment)
{
string table = comment + "\r\n";
if (Roots != null)
{
table += " Таблиця нулів " + Title + " функції: \r\n";
for (int j = 0; j < Roots.Count; j++)
table += $"{j,3}" + Roots[j].ToPrint() + "\r\n";
}
else { table += " Таблиця нулів " + Title + " порожня!\r\n"; }
return table;
}
// Додатковий віртуальний метод, презначений для операцій виводу
// в текстовій формі різноманітної інформації по даній таблиці
public virtual string ToPrint(string comment) { return comment; }
#endregion <--- Додаткові Методи MyTable --->
}

Тепер розширимо функціонал методу **`MyTable.Table_of_Function()`**:

```
45      #region    <--- Основні Методи MyTable --->
46
47      // Індексатор MyTable
48      public FTN this[int index] { get { return (0 <= index) && (index < Length) ? Nodes[index] : null; } }
49
50      // Повертає значення x для рядка таблиці з номером index
51      public double X(int index) => (0 <= index) && (index < Length) ? this[index].X : double.NaN;
52
53      // Повертає значення f для рядка таблиці з номером index
54      public double F(int index) => (0 <= index) && (index < Length) ? this[index].F : double.NaN;
55
56      // Повертає два масиви із значеннями аргументу x і функції f
57      public void ToArrays(out double[] x, out double[] f)...
62
63      // Створює таблицю функції в текстовому форматі
64      public virtual string Table_of_Function()
65      {
66        string txt = "\r\n  Таблиця функції " + Title + " : \r\n";
67        for (int i = 0; i < Length; i++) { txt += $"{i,4}" + Nodes[i].ToPrint() + "\r\n"; }
68        txt += $"\r\n  x = [{Nodes[0].X,17:F12} :{Nodes[Length - 1].X,17:F12} ]";
69        txt += $"   x_Reg = {Region_x,16:F12}\r\n";
70        txt += $"\r\n Min  ({Minimum.X,18:F12},{Minimum.F,18:F12} )";
71        txt += $"\r\n Max  ({Maximum.X,18:F12},{Maximum.F,18:F12} )";
72        txt += $"  f_Reg = {Region_f,16:F12}\r\n";
73        return txt + Table_of_Roots("");  //   return txt;
74      }
75
76      // Віртуальний метод, призначений для збереження таблиці у текстовому файлі path із коментарем comment
77      public virtual void To_txt_File(string path, string comment)...
84
85      #endregion <--- Основні Методи MyTable --->
86
87      <--- Додаткові Методи MyTable --->
114   }
```

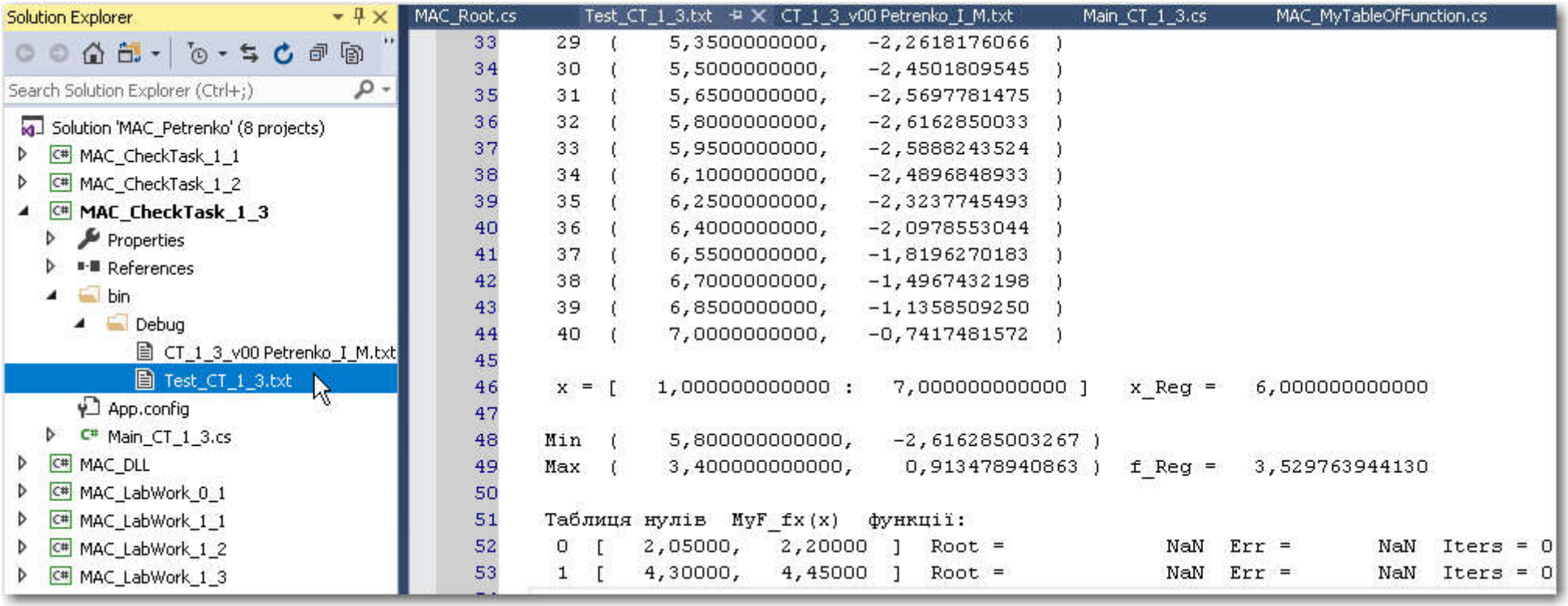
Потім додамо виклик методу **Roots_Location()** в код конструкторів класів-нащадків **MyTableOfData** та **MyTableOfFunction**:





Перекомпілюємо проект **`MAC_DLL`** та впевнимося у відсутності помилок.

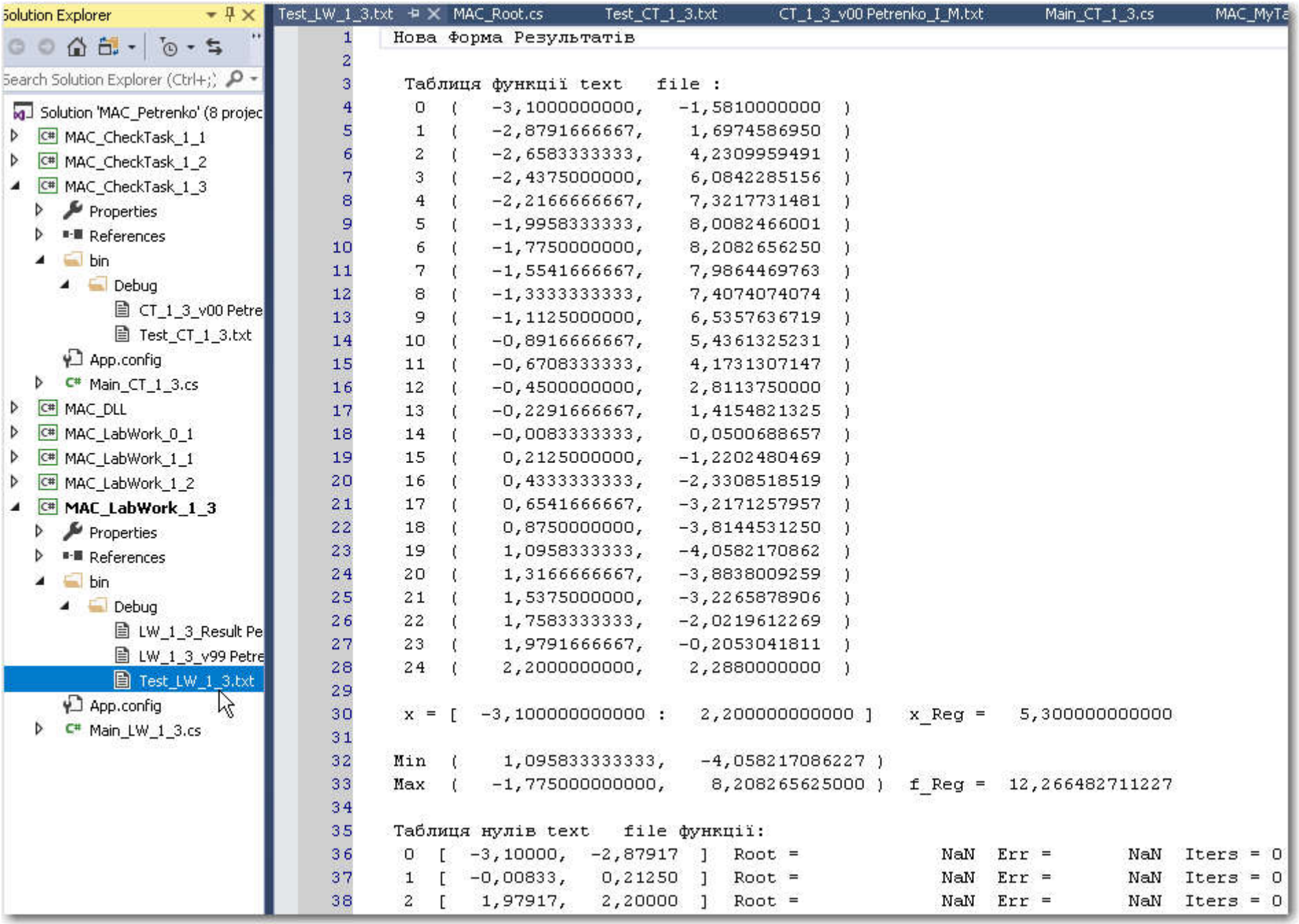
Запустимо на виконання додаток **MAC_CheckTask_1_3** (із значенням кількості вузлів таблиці **40**) і отримаємо новий результат – таблицю функції із виділеними інтервалами, на кінцях яких функція змінює знак (дивись нижню частину текстового файлу, що наводиться далі):



Тепер повернемося до проекту **MAC_LabWork_1_3** і внесемо в код відповідні зміни, після чого здійснимо перевірку коректності роботи нових методів:

```
11  namespace MAC_LabWork_1_3
12  {
13    class Main_LW_1_3
14    {
15      static void Main(string[] args)
16      {
17        using (StreamWriter SW = UTL.ResultWriter("LW_1_3_Result"))
18        {
19          MyToD T1 = new MyToD("MAC_LW_1_3_v00.bin", "binary file");
20          string txt = $"\r\n Прочитано рядків: {T1.Length,0}" + T1.ToPrint("   Обробка файлу *.bin");
21          SW.WriteLine(txt);
22
23          MyToD T2 = new MyToD("MAC_LW_1_3_v00.txt", "text   file");
24          txt = $"\r\n\r\n\r\n\r\n  Прочитано рядків: {T2.Length,0}" + T2.ToPrint("   Обробка файлу *.txt");
25          SW.WriteLine(txt);
26
27          // FwG.SingleGraphic(T2, 300, 500);
28          T2.To_txt_File("Test_LW_1_3.txt", " Нова Форма Результатів");
29        }
30      }
31    }
32  }
```

Не забудьте включити файл результатів **Test_LW_1_3.txt** до складу проекту **MAC_LabWork_1_3** та відкрити його для перегляду в середовищі **MS VS**:


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MAC_Petrenko' (8 projec
MAC_CheckTask_1_1
MAC_CheckTask_1_2
MAC_CheckTask_1_3
Properties
References
bin
Debug
CT_1_3_v00 Petre
Test_CT_1_3.txt
App.config
Main_CT_1_3.cs
MAC_DLL
MAC_LabWork_0_1
MAC_LabWork_1_1
MAC_LabWork_1_2
MAC_LabWork_1_3
Properties
References
bin
Debug
LW_1_3_Result Pe
LW_1_3_v99 Petre
Test_LW_1_3.txt
App.config
Main_LW_1_3.cs
Test_LW_1_3.txt
MAC_Root.cs
Test_CT_1_3.txt
CT_1_3_v00 Petrenko_I_M.txt
Main_CT_1_3.cs
MAC_MyTa
1 Нова Форма Результатів
3 Таблиця функції text file :
4 0 (-3,1000000000, -1,5810000000)
5 1 (-2,8791666667, 1,6974586950)
6 2 (-2,6583333333, 4,2309959491)
7 3 (-2,4375000000, 6,0842285156)
8 4 (-2,2166666667, 7,3217731481)
9 5 (-1,9958333333, 8,0082466001)
10 6 (-1,7750000000, 8,2082656250)
11 7 (-1,5541666667, 7,9864469763)
12 8 (-1,3333333333, 7,4074074074)
13 9 (-1,1125000000, 6,5357636719)
14 10 (-0,8916666667, 5,4361325231)
15 11 (-0,6708333333, 4,1731307147)
16 12 (-0,4500000000, 2,8113750000)
17 13 (-0,2291666667, 1,4154821325)
18 14 (-0,0083333333, 0,0500688657)
19 15 (0,2125000000, -1,2202480469)
20 16 (0,4333333333, -2,3308518519)
21 17 (0,6541666667, -3,2171257957)
22 18 (0,8750000000, -3,8144531250)
23 19 (1,0958333333, -4,0582170862)
24 20 (1,3166666667, -3,8838009259)
25 21 (1,5375000000, -3,2265878906)
26 22 (1,7583333333, -2,0219612269)
27 23 (1,9791666667, -0,2053041811)
28 24 (2,2000000000, 2,2880000000)
30 x = [-3,100000000000 : 2,200000000000] x_Reg = 5,300000000000
32 Min (1,095833333333, -4,058217086227)
33 Max (-1,775000000000, 8,208265625000) f_Reg = 12,266482711227
35 Таблиця нулів text file функції:
36 0 [-3,10000, -2,87917] Root = NaN Err = NaN Iters = 0
37 1 [-0,00833, 0,21250] Root = NaN Err = NaN Iters = 0
38 2 [1,97917, 2,20000] Root = NaN Err = NaN Iters = 0

Додаткові зауваження.

Основний додаток **MAC_CheckTask_1_3** повинен нам продемонструвати коректність роботи нового конструктора **MyTableOfFunction()**.

Тестова функція (с.1.3.1) вже була раніше нами розроблена й перевірена під час виконання самостійного завдання 1.2, і знаходиться в класі **MAC_My_Functions** бібліотеки **MAC_DLL**.

Основною проблемою є несумісність цієї функції (з чотирма дійсними змінними) з визначенням узагальненого делегату **Func<double, double>** (для функції від однієї змінної), який був використаний в конструкторі класу **MyTableOfFunction()**.

Розв'язок цієї проблеми може бути наступним.

В класі **Main_CT_1_3** основної програми ми об'явимо статичні члени, які будуть грати роль параметрів ε, a і b для функції (с.1.3.1).

Додамо до цього ж класу **Main_CT_1_3** «допоміжний метод» **MyF_fx(x)**, який «перетворить» виклик функції від чотирьох змінних на виклик функції від однієї змінної.

Саме цей допоміжний метод ми й будемо передавати в конструктор **MyTableOfFunction()**, оскільки його сигнатура тепер вже цілком відповідатиме узагальненому делегату **Func<double, double>**.

Інші параметри обчислень задамо таким чином, аби в таблиці, що обчислюється, опинилося два контрольних вузла для функції з самостійного завдання 1.2.

В самостійному завданні 1.2 обчислення функції здійснювалися для значень $x_1 = 3.9$ і $x_2 = 5.7$ її аргументу, при значеннях $\varepsilon = 10^{-9}$, $a = 3.0$ і $b = 1.8$ її параметрів.

Тому в теперішньому прорахунку задамо $x_0 = 1.0$, $x_n = 7.0$ і $n = 60$, що забезпечить нам обчислення значень функції (1) в необхідних точках $x_1 = 3.9$ і $x_2 = 5.7$.

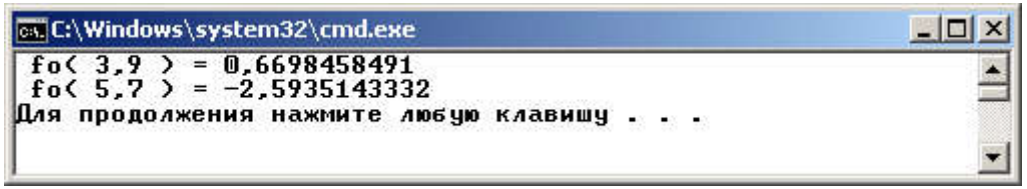
Отже, в функції **`Main()`** при ініціалізації екземпляра **`MyTableOfFunction`** в його конструктор передаються відповідні числові значення:

```
12  namespace MAC_CheckTask_1_3
13  {
14    class Main_CT_1_3
15    {
16      public static double par_a, par_b, par_e, ksi1, ksi2;
17
18      static void Main(string[] args)
19      {
20        par_a = 3.0; par_b = 1.8; ksi1 = 3.9; ksi2 = 5.7; par_e = 1.0E-9;
21
22        MyToF TF = new MyToF(1.0, 7.0, 60, MyF_fx, " MyF_fx(x) ");
23
24        using (StreamWriter SW = UTL.ResultWriter("CT_1_3_v00"))...
30
31        TF.To_txt_File("Test_CT_1_3.txt", " Нова Форма Результатів");
32        //FwG.SingleGraphic(TF, 300, 500);
33      }
34      public static double MyF_fx(double x)
35      {
36        return MyF.f0(x, par_a, par_b, par_e);
37      }
38    }
39  }
```

Перевіримо новий «зміст» текстового файлу з обробленою таблицею функції (1):

```
32      28   (     3,8000000000,     0,7706493027   )
33      29   (     3,9000000000,     0,6698458491   )
34      30   (     4,0000000000,     0,5428139298   )
35      31   (     4,1000000000,     0,3907171922   )
36      32   (     4,2000000000,     0,2154392237   )
37      33   (     4,3000000000,     0,0195489511   )
38      34   (     4,4000000000,    -0,1937565842   )
39      35   (     4,5000000000,    -0,4207303987   )
40      36   (     4,6000000000,    -0,6571722337   )
41      37   (     4,7000000000,    -0,8985405513   )
42      38   (     4,8000000000,    -1,1400760842   )
43      39   (     4,9000000000,    -1,3769326959   )
44      40   (     5,0000000000,    -1,6043109859   )
45      41   (     5,1000000000,    -1,8175899029   )
46      42   (     5,2000000000,    -2,0124516605   )
47      43   (     5,3000000000,    -2,1849954185   )
48      44   (     5,4000000000,    -2,3318355648   )
49      45   (     5,5000000000,    -2,4501809545   )
50      46   (     5,6000000000,    -2,5378921308   )
51      47   (     5,7000000000,    -2,5935143332   )
52      48   (     5,8000000000,    -2,6162850033   )
53      49   (     5,9000000000,    -2,6061154285   )
54      50   (     6,0000000000,    -2,5635471705   )
```

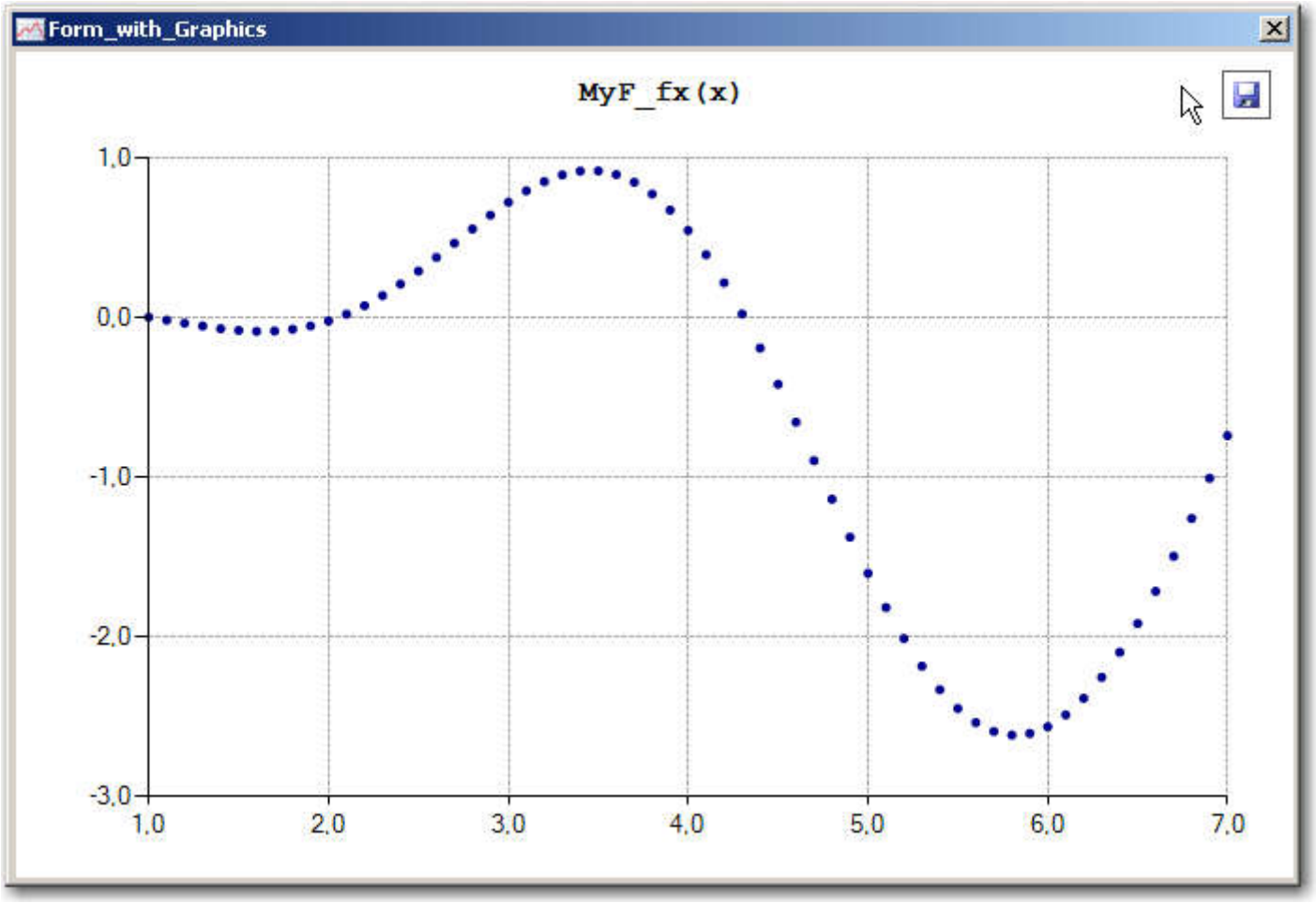
Ми можемо зіставити ці результати з результатами самостійного завдання 1.2:



Вочевидь, що вони співпадають.

Крім цього, нами виявлені ті інтервали таблиці функції, на яких вона змінює знак, а, отже, її графік перетинає вісь абсцис (нулі функції).

```
Таблиця нулів  MyF_fx(x)  функції:
 0  [   2,00000,    2,10000  ]  Root
 1  [   4,30000,    4,40000  ]  Root
```



Загальну частину завдання можна вважати виконаною.

Індивідуальне самостійне завдання 1.3

Тема: **«Побудування таблиці функції із заданою абсолютною похибкою»**

Завдання:

Розробити алгоритм і налагодити консольний **Windows**–додаток, призначений для побудування таблиці заданої функції $f(x)$ (об'єкт класу **MyTableOfFunction**) на інтервалі $[x_0, x_n]$ при заданих значеннях параметрів a, b, n та з абсолютною похибкою $\varepsilon \sim 10^{-9}$.

За цією таблицею визначити:

1) кількість інтервалів m, на яких функція $f(x)$ змінює знак;

2) найбільше $f_{\max}$ та найменше $f_{\min}$ значення функції $f(x)$ на інтервалі $[x_0, x_n]$;

3) табличні значення $f_1 = f(\xi_1)$ і $f_2 = f(\xi_2)$ для двох контрольних значень аргументу;

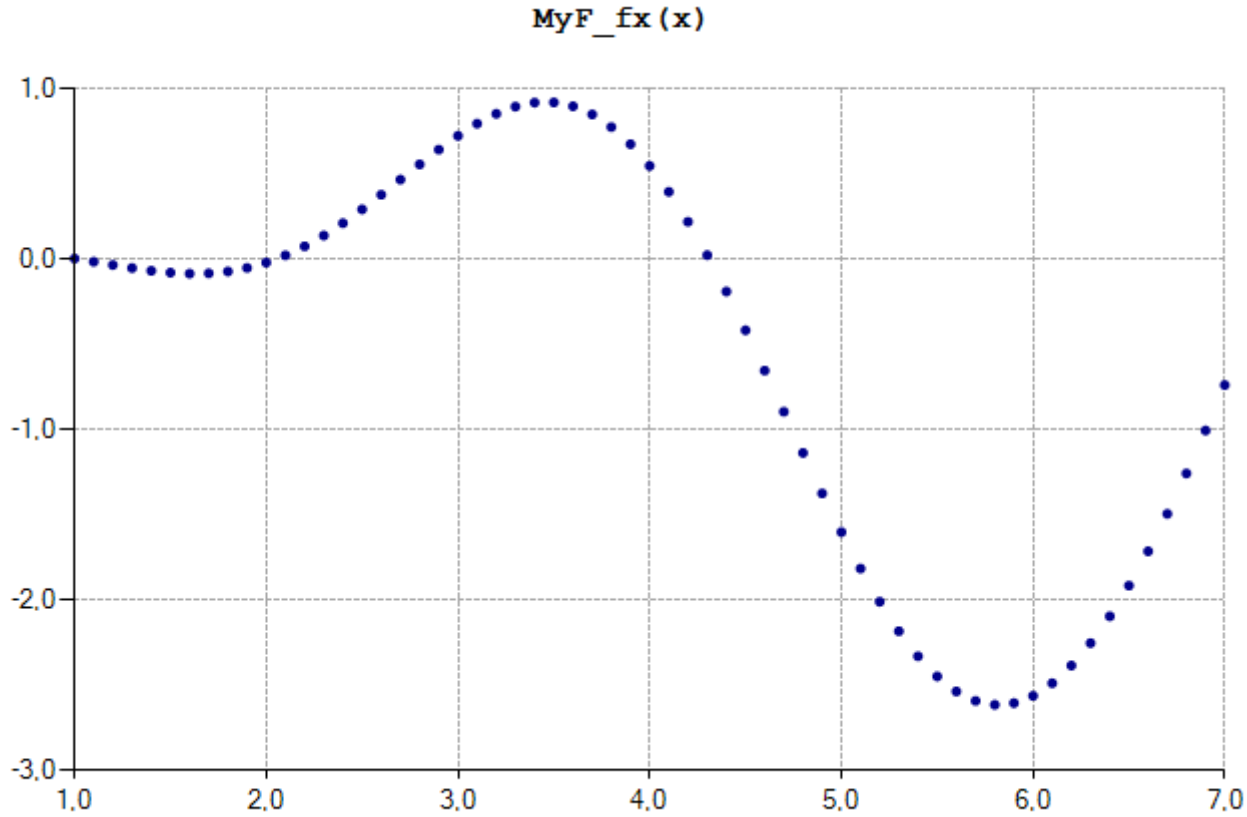
4) побудувати графік функції.

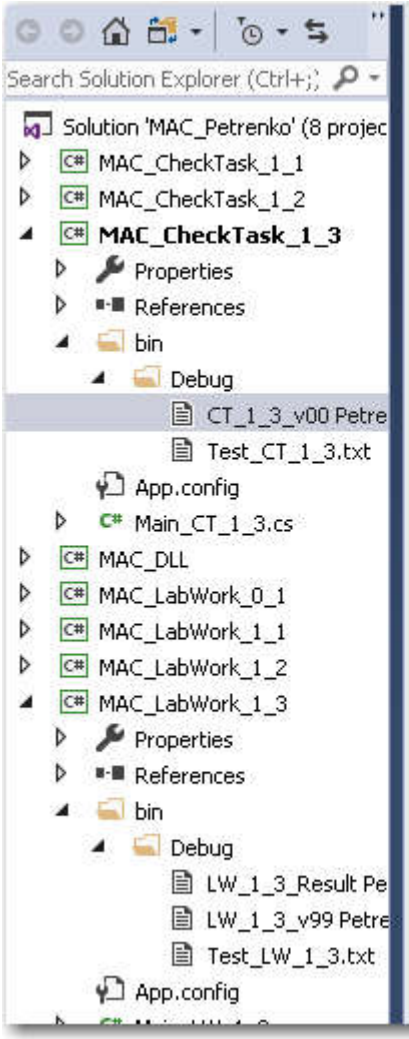
Рекомендується скористатися вже існуючим проектом **MAC_CheckTask_1_3**.

Результати обчислень записуються із **дев'ятьма** цифрами після десятинної точки (крім значення m).

Файл результатів (в текстовому форматі) із зазначеними числовими даними надсилається викладачу одночасно із кодом основного консольного додатка.

Крім цього, до текстового файлу результатів додається зображення графіка функції.






```
 98      43
 99      44
100      45
101      46
102      47
103      48
104      49
105      50
106      51
107      52
108      53
109      54
110      55
111      56
112      57
113      58
114      59
115      60   (    7,0000000000,    -0,7417481572   )
116
117      x = [    1,000000000000 :    7,000000000000 ]    x_Reg =    6,000000000000
118
119     Min   (    5,800000000000,    -2,616285003267 )
120     Max   (    3,500000000000,     0,914922357255 )   f_Reg =    3,531207360522
121
122     Таблиця нулів  MyF_fx(x)  функції:
123      0   [    2,00000,    2,10000   ]   Root =              NaN   Err =       NaN   Iters = 0
124      1   [    4,30000,    4,40000   ]   Root =              NaN   Err =       NaN   Iters = 0
125
126     f1 ( 3,90) =  0,669845849
127     f2 ( 5,70) = -2,593514333
```

Таблиця функцій за варіантами

функція $f_V(x)$	V
$f_1(x) = \sqrt{\frac{x}{2}} \cdot \sum_{k=0}^{\infty} \left[\frac{(-x^2/4)^k}{(k)! \cdot (k+1)!} \cdot \sin\left(\sqrt{\frac{a}{k+1}} \cdot x + b \right) \right]$	1
$f_2(x) = \sum_{k=0}^{\infty} \left[\frac{(-1)^k x^{2k+1}}{(2k+b) \cdot (2k+1)!} \cdot \cos\left(\frac{\pi}{2} \frac{ax^2}{kx+1} \right) \right]$	2
$f_3(x) = \sum_{k=1}^{\infty} \left[\frac{(-1)^k x^{2k}}{2k \cdot (2k)!} \cdot \left(\cos\left(\frac{a\sqrt{x+\pi}}{k} \right) + \sin\left(\frac{bx^{3/2}+\pi}{k} \right) \right) \right]$	3
$f_4(x) = \sum_{k=0}^{\infty} \left[\frac{(-1)^k x^{2k}}{(k)! \cdot (2k+1)} \cdot \left(\frac{\cos(ax)}{kx^2 + \sqrt{\pi}} - \frac{1}{\sqrt{2}} \cdot \frac{\sin(bx)}{kx+b} \right) \right]$	4
$f_5(x) = \sin(ax-b) \cdot \sum_{k=0}^{\infty} \left[\frac{(-1)^k \cdot x^{4k+1}}{(2k)! \cdot (4k+3)} \cdot \left(1 + \mathrm{th}(kx^2 + b) \right) \right]$	5

функція $f_V(x)$	V
$f_6(x) = \sum_{k=0}^{\infty}\left[\frac{(-1)^k \cdot (2x)^{k+1}}{(ak+b)\cdot (k!)^{1.5}} \cdot \cos\left(\pi x + \log_{10}(kx+b)\right)\right]$	6
$f_7(x) = \left(\cos(ax) + \sin(bx)\right) \cdot \sum_{k=0}^{\infty}\left[\frac{(-1)^k \cdot (x/2)^{2k}}{(bk+a)\cdot (2k+1)!}\right]$	7
$f_8(x) = \left(\sin\left(ax^2\right) + \sin(bx)\right) \cdot \sum_{k=1}^{\infty}\left[\frac{(-1)^k \cdot (x/3)^{2k}}{\sqrt{ak+b}\cdot (2k-1)!}\right]$	8
$f_9(x) = \ln(x+a)\cdot\cos(bx)\cdot \sum_{k=1}^{\infty}\left[\frac{(-1)^{k+1} \cdot (x/b)^{k}}{(a+b\sqrt{k})\cdot (2k)!}\right]$	9
$f_{10}(x) = \sum_{k=1}^{\infty}\left[\frac{(-1)^{k-1} x^{k-1}}{(ak-b)\cdot (k+1)!} \cdot \sin\left(ax + \log_{10}(k+x)\right)\right]$	10

Таблиця даних за варіантами

V.n	a	b	x_0	x_n	ξ_1	ξ_2	n
01.1	+1.90	-2.20	+0.40	+5.10	+2.30	+3.70	235
01.2	+1.70	-2.40	+0.60	+5.50	+1.80	+4.30	245
01.3	+1.60	-2.10	+0.70	+5.20	+2.20	+4.10	225
02.1	+0.70	+1.30	+1.40	+4.70	+2.30	+3.50	165
02.2	+0.60	+1.70	+1.60	+4.20	+2.80	+3.70	155
02.3	+0.90	+1.60	+1.70	+4.50	+2.60	+3.30	145
03.1	+0.50	-0.90	+1.10	+5.70	+2.90	+4.60	230
03.2	+0.70	-1.30	+0.80	+6.40	+2.20	+4.90	280
03.3	+0.60	-1.10	+1.30	+6.80	+2.40	+4.70	260
04.1	+2.30	+3.70	+0.40	+6.20	+2.10	+5.20	295
04.2	+2.10	+3.40	+0.30	+6.70	+1.80	+4.90	320
04.3	+2.20	+3.60	+0.70	+6.50	+1.90	+5.30	305
05.1	+5.80	-1.70	+0.50	+4.70	+2.30	+3.80	210
05.2	+5.60	-1.30	+0.80	+4.90	+2.20	+4.10	195
05.3	+5.70	-1.40	+0.60	+5.10	+1.90	+4.30	205

V.n	a	b	x_0	x_n	ξ_1	ξ_2	n
06.1	+1.80	+2.30	+1.60	+6.20	+2.50	+4.10	230
06.2	+2.10	+1.90	+1.30	+7.70	+2.60	+5.40	320
06.3	+1.90	+2.10	+1.40	+7.30	+2.40	+4.80	270
07.1	+2.50	+2.90	+1.10	+7.00	+3.30	+5.20	295
07.2	+2.40	+3.00	+1.20	+7.40	+2.60	+5.90	310
07.3	+2.20	+3.70	+0.90	+7.10	+2.80	+5.60	330
08.1	+0.60	+1.10	+0.70	+6.50	+2.70	+4.80	290
08.2	+0.40	+1.30	+0.80	+6.70	+3.20	+5.50	295
08.3	+0.70	+1.40	+0.90	+6.30	+3.10	+5.20	305
09.1	+1.30	+2.40	+1.10	+7.50	+3.60	+5.80	320
09.2	+1.50	+2.70	+0.80	+7.60	+3.30	+6.10	340
09.3	+1.40	+2.50	+0.90	+7.30	+3.40	+5.90	360
10.1	+1.30	-2.40	+1.10	+8.50	+3.30	+5.20	370
10.2	+1.70	-2.60	+0.80	+7.90	+2.40	+5.10	355
10.3	+1.40	-2.50	+0.90	+6.80	+2.70	+5.50	385

Таблиця тестових даних за варіантами

V.n	a	b	x_0	x_n	ξ_1	ξ_2	n
01.0	+1.50	-2.00	+1.00	+5.00	+2.50	+3.50	200
02.0	+0.50	+1.50	+1.30	+4.40	+2.50	+3.20	155
03.0	+1.00	-0.50	+1.50	+6.50	+2.50	+5.00	250
04.0	+2.00	+4.00	+0.50	+6.00	+2.00	+5.00	275
05.0	+5.50	-1.50	+1.00	+5.00	+2.00	+3.50	200
06.0	+1.50	+2.50	+1.50	+7.50	+3.00	+5.00	300
07.0	+2.30	+3.10	+0.70	+7.50	+3.00	+4.50	340
08.0	+0.50	+1.50	+1.50	+6.00	+2.00	+4.00	225
09.0	+1.00	+3.00	+0.50	+7.00	+3.00	+5.50	325
10.0	+1.00	-2.00	+1.50	+7.00	+3.00	+4.50	275

Таблиця тестових результатів за варіантами

V.n	$f_V(\xi_1)$	$f_V(\xi_2)$	$f_{\min}$	$f_{\max}$	m
01.0	+0.7925680161	-0.3069116197	-1.9838974539	+0.7931198736	2
02.0	+0.2890879467	+0.3050709488	-1.5018228200	+5.2542740122	4
03.0	+0.0098364490	+5.4693535330	-6.0158042675	+7.4362205612	2
04.0	-0.3767224509	-0.4664821840	-0.5003017049	+0.4782533950	4
05.0	+0.1206466332	-2.2252029495	-2.7806347196	+2.7185306998	7
06.0	+0.0580223995	+0.1243954257	-0.8146874594	+0.6603617119	6
07.0	+0.3479710586	+0.1148569555	-0.8202073868	+0.7449582949	7
08.0	-0.3086823606	-0.6743965722	-2.6311891698	+1.2963135994	5
09.0	-0.1481269063	-0.2679400356	-0.3584013519	+0.4555560097	7
10.0	-0.0361313697	-0.0644695471	-0.0779769543	+0.1148082413	2

Лабораторна робота № 1.4 «Обчислення коренів нелінійного рівняння методом дихотомії»

Однією з основних проблем обчислювальної математики є задача визначення простих коренів для трансцендентних рівнянь виду $F(x) = 0$.

Цю ж задачу можна сформулювати таким чином: «*проблема пошуку некратних нулів для деякої функції* $F = F(x)$».

В класичній теорії наближених обчислень відома велика кількість різноманітних методів розв'язування даної проблеми, які враховують специфіку та властивості функції $F(x)$. Самим надійним і, мабуть, самим простим з точки зору організації ітераційних обчислень на комп'ютері, є ***метод дихотомії*** – метод ділення відрізку навпіл.

Мета лабораторної роботи

Побудувати алгоритм та налагодити програму визначення коренів заданого рівняння $F(x) = 0$ на інтервалі $x \in [x_0; x_n]$ з абсолютною похибкою $\varepsilon \approx 10^{-12}$. Розв'язування даної задачі будемо здійснювати в два послідовних етапи.

Перший етап:

10. розробка програмної реалізації для метода дихотомії;
10. оформлення цієї реалізації в виді бібліотечного метода;
10. тестування метода на модельному трансцендентному рівнянні.

Другий етап:

10. використання бібліотечного метода дихотомії для обчислення всіх нулів для таблиці заданої функції $F(x)$;
10. розширення функціонала для об'єкта класу `MyTableOfFunction` шляхом додавання до його складу членів і методів, які оперують із нулями функції $F(x)$;
10. тестування другого етапу на таблиці модельної функції.

Заключний етап – виконання підсумкових обчислень за індивідуальним варіантом даної лабораторної роботи.

Вимоги до програмних компонентів

Програмні компоненти, що розробляються, повинні бути додані до Вашого робочого простору **MAC_Petrenko**.

Програмна реалізація методу дихотомії повинна входити до складу окремого (нового) класу **MAC_Equations** бібліотеки класів **MAC_DLL**. Основну програмну одиницю даної лабораторної роботи треба генерувати в складі окремого консольного проекту **MAC_LabWork_1_4**.

Порядок виконання завдання лабораторної роботи

У відповідності до задач першого етапу додамо до складу динамічної бібліотеки **MAC_DLL** новий клас, який буде містити програмні реалізації методів уточнення коренів нелінійних рівнянь, які будуть побудовані на основі різних чисельних алгоритмів.

Цей клас будемо іменувати **MAC_Equations**.

Алгоритм методу дихотомії **Dichotomy()**, який ми першим включимо до класу **MAC_Equations**, детально розглядався на лекційному занятті і його можна повторити із використанням відповідного конспекту, який Ви маєте в електронній формі.

Обов'язковими параметрами цього методу повинні бути:

- границі інтервалу $[x_L; x_R]$, на кінцях якого функція $F(x)$ має різні знаки, тобто обов'язково повинна виконуватися нерівність $F(x_L) \cdot F(x_R) < 0$;
- делегат, який має сигнатуру функції $F(x)$ від однієї дійсної змінної x;
- ε – абсолютна похибка обчислень розв'язку (кореня заданого рівняння);
- K – кількість ітерацій, які були здійснені під час обчислень розв'язку.

Метод, який розробляється, повинен повертати до основної програми значення $x_*^{(K)}$, яке задовольняє умові $|F(x_*^{(K)})| \le \varepsilon$, де K – номер останньої ітерації.

Для запобігання можливого зациклення обчислень в алгоритмі методу дихотомії слід обмежити кількість ітерацій K двома додатковими умовами (примусовий вихід з циклу при виконанні однієї умови з двох наступних): а) $K > 70$; б) $10 \cdot \left(x_R^{(K)} - x_L^{(K)}\right) \le \varepsilon$.

В коді методу **`Dichotomy()`**, який наводиться далі, застосований спеціальний алгоритм обчислення середнього арифметичного для двох дійсних чисел із урахуванням їх знаків.

```
7   namespace MAC_DLL
8   {
9     public class MAC_Equations
10    {
11      public static double Dichotomy(double a, double b, double eps, Func<double, double> f, out int K)
12      {
13        double fa = f(a), fc, c = 0.0; K = 0;
14        while (K < 70)
15        {
16          if ((a * b) < 0) c = (a + b) * 0.5; else c = a + (b - a) * 0.5;
17          fc = f(c); K++;
18          if ((fa * fc) < 0) b = c; else a = c;
19          if ((Math.Abs(fc) < eps) || ((b - a) * 10 < eps)) break;
20        }
21        return c;
22      }
23    }
24  }
```

Отже, метод дихотомії ми додали до нового класу **MAC_Equations**, і ми можемо переходити до його тестування в рамках консольного додатку **MAC_LabWork_1_4.**

Виконаємо стандартні дії для організації взаємодії проектів **MSVS** в робочому просторі **MAC_Petrenko** і додамо посилання на бібліотеку **MAC_DLL** до переліку **References** нового проекту **MAC_LabWork_1_4**.

Для подальшої роботи нам знадобиться просте нелінійне рівняння $F(x) = 0$ з достатньо простою функцією $F(x)$ та відомим (очевидним) розв'язком (тестове рівняння).

При цьому бажано, аби корінь цього рівняння виражався достатньо простим числом, наприклад, цілим числом, або простим десятковим дробом.

Нехай нами обрано наступне тестове рівняння:

$$\cos(\pi x) = 0\,. \qquad \text{(л.1.4.1)}$$

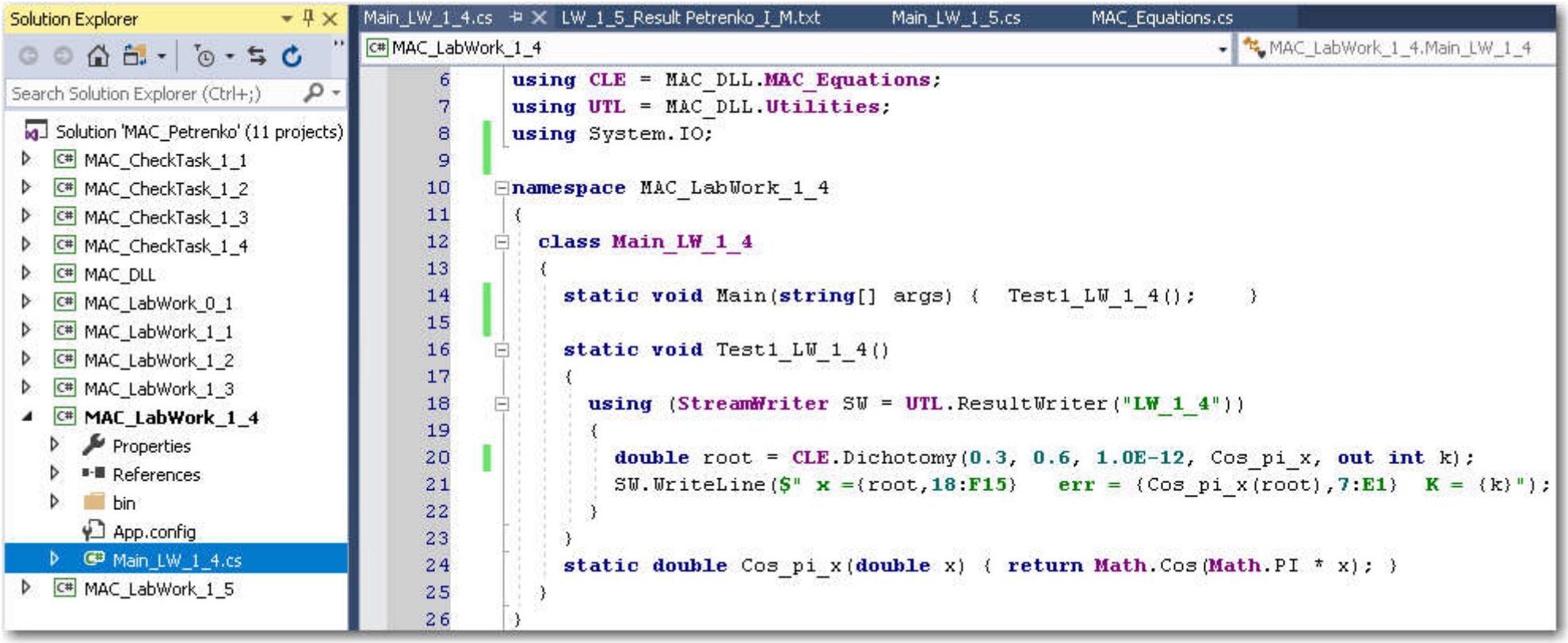
Це тригонометричне рівняння має нескінченну множину простих коренів, які є раціональними числами та описуються простою формулою

$$x_m = \frac{1}{2} + m\,, \quad \text{де } m \in Z\,. \qquad \text{(л.1.4.2)}$$

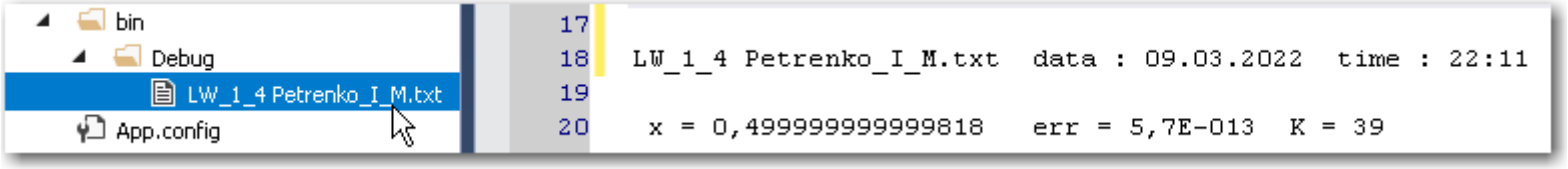
Нас буде цікавити лише перший додатний корінь $x_* = 0.5$ цього рівняння.

Цей корінь, вочевидь, належить інтервалу $[\,0.3,\,0.6\,]$, яким ми можемо скористатися в якості відрізку $[a;b]$, який містить ізольований простий корінь x_*.

Побудуємо код основного додатку **MAC_LabWork_1_4**, в якому обчислимо з заданою похибкою цей корінь із застосуванням вже існуючого бібліотечного метода **Dichotomy()**.



Результат виконання тесту:



Аналіз результату свідчить про те, що алгоритм метода дихотомії, який був щойно нами розроблений, витратив 39 кроків ітерацій на уточнення шуканого кореня x_* до значення, яке забезпечує нам абсолютну похибку 10^{-12}.

Такий результат можна вважати цілком задовільним і перший етап лабораторної роботи 1.4 успішно завершеним.

Переходимо до **другого** етапу лабораторної роботи 1.4.

Суть цього етапу нам допоможе усвідомити наступна задача.

Нехай нам задане трансцендентне рівняння (л.1.4.3), для якого необхідно: спочатку з'ясувати саму наявність інтервалів, які містять корені рівняння, на заданому відрізку $x \in [\,0.0, 15.0\,]$ зміни аргументу, а потім вже уточнити всі ці корені із абсолютною похибкою 10^{-12}:

$$F(x) = \mathrm{th}(x) - 2\cos(\sqrt{10}x) = 0\,. \qquad \text{(л.1.4.3)}$$

Для виконання цієї задачі ми повинні будемо скористатися об'єктом класу **MyTableOfFunction**, який являє собою впорядковану таблицю функції $F(x)$, яку заздалегідь обчислено у відповідності до функціонального зображення (1.4.3).

Екземпляр класу **MyTableOfFunction** (який є нащадком абстрактного класу **MyTable**) вже має в собі властивість **Roots** – список об'єктів типу **Root**, які є визначеними інтервалами, на кінцях яких задана функція $F(x)$ змінює свій знак:

```
9   public class Root
10  {
11      public double XL, XR, X, Err; public int Iters;
12
13      public Root(double x_left, double x_right)
14      {
15          XL = x_left; XR = x_right; X = double.NaN; Err = double.NaN; Iters = 0;
16      }
17      public string ToPrint()
18      {
19          return $"  [{XL,10:F5},{XR,10:F5}  ]  Root ={X,16:F12}  Err ={Err,10:E1}  Iters = {Iters}";
20      }
21  }
```

Кожен елемент списку **Roots** має поле **Roots.X**, яке призначене для «зберігання» абсциси нуля функції (кореня рівняння), що відповідає даному інтервалу **[xL,xR]**.

Поле **Roots.Err** призначене для запису похибки, яку було припущено при обчисленні цього нуля, а поле **Roots.Iters** – для кількості ітерацій, які були виконані конкретним чисельним методом для уточнення цього нуля.

Отже, нам необхідно в рамках класу **MAC_Equations** перевизначити метод дихотомії таким чином, аби він був здатним «працювати» з об'єктом класу **Root** та визначати для заданої функції $F(x)$ корінь рівняння з необхідною точністю на «власному» інтервалі **[xL,xR]**.

Існуючий і вже налагоджений алгоритм цього методу дихотомії дозволить нам без зайвих зусиль виконати цю задачу.

Далі ми наводимо сигнатуру перевантаженого методу:

```
7   namespace MAC_DLL
8   {
9     public class MAC_Equations
10    {
11      public static double Dichotomy(double a, double b, double eps, Func<double, double> f, out int K)...
23
24      public static void Dichotomy(Func<double, double> f, Root root, double eps)
25      {
26        double a = root.XL, b = root.XR, c = 0.0, fc, fa = f(a); root.Iters = 0;
27        while (root.Iters < 70)
28        {
29          if ((a * b) < 0) c = (a + b) * 0.5; else c = a + (b - a) * 0.5;
30          fc = f(c); root.Iters++;
31          if ((fa * fc) < 0) b = c; else a = c;
32          root.Err = Math.Abs(fc);
33          if ((root.Err < eps) || ((b - a) * 10 < eps)) break;
34        }
35        root.X = c; return;
36      }
37    }
38  }
```

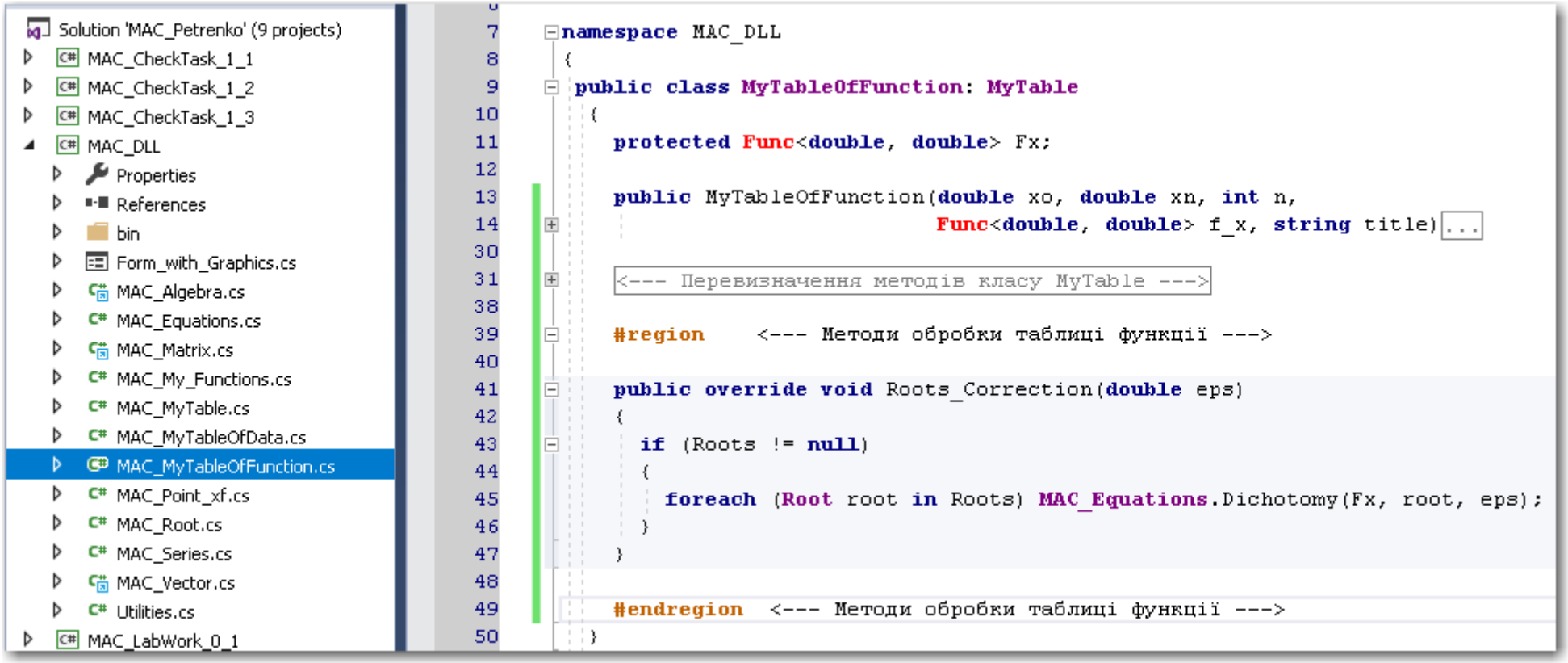
Працюємо далі.

Додаємо до регіону «Додаткові методи» абстрактного класу **MyTable** декларування нового віртуального методу **Roots_Correction()**:

```
 8   namespace MAC_DLL
 9   {
10     public abstract class MyTable
11     {
12       <--- Основні Властивості MyTable --->
40
41       <--- Додаткові Властивості MyTable --->
44
45       <--- Основні Методи MyTable --->
92
93       #region    <--- Додаткові Методи MyTable --->
94
95       public virtual void Roots_Correction(double eps) { }
96
97       protected void Roots_Location()...
107
108      public string Table_of_Roots(string comment)...
120
121      #endregion <--- Додаткові Методи MyTable --->
122    }
123  }
```

Після цього додаємо до коду класу **MyTableOfFunction** новий регіон «Методи обробки таблиці функції».

В цьому регіоні сформуємо новий метод (імплементацію віртуального методу **Roots_Correction()**), який буде здійснювати уточнення всіх нулів заданої функції із використанням перевантаженого методу дихотомії:



Повертаємося до основного додатку **MAC_LabWork_1_4** даної лабораторної роботи.

Додаємо новий метод тестування, в якому будуть виконані наступні дії:

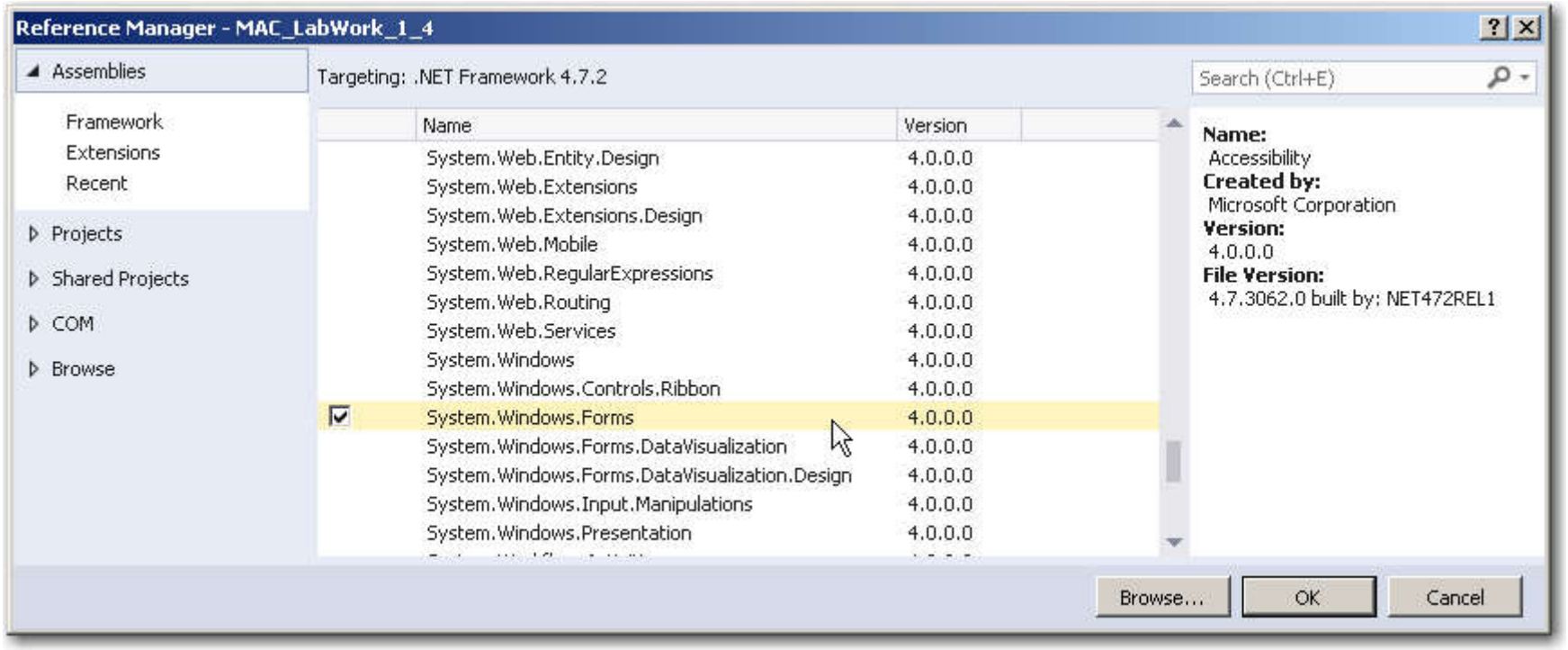
1) генерація таблиці функції (л.1.4.3) на заданому відрізку **[0.0, 15.0]**;

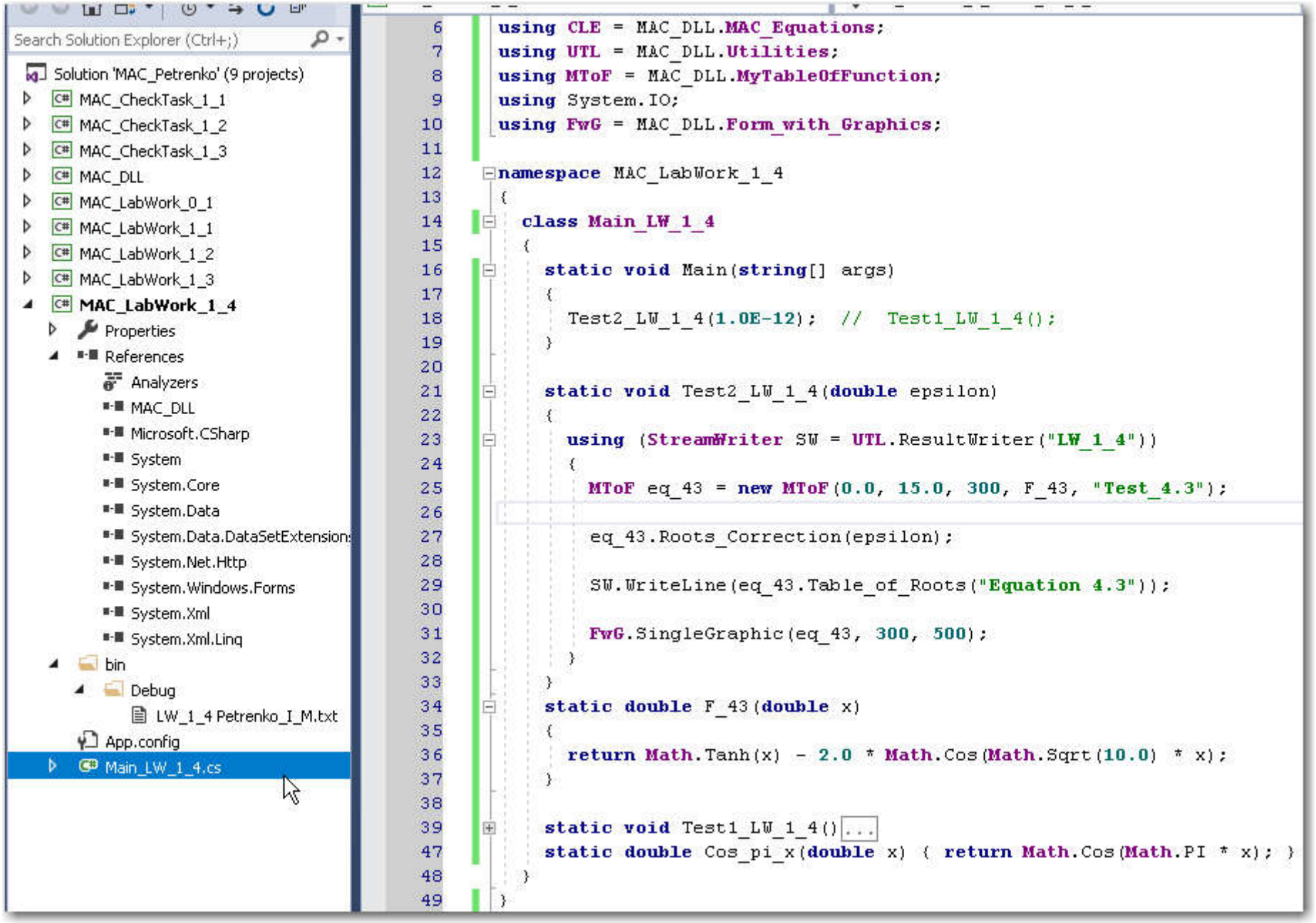
2) уточнення всіх нулів функції із застосуванням бібліотечного методу дихотомії з похибкою 10^{-12} та виведення результатів у текстовий файл для подальшого аналізу;

3) порівняльний аналіз результатів обчислень із застосуванням графіка рівняння.

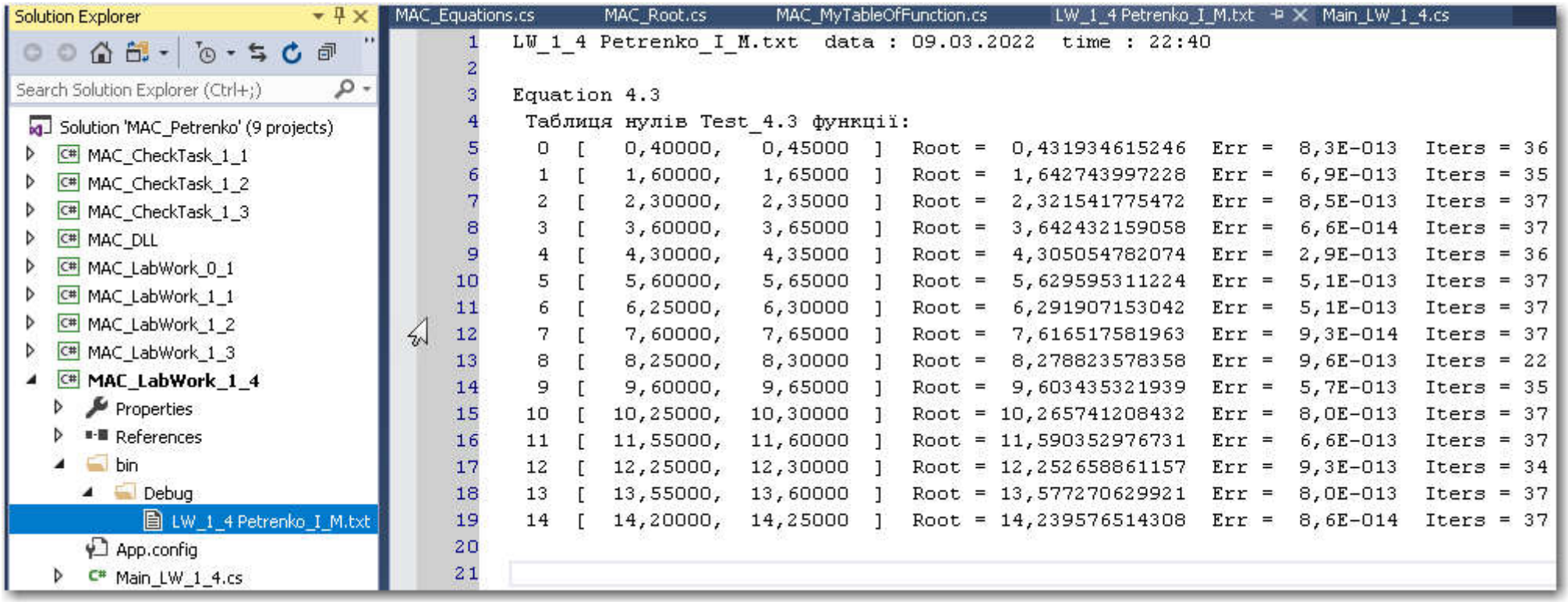
Крім цього, додамо до основного додатку статичний метод, який обчислюватиме значення функції (л.1.4.3) та який відповідатиме сигнатурі делегата **`Func<double, double>`**.

Нагадаємо, що задля перегляду графіків функцій у консольному додатку нам слід підключити до переліку посилань **`References`** даного проекту посилання на системну бібліотеку **`System.Windows.Forms`**:

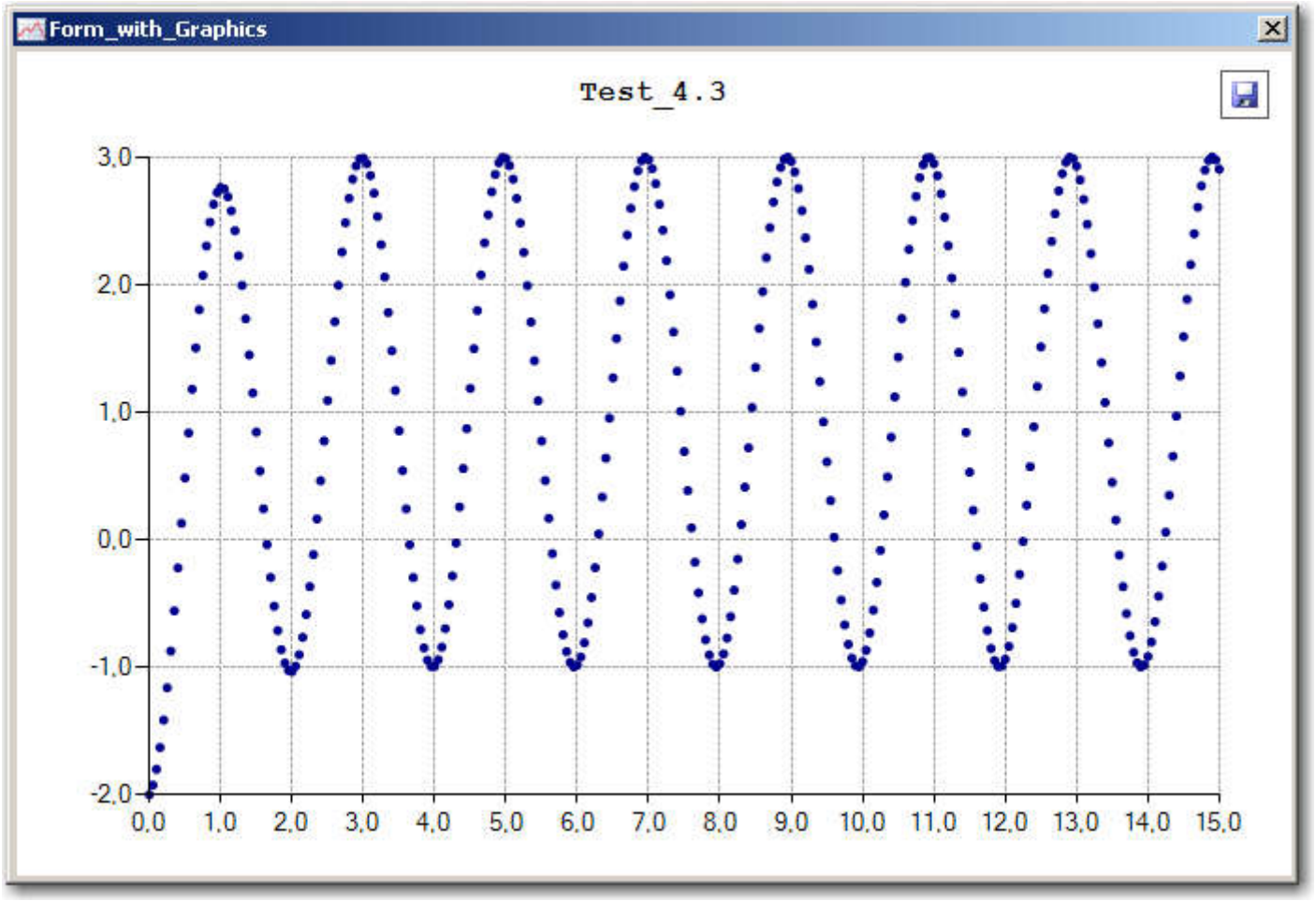



Search Solution Explorer (Ctrl+;)
Solution 'MAC_Petrenko' (9 projects)
MAC_CheckTask_1_1
MAC_CheckTask_1_2
MAC_CheckTask_1_3
MAC_DLL
MAC_LabWork_0_1
MAC_LabWork_1_1
MAC_LabWork_1_2
MAC_LabWork_1_3
MAC_LabWork_1_4
Properties
References
Analyzers
MAC_DLL
Microsoft.CSharp
System
System.Core
System.Data
System.Data.DataSetExtension
System.Net.Http
System.Windows.Forms
System.Xml
System.Xml.Linq
bin
Debug
LW_1_4 Petrenko_I_M.txt
App.config
Main_LW_1_4.cs
using CLE = MAC_DLL.MAC_Equations;
using UTL = MAC_DLL.Utilities;
using MToF = MAC_DLL.MyTableOfFunction;
using System.IO;
using FwG = MAC_DLL.Form_with_Graphics;
namespace MAC_LabWork_1_4
{
class Main_LW_1_4
{
static void Main(string[] args)
{
Test2_LW_1_4(1.0E-12); // Test1_LW_1_4();
}
static void Test2_LW_1_4(double epsilon)
{
using (StreamWriter SW = UTL.ResultWriter("LW_1_4"))
{
MToF eq_43 = new MToF(0.0, 15.0, 300, F_43, "Test_4.3");
eq_43.Roots_Correction(epsilon);
SW.WriteLine(eq_43.Table_of_Roots("Equation 4.3"));
FwG.SingleGraphic(eq_43, 300, 500);
}
}
static double F_43(double x)
{
return Math.Tanh(x) - 2.0 * Math.Cos(Math.Sqrt(10.0) * x);
}
static void Test1_LW_1_4()...
static double Cos_pi_x(double x) { return Math.Cos(Math.PI * x); }
}
}

Результати виконання додатку **MAC_LabWork_1_4** повинні бути наступними:



Графік рівняння (л.1.4.3)



Ми спостерігаємо 15 перетинів графіком рівняння осі абсцис, та обчислили рівно 15 коренів.

Тому математичний підсумок роботи, яку ми щойно виконали, можна сформулювати так:

«На відрізку $[\,0, 15\,]$ тестова функція (л.1.4.3) містить загалом п'ятнадцять нулів, які нам вдалося обчислити із використанням методу дихотомії з абсолютною похибкою 10^{-12}».

Індивідуальне завдання з лабораторної роботи 1.4 за варіантами

Тема: ***«Обчислення коренів нелінійного рівняння методом дихотомії»***

Завдання:

Побудувати алгоритм та налагодити консольний Windows–додаток, в рамках якого здійснити наступні обов'язкові дії:

1) обчислити таблицю заданої функції $F(x)$ на заданому інтервалі $[x_0; x_n]$ зміни її аргументу x, з шагом таблиці, який повинен не перевищувати 1% від загальної довжини інтервалу $[x_0; x_n]$;

2) визначити всі інтервали з таблиці функції, які містять простий корінь рівняння $F(x) = 0$;

3) за допомогою існуючого методу дихотомії уточнити корені рівняння $F(x) = 0$ з абсолютною похибкою $\varepsilon \sim 10^{-12}$ та отримати відповідну таблицю коренів.

У Windows–додатку **обов'язково** повинні бути використані програмні компоненти (класи, методи тощо), які були Вами розроблені та налагоджені під час виконання лабораторної роботи 1.4.

Файл результатів повинен містити програмний код за яким виконувалися обчислення.

Таблиця рівнянь та їх параметрів за варіантами

V	рівняння $F_V(x)=0$	x_0	x_n	a	b	n	V.v
1	$F_1(x)=\mathrm{th}(a\cdot x)-b\cdot\cos\left(x\cdot\sqrt{\lg 2+\pi^2}\right)=0$	-24.5	-21.1	-0.21	+1.90	500	01.1
		-24.8	-22.1	-0.04	-1.30	700	01.2
		-24.7	-20.9	+0.05	-1.80	900	01.3
2	$F_2(x)=\mathrm{th}(a\cdot x)+b\cdot\cos\left(x\cdot\sqrt{\lg 3+\pi^2}\right)=0$	-19.1	-16.2	-0.13	+1.50	600	02.1
		-19.6	-17.1	+0.04	+1.80	800	02.2
		-19.3	-15.8	-0.09	+1.70	700	02.3
3	$F_3(x)=\mathrm{th}(a\cdot x)-b\cdot\cos\left(x\cdot\sqrt{\lg 4+\pi^2}\right)=0$	-14.5	-11.0	+0.12	-1.70	700	03.1
		-14.8	-11.8	-0.18	+2.40	900	03.2
		-14.6	-10.1	-0.07	-2.30	800	03.3
4	$F_4(x)=\mathrm{th}(a\cdot x)+b\cdot\cos\left(x\cdot\sqrt{\lg 5+\pi^2}\right)=0$	-9.3	-6.1	+0.09	-1.40	800	04.1
		-9.6	-6.9	-0.23	+2.10	700	04.2
		-9.4	-5.2	-0.07	-1.60	900	04.3
5	$F_5(x)=\mathrm{th}(a\cdot x)-b\cdot\cos\left(x\cdot\sqrt{\lg 6+\pi^2}\right)=0$	-4.5	-1.3	+0.12	+1.60	700	05.1
		-4.1	-0.8	+0.17	+2.10	900	05.2
		-4.8	-1.1	+0.14	+1.90	800	05.3

V	рівняння $F_V(x)=0$	x_0	x_n	a	b	n	V.v
6	$F_6(x)=\operatorname{th}(a\cdot x)+b\cdot\cos\left(x\cdot\sqrt{\lg 7+\pi^2}\right)=0$	+1.3	+4.4	-0.18	+2.10	600	06.1
		+1.9	+4.9	-0.25	+1.80	800	06.2
		+1.4	+4.7	-0.21	+1.90	900	06.3
7	$F_7(x)=\operatorname{th}(a\cdot x)-b\cdot\cos\left(x\cdot\sqrt{\lg 8+\pi^2}\right)=0$	+6.1	+9.4	-0.07	-1.80	800	07.1
		+6.8	+9.1	+0.12	-2.10	900	07.2
		+5.1	+9.7	+0.08	-1.90	700	07.3
8	$F_8(x)=\operatorname{th}(a\cdot x)+b\cdot\cos\left(x\cdot\sqrt{\lg 9+\pi^2}\right)=0$	+11.1	+13.8	+0.28	-1.50	900	08.1
		+11.4	+14.3	+0.17	-2.40	700	08.2
		+10.6	+14.1	+0.24	-1.70	800	08.3
9	$F_9(x)=\operatorname{th}(a\cdot x)-b\cdot\cos\left(x\cdot\sqrt{1.0+\pi^2}\right)=0$	+15.6	+18.8	-0.34	+1.40	700	09.1
		+16.1	+19.3	-0.27	+1.90	900	09.2
		+15.6	+19.8	-0.25	+2.30	800	09.3
10	$F_{10}(x)=\operatorname{th}(a\cdot x)+b\cdot\cos\left(x\cdot\sqrt{\lg 11+\pi^2}\right)=0$	+21.5	+24.3	+0.35	+1.70	900	10.1
		+22.1	+24.9	+0.24	+2.10	700	10.2
		+21.9	+25.3	+0.31	+1.90	800	10.3

Таблиця тестових результатів за варіантами

V	x_0	x_n	a	b	n	x_*	x_{**}	k
1	-25.0	-20.0	-0.05	+2.00	1000	-23.9998845648	-20.0715917207	5
2	-20.0	-15.0	-0.05	+2.00	1000	-18.9275728621	-15.0377514909	5
3	-15.0	-10.0	-0.05	-2.00	1000	-14.9482090630	-10.2674017881	6
4	-10.0	-5.0	+0.05	-2.00	1000	-9.1143595758	-5.2752507909	5
5	-5.0	0.0	+0.05	+2.00	1000	-4.3654359191	-0.4850990238	5
6	0.0	+5.0	-0.05	+2.00	1000	+0.4762401999	+4.2865852345	5
7	+5.0	+10.0	-0.05	-2.00	1000	+5.3040084889	+9.9793405411	6
8	+10.0	+15.0	+0.05	-2.00	1000	+11.0586086675	+14.8987331692	5
9	+15.0	+20.0	+0.05	+2.00	1000	+15.6217585509	+19.4178104133	5
10	+20.0	+25.0	-0.05	+2.00	1000	+20.5684303362	+24.3836965270	5

В якості тестового результату в цій таблиці записані абсциси молодшого x_* та старшого x_{**} коренів рівняння $F(x) = 0$.

Мається на увазі, що молодший корінь x_* – це той, що має найменшу абсцису серед усіх коренів, що були обчислені для заданого інтервалу $[x_0; x_n]$, а старший корінь x_{**} – це той, що має найбільшу абсцису.

Параметр k визначає загальну кількість коренів рівняння на інтервалі $[x_0; x_n]$.

Методика виконання самостійного завдання № 1.4
«Визначення абсцис, які відповідають заданому значенню функції»

Постановка завдання

Побудувати алгоритм та налагодити консольний **Win32**–додаток, призначений для наближеного обчислення всіх абсцис $\tilde{x} \in [\, x_0, x_n \,]$, які для деякої заданої функції $f_0(x; a, b)$ задовольняють рівнянню:

$$f_0(\tilde{x}; a, b) = f_*, \qquad \text{(с.1.4.1)}$$

де f_* – деяке задане числове значення, $f_0(x; a, b)$ – функція, яка неперервна на інтервалі $x \in [\, x_0, x_n \,]$, a і b – її параметри.

Математичне зображення функції $f_0(x; a, b)$ нам невідомо, тому що вона задається у формі статичного метода класу **Functions**, який міститься у бібліотеці класів **MAC_Hidden_Data**.

Ці компоненти додаються вам окремо у вигляді скомпільованого файлу **MAC_Hidden_Data.dll**.

Вимоги до програмних компонентів

Основну програму обчислень треба розробляти в виді окремого проекту **MAC_CheckTask_1_4** консольного додатку **Win32** в робочому просторі **MAC_Petrenko**.

Для розв'язування поставленої задачі слід скористатися екземпляром класу **MAC_DLL.MyTableOfFunction**.

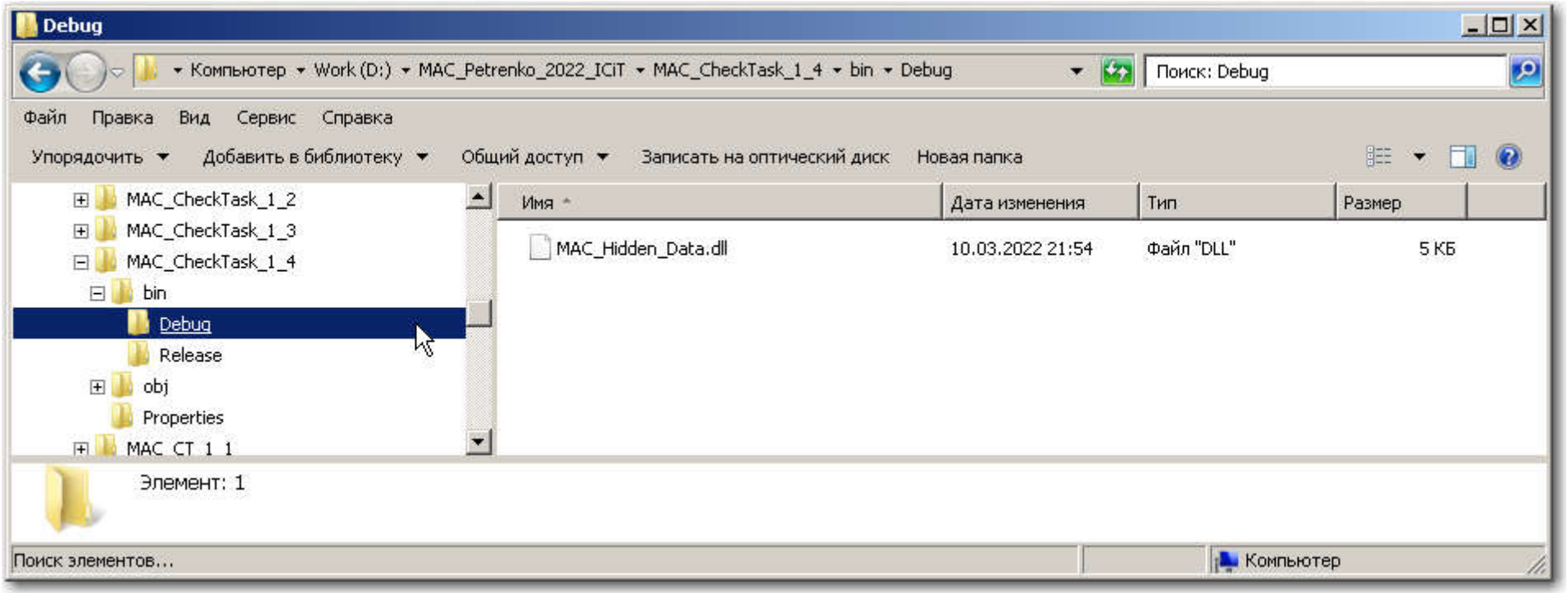
Обчислені розв'язки рівняння (1) повинні зберігатися у списку **Roots** зазначеного екземпляра **MyTableOfFunction**.

Додатковим результатом обчислень будемо вважати довжину відрізка між молодшим (першим) та старшим (останнім) коренем рівняння із отриманого списку **Roots**.

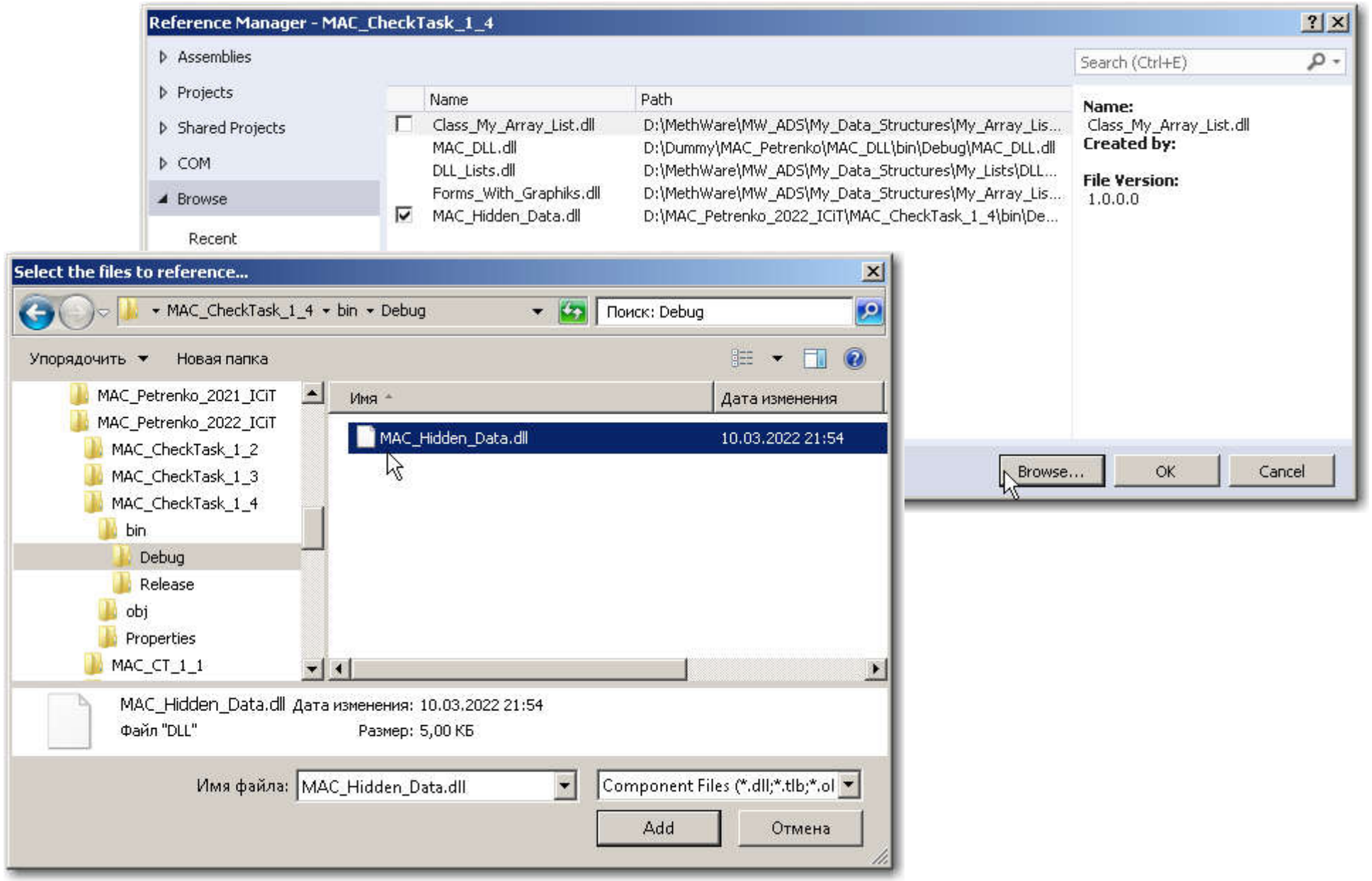
Етапи виконання завдання

В існуючому робочому просторі **`MAC_Petrenko`** згенеруємо новий проект **`MAC_CheckTask_1_4`** консольного додатку.

За допомогою «Провідника» скопіюємо файл **`MAC_Hidden_Data.dll`** до директорії **`Debug`** нового проекту:



Після цього додаємо його, та власну бібліотеку класів **`MAC_DLL`**, до переліку посилань **`References`** поточного проекту.



Далі.

За умовами завдання ми повинні скористатися екземпляром класу **MyTableOfFunction**.

В цьому класі ми вже реалізували алгоритм обчислення всіх нулів заданої функції на заданому інтервалі її визначення.

Конструктор цього класу приймає у якості фактичного параметру конкретну функцію, яка повинна своєю сигнатурою відповідати узагальненому делегату `Func<double, double> Fx;`.

Тобто, це повинна бути функція від однієї дійсної змінної та мати вид **Fx(x)**.

Але в бібліотеці **MAC_Hidden_Data.dll**, яка містить надану нам функцію $f_0(x;a,b)$ із невідомим математичним зображенням, структура всіх функцій інша, оскільки вони мають крім змінної ще й два додаткові параметри.

Тому в класі **Main_CT_1_4** ми повинні сформувати «допоміжний» статичний метод **Fx(x)**, призначений для обчислень значень функції $F_0(x) = f_0(x;a,b) - f_*$, яка моделюватиме нелінійне рівняння $f_0(\tilde{x};a,b) = f_*$, і яку ми зможемо передавати в конструктор класу **MyTableOfFunction**.

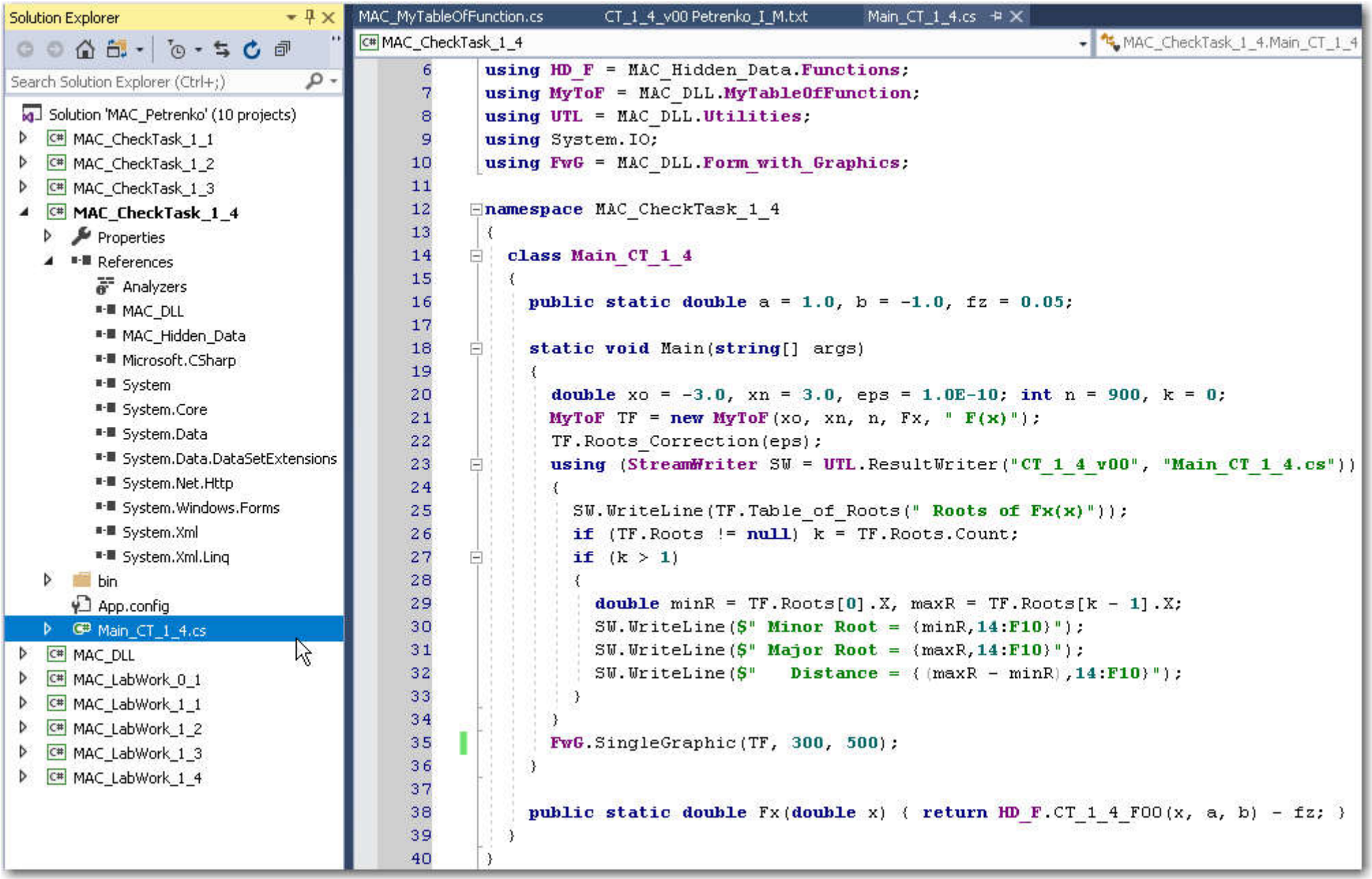
Крім того, в нашій програмі слід передбачити створення графіка функції, яка досліджується, з метою додаткового візуального аналізу завдання та впевненості у його остаточних результатах.

Тому слід до переліку посилань **References** додати ще й системну бібліотеку **System.Windows.Forms**.

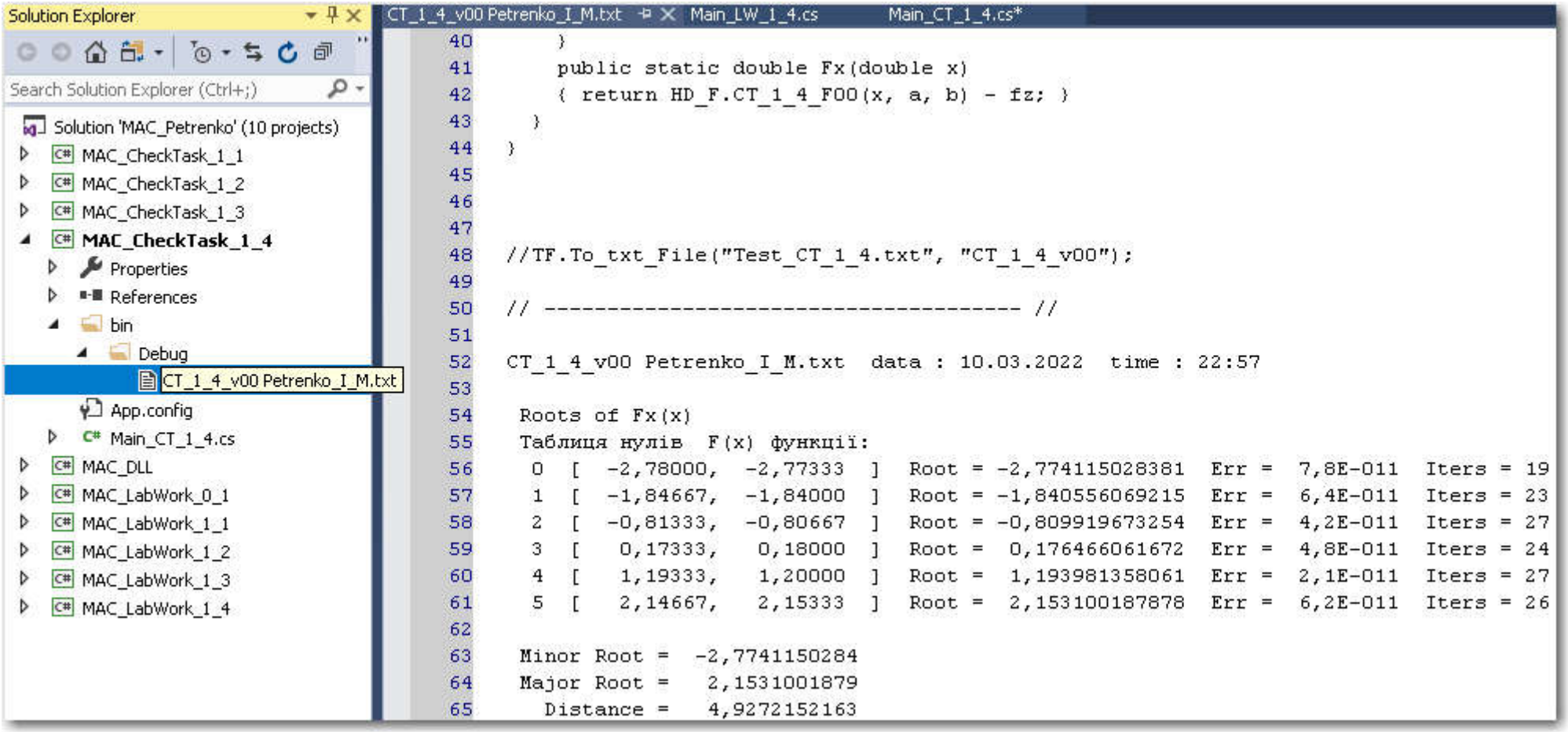
Отже, нехай таблиця параметрів контрольного завдання 1.4 має наступний вид:

f_*	a	b	x_0	x_n	n
+0.05	+1.0	-1.0	-3.0	+3.0	900

Повна програмна реалізація самостійного завдання:



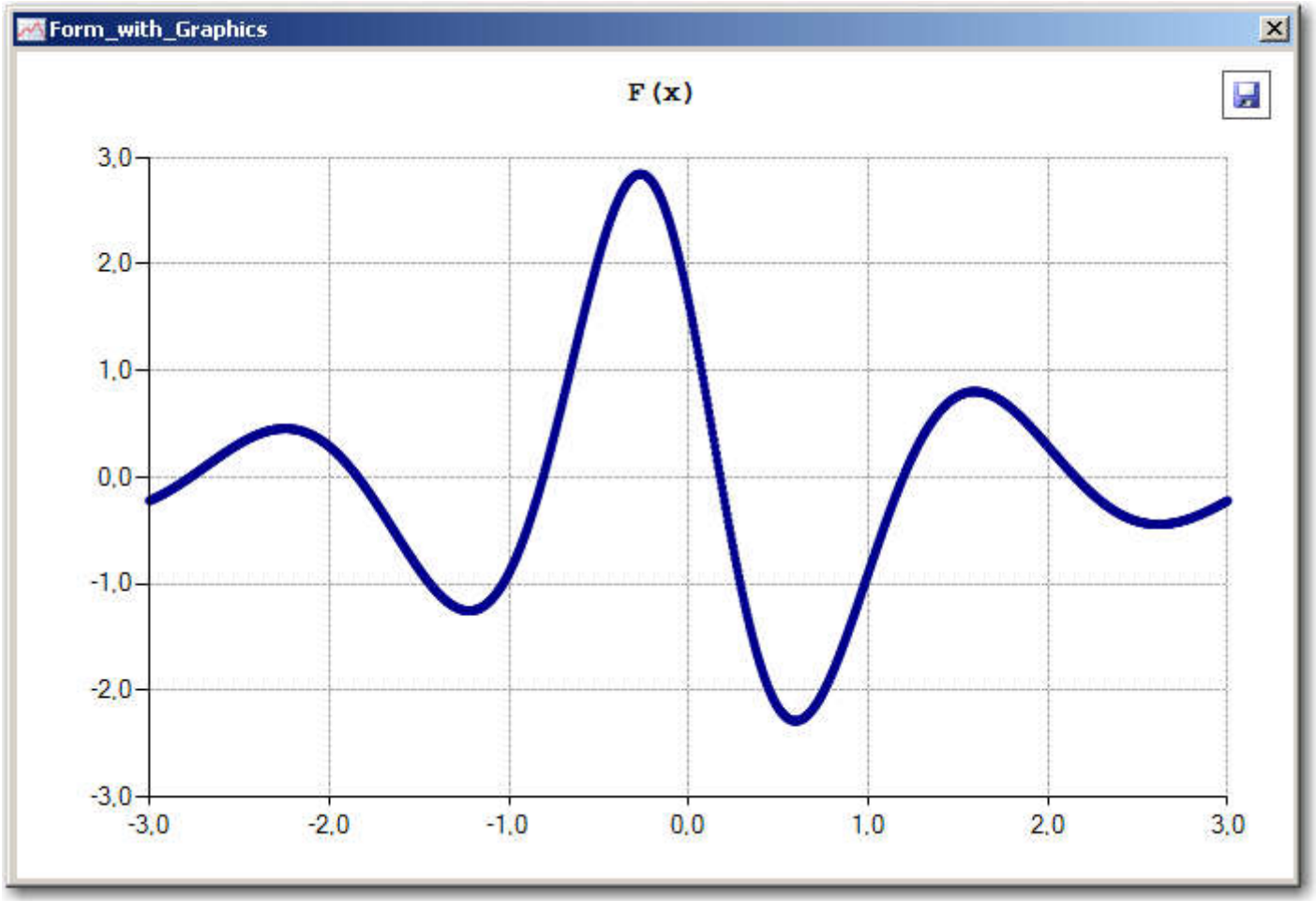
Файл результатів з повною таблицею розв’язків рівняння (с.1.4.1) «приєднаємо» до проекту:



В якості одного з контрольних результатів обчислень в основній програмі проекту **`MAC_CheckTask_1_4`** визначається відстань **`Distance`** між найбільшим та найменшим нулями функції **`Fx(x)`**.

Зверніть увагу, що функція $f_0(x;a,b)$ «запрограмована» у бібліотечному методі **`HD_F.CT_1_4_F00(x, a, b)`**.

Крім того, ви маєте можливість переглянути додатково і графік нашого тестового рівняння:



Вам залишилося виконати за даною схемою індивідуальний варіант контрольного завдання.

Файл результатів повинен обов'язково містити програмний код за яким виконувалися обчислення та таблицю результатів обчислень. Можна також додати й малюнок із графіком вашого рівняння.

Таблиця рівнянь та їх параметрів за варіантами

V	функція $F_V(x;a,b)$	f_*	x_0	x_n	a	b	n	V.v
1	HD_F.CT_1_4_F01(x, a, b)	+0.14	-4.5	+2.4	-0.21	+0.90	500	01.1
		-0.08	-3.1	+3.5	-0.15	-0.80	600	01.2
		+0.11	-2.7	+4.2	+0.17	+0.68	450	01.3
2	HD_F.CT_1_4_F02(x, a, b)	+0.03	-4.6	+3.2	-0.73	+0.54	600	02.1
		-0.06	-3.7	+2.4	+0.19	-0.72	430	02.2
		+0.09	-2.3	+4.6	-0.31	+0.49	540	02.3
3	HD_F.CT_1_4_F03(x, a, b)	-0.04	-4.2	+3.7	-0.82	-0.76	400	03.1
		+0.05	-2.6	+5.1	+0.44	-0.39	470	03.2
		-0.13	-3.3	+4.5	-0.27	+0.58	390	03.3
4	HD_F.CT_1_4_F04(x, a, b)	+0.17	-4.3	+3.9	+0.95	-0.81	500	04.1
		+0.15	-4.1	+5.2	-0.19	-0.62	530	04.2
		-0.07	-5.5	+3.6	+0.28	+0.75	450	04.3
5	HD_F.CT_1_4_F05(x, a, b)	-0.06	-4.5	+3.3	+0.72	+0.63	600	05.1
		+0.08	-4.9	+3.8	-0.19	-0.35	520	05.2
		-0.11	-3.3	+4.7	+0.67	+0.58	490	05.3

V	функція $F_V(x;a,b)$	f_*	x_0	x_n	a	b	n	V.v
6	HD_F.CT_1_4_F06(x, a, b)	+0.08	-4.7	+4.1	-0.88	+0.52	450	06.1
		-0.06	-3.7	+2.5	+0.39	-0.43	410	06.2
		+0.09	-2.8	+4.3	-0.36	+0.38	550	06.3
7	HD_F.CT_1_4_F07(x, a, b)	+0.13	-4.1	+4.4	+0.75	+0.86	530	07.1
		+0.05	-2.6	+5.2	+0.56	-0.49	470	07.2
		-0.07	-3.3	+4.9	+0.17	+0.61	390	07.3
8	HD_F.CT_1_4_F08(x, a, b)	-0.11	-5.2	+1.8	+0.64	-0.57	490	08.1
		+0.15	-4.1	+3.5	-0.19	-0.82	530	08.2
		-0.07	-5.5	+3.6	+0.28	+0.68	450	08.3
9	HD_F.CT_1_4_F09(x, a, b)	+0.05	-5.6	+1.4	+0.34	+0.48	620	09.1
		-0.06	-3.7	+4.4	+0.29	-0.72	430	09.2
		+0.09	-2.3	+4.6	-0.31	+0.59	540	09.3
10	HD_F.CT_1_4_F10(x, a, b)	-0.06	-7.6	-1.3	-0.38	+0.74	450	10.1
		+0.13	-5.8	+4.6	+0.19	+0.36	520	10.2
		-0.11	-3.3	+5.7	+0.24	+0.55	490	10.3

Таблиця тестових результатів за варіантами

V	x_0	x_n	a	b	n	f_*	$x_{**} - x_*$	k
1	-3.0	+3.0	-1.00	+1.00	500	+0.01	4.9627319708	6
2	-3.0	+3.0	-1.00	+1.00	500	-0.01	4.9898148164	6
3	-3.0	+3.0	-1.00	+1.00	500	-0.01	4.9479582190	6
4	-3.0	+3.0	-1.00	+1.00	500	+0.01	4.9160354110	6
5	-3.0	+3.0	-1.00	+1.00	500	+0.01	5.0015563336	6
6	-3.0	+3.0	-1.00	+1.00	500	-0.01	5.0153631815	6
7	-3.0	+3.0	-1.00	+1.00	500	-0.01	5.1032735669	6
8	-3.0	+3.0	-1.00	+1.00	500	+0.01	4.9354224297	6
9	-3.0	+3.0	-1.00	+1.00	500	+0.01	4.9572919618	6
10	-3.0	+3.0	-1.00	+1.00	500	-0.01	4.9818438900	6

В якості тестового результату в таблиці записана дистанція між молодшим x_* та старшим x_{**} коренями рівняння (с.1.4.1).

Мається на увазі, що молодший корінь x_* – це той, що має найменшу абсцису серед усіх коренів, що були обчислені для заданого інтервалу $[x_0; x_n]$, а старший корінь x_{**} – це той, що має найбільшу абсцису.

Параметр k визначає загальну кількість коренів рівняння на інтервалі $[x_0; x_n]$.

Лабораторна робота № 1.5

«Дослідження збіжності ітераційних схем методу Ньютона»

Основною характеристикою методів розв'язування нелінійних рівнянь $F(x) = 0$ є кількість ітерацій, які були витрачені цими чисельними алгоритмами на уточнення кореня.

Вочевидь, що ті методи, які мають найбільшу швидкість збіжності, потребують обов'язкового виконання певних додаткових умов для функції $F(x)$.

І навпаки, методи з малою швидкістю збіжності (наприклад, дихотомії), як правило, ніяких додаткових умов на функцію $F(x)$ не накладають.

В класичній теорії наближеного розв'язування нелінійних рівнянь відоме сімейство алгоритмів, яке називають методами дотичних (або методами Ньютона).

Ці методи мають саму високу швидкість збіжності по відношенню до інших ітераційних алгоритмів.

Підтвердимо цей факт в рамках даної лабораторної роботи.

Мета лабораторної роботи

Побудувати та реалізувати алгоритми для двох ітераційних схем з сімейства методів дотичних та налагодити консольну програму обчислення коренів тестового рівняння $F(x) = 0$ на інтервалі $x \in [x_0; x_n]$ із заданою абсолютною похибкою $\varepsilon \approx 10^{-12}$.

Здійснити порівняльний аналіз швидкості збіжності обох схем (за кількістю виконаних ітерацій) на заданому тестовому рівнянні $F(x) = 0$.

***Ітераційна схема* 1**: $$x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(k)})}, \qquad \text{(л.1.5.1)}$$

де в якості нульового наближення $x^{(0)}$ до шуканого кореня обирається та з границь інтервалу $[x_{i-1}; x_i]$ таблиці функції, для якої виконується додаткова умова $F(x^{(0)}) \cdot F''(x^{(0)}) > 0$.

***Ітераційна схема* 2**: $$x^{(k+1)} = x^{(k)} - \frac{F(x^{(k)})}{F'(x^{(0)})},$$

в її спрощеному варіанті $$x^{(k+1)} = x^{(k)} - \delta_0 \cdot F(x^{(k)}), \qquad \text{(л.1.5.2)}$$

де $$\delta_0 = \frac{x_i - x_{i-1}}{F(x_i) - F(x_{i-1})} \approx \left[F'(x^{(0)}) \right]^{-1}, \qquad \text{(л.1.5.3)}$$ із початковим наближенням $x^{(0)} = \frac{x_i + x_{i-1}}{2}$.

Для обох зазначених схем умовою завершення ітераційного процесу слід вважати виконання нерівності $| F(x^{(k)}) | \le \varepsilon$.

Для запобігання можливих зациклень слід додати умови виходу з ітераційного процесу: при $k > 15$ для схеми 1 та при $k > 25$ для схеми 2.

Розв’язування поставленого завдання слід виконувати в два етапи.

Перший етап – включення нових методів до складу класу **`Class_Equations`** динамічної бібліотеки **`MAC_DLL`**, які будуть містити програмні реалізації ітераційних схем уточнення коренів нелінійних рівнянь, які побудовані на основі алгоритмів (л.1.5.1) та (л.1.5.2)–(л.1.5.3).

Обидва алгоритми можна об’єднати під одним іменем **`Tangent()`** та використати перевантаження, оскільки програмні реалізації для схем (л.1.5.1) та (л.1.5.2)–(л.1.5.3), вочевидь, повинні мати різні набори вхідних параметрів.

Обов’язковими параметрами методу, який реалізує ***схему (л.1.5.1)***, повинні бути:

- границі інтервалу $[x_{i-1}; x_i]$, для якого виконується умова $F(x_{i-1}) \cdot F(x_i) < 0$;
- делегати, які мають сигнатуру функції від дійсної змінної, для зображення математичних функцій $F(x)$, $F'(x)$ та $F''(x)$;
- ε – задана абсолютна похибка уточнення кореня;
- K – кількість виконаних ітерацій (параметр, який повертає відповідне числове значення).

Обов’язковими параметрами методу, який реалізує ***схему (л.1.5.2)***, повинні бути:

- границі інтервалу $[x_{i-1}; x_i]$, для якого виконується умова $F(x_{i-1}) \cdot F(x_i) < 0$;
- делегат, який має сигнатуру функції від дійсної змінної, для зображення математичної функції $F(x)$;
- ε – задана абсолютна похибка уточнення кореня;
- K – кількість виконаних ітерацій (параметр, який повертає відповідне числове значення).

Результатом виклику обох методів в основному консольному додатку повинно бути уточнене значення кореня $\tilde{x}^{(K)}$ рівняння $F(x) = 0$, яке задовольняє умовам $x_{i-1} \le \tilde{x}^{(\mathrm{K})} \le x_i$ та $| F(x^{(\mathrm{K})}) | \le \varepsilon$.

Другий етап (основна програма) – етап обчислення списку всіх коренів нелінійного рівняння $F(x) = 0$ на заданому інтервалі аргументу $x \in [x_0; x_n]$.

Уточнення кожного кореня $\tilde{x}$ здійснюється заданим чисельним методом.

Цей етап виконується практично так само, як і в попередній лабораторній роботі.

Вимоги до програмних компонентів

Нові програмні компоненти, які розробляються, повинні бути реалізованими в складі існуючої динамічної бібліотеки **MAC_DLL**, проект якої вже існує в Вашому робочому просторі **MAC_Petrenko**.

Основна програмна одиниця даної лабораторної роботи повинна бути генерованою в рамках окремого проекту (типу **Console Application** з м.'ям **MAC_LabWork_1_5** в робочому просторі **MAC_Petrenko**), який має доступ до динамічної бібліотеки, в якій визначені класи **MAC_MyTableOfFunction** та **MAC_Equations**.

Результати виконання основної програми – дві таблиці коренів тестового рівняння – повинні зберігатися в текстовому файлі **LW_1_5_Results.txt**, який необхідно включити до проекту **MAC_LabWork_1_5**.

Порядок виконання завдання лабораторної роботи

Почнемо з розробки програмних реалізацій алгоритмів (л.1.5.1) та (л.1.5.2)–(л.1.5.3) уточнення коренів для нелінійного рівняння $F(x) = 0$. Ці методи додамо до вже існуючого класу **Class_Equations** динамічної бібліотеки **MAC_DLL**.

Перший метод, з м.'ям **Tangent()** буде відповідати алгоритму (л.1.5.1), а другий – вже перевантажений – буде виконувати ітерації згідно із схемою (л.1.5.2)–(л.1.5.3).

```
 9   public class MAC_Equations
10   {
11     public static double Dichotomy(double a, double b, double eps, Func<double, double> f, out int K)...
23     public static void Dichotomy(Func<double, double> f, Root root, double eps)...
36
37     public static double Tangent(Func<double, double> Fx, Func<double, double> D1Fx, Func<double, double> D2Fx,
38                                  double xL, double xR, double eps, out int K)
39     {
40       double xK = (xR + xL) * 0.5; K = 0;
41       if ((Fx(xL) * D2Fx(xL)) > 0) xK = xL; if ((Fx(xR) * D2Fx(xR)) > 0) xK = xR;
42       while (Math.Abs(Fx(xK)) > eps)
43       {
44         xK = xK - Fx(xK) / D1Fx(xK); K++; if (K > 15) break;
45       }
46       return xK;
47     }
48
49     public static double Tangent(Func<double, double> Fx, double xL, double xR, double eps, out int K)
50     {
51       double xK = (xR + xL) * 0.5; K = 0;
52       double dF = (Fx(xR) - Fx(xL)) / (xR - xL);
53       while (Math.Abs(Fx(xK)) > eps)
54       {
55         xK = xK - Fx(xK) / dF; K++; if (K > 25) break;
56       }
57       return xK;
58     }
59   }
```

Тепер перейдемо до основного консольного додатка лабораторної роботи.

Задача цієї програми – тестування методів **`Tangent()`** на модельному рівнянні:

$$F(x) = \mathrm{th}(x) - 2\cos(\sqrt{10}\cdot x) = 0\,, \quad [x_0; x_n] = [\,0\,;\,15\,], \quad \varepsilon \approx 10^{-12}. \qquad \text{(л.1.5.4)}$$

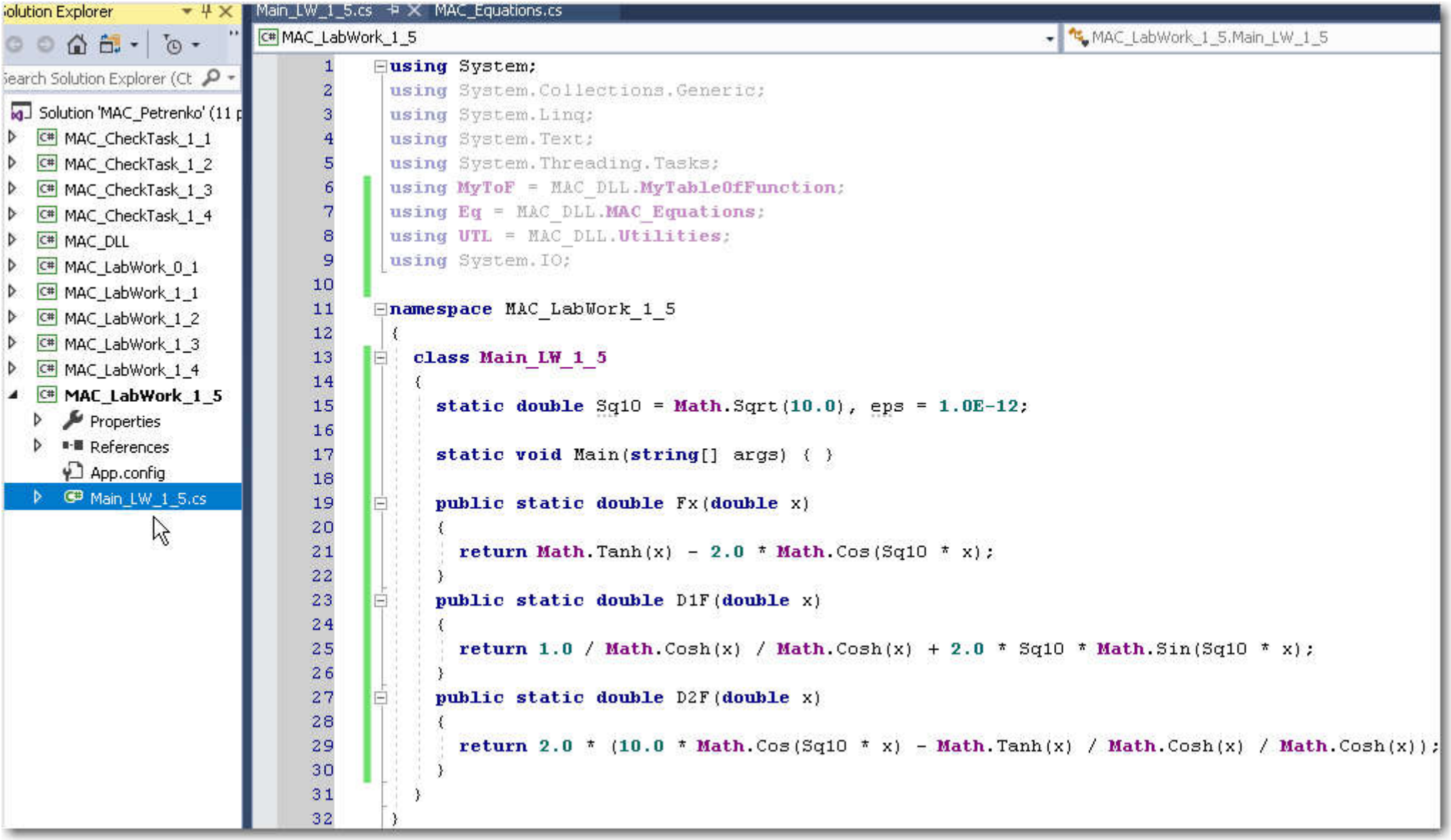
Це рівняння нам зручне тим, що ми вже обчислили його корені під час виконання попередньої лабораторної роботи 1.4 і нам є з чим порівнювати нові результати.

Для реалізації виклику схеми (л.1.5.1) окрім функції $F(x)$ будуть потрібні перша $F'(x)$ та друга $F''(x)$ похідні в формі відповідних статичних методів в класі основного консольного додатку.

Тому попередньо знаходимо аналітичні вирази для функції $F'(x)$ і $F''(x)$:

$$F'(x) = \mathrm{ch}^{-2}(x) + 2\sqrt{10}\sin(\sqrt{10}\cdot x)\,, \qquad F''(x) = 2\cdot\left(10\cos(\sqrt{10}\cdot x) - \mathrm{th}(x)\,\mathrm{ch}^{-2}(x)\right). \qquad \text{(л.1.5.5)}$$

Виконуємо «підготовчу роботу»:



В функції **Main()** класу **MAC_LabWork_1_5** будемо виконувати наступні дії:

1) створимо об'єкт **T_Fx** класу **MAC_MyTableOfFunction**, який відповідатиме таблиці функції (л.1.5.4);

2) здійснимо уточнення нулів для отриманої таблиці функції із використанням методу дихотомії і запишемо в файл результатів **LW_1_5_Results.txt** відповідну таблицю **Table_of_Roots()**;

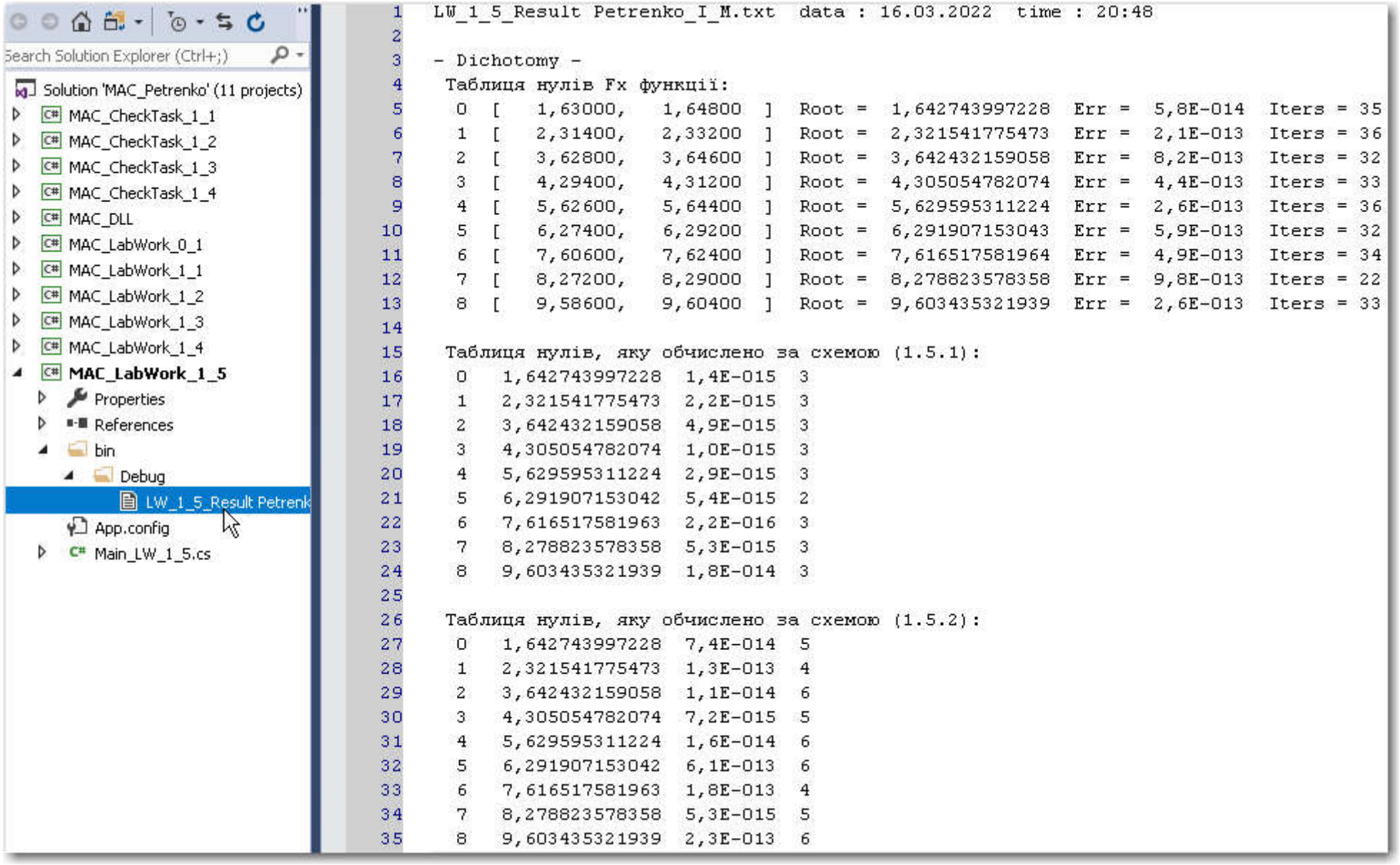
3) скористуємося списком **Roots** таблиці **T_Fx** для організації циклів обчислення тих самих нулів, але вже із використанням методів **Tangent()**, щойно розроблених вище;

4) результати будемо дописувати в текстовий файл **LW_1_5_Results.txt**.

Код основної функції **`Main()`** може бути наступним:

```
11  namespace MAC_LabWork_1_5
12  {
13    class Main_LW_1_5
14    {
15      static double Sq10 = Math.Sqrt(10.0), eps = 1.0E-12;
16
17      static void Main(string[] args)
18      {
19        using (StreamWriter SW = UTL.ResultWriter("LW_1_5_Result"))
20        {
21          MyToF T_Fx = new MyToF(1.0, 10.0, 500, Fx, "Fx");
22          T_Fx.Roots_Correction(eps);
23          SW.Write(T_Fx.Table_of_Roots("- Dichotomy -"));
24
25          double xr = double.NaN;
26
27          SW.WriteLine("\r\n Таблиця нулів, яку обчислено за схемою (1.5.1):");
28          for (int j = 0; j < T_Fx.Roots.Count; j++)
29          {
30            xr = Eq.Tangent(Fx, D1F, D2F, T_Fx.Roots[j].XL, T_Fx.Roots[j].XR, eps, out int K);
31            SW.WriteLine($"{j,3}{xr,17:F12}{Math.Abs(Fx(xr)),10:E1}{K,3}");
32          }
33
34          SW.WriteLine("\r\n Таблиця нулів, яку обчислено за схемою (1.5.2):");
35          for (int j = 0; j < T_Fx.Roots.Count; j++)
36          {
37            xr = Eq.Tangent(Fx, T_Fx.Roots[j].XL, T_Fx.Roots[j].XR, eps, out int K);
38            SW.WriteLine($"{j,3}{xr,17:F12}{Math.Abs(Fx(xr)),10:E1}{K,3}");
39          }
40        }
41      }
42      public static double Fx(double x)...
46      public static double D1F(double x)...
50      public static double D2F(double x)...
54    }
55  }
```

Файл результатів **`LW_1_5_Results.txt`** повинен містити наступні числові дані:




```
LW_1_5_Result Petrenko_I_M.txt  data : 16.03.2022  time : 20:48

- Dichotomy -
 Таблиця нулів Fx функції:
  0  [   1,63000,   1,64800  ]  Root =  1,642743997228  Err =  5,8E-014  Iters = 35
  1  [   2,31400,   2,33200  ]  Root =  2,321541775473  Err =  2,1E-013  Iters = 36
  2  [   3,62800,   3,64600  ]  Root =  3,642432159058  Err =  8,2E-013  Iters = 32
  3  [   4,29400,   4,31200  ]  Root =  4,305054782074  Err =  4,4E-013  Iters = 33
  4  [   5,62600,   5,64400  ]  Root =  5,629595311224  Err =  2,6E-013  Iters = 36
  5  [   6,27400,   6,29200  ]  Root =  6,291907153043  Err =  5,9E-013  Iters = 32
  6  [   7,60600,   7,62400  ]  Root =  7,616517581964  Err =  4,9E-013  Iters = 34
  7  [   8,27200,   8,29000  ]  Root =  8,278823578358  Err =  9,8E-013  Iters = 22
  8  [   9,58600,   9,60400  ]  Root =  9,603435321939  Err =  2,6E-013  Iters = 33

 Таблиця нулів, яку обчислено за схемою (1.5.1):
  0   1,642743997228  1,4E-015  3
  1   2,321541775473  2,2E-015  3
  2   3,642432159058  4,9E-015  3
  3   4,305054782074  1,0E-015  3
  4   5,629595311224  2,9E-015  3
  5   6,291907153042  5,4E-015  2
  6   7,616517581963  2,2E-016  3
  7   8,278823578358  5,3E-015  3
  8   9,603435321939  1,8E-014  3

 Таблиця нулів, яку обчислено за схемою (1.5.2):
  0   1,642743997228  7,4E-014  5
  1   2,321541775473  1,3E-013  4
  2   3,642432159058  1,1E-014  6
  3   4,305054782074  7,2E-015  5
  4   5,629595311224  1,6E-014  6
  5   6,291907153042  6,1E-013  6
  6   7,616517581963  1,8E-013  4
  7   8,278823578358  5,3E-015  5
  8   9,603435321939  2,3E-013  6
```

Таблиця коренів тестового рівняння (л.1.5.4), яку ми отримали як результат застосування методу Ньютона за схемою (л.1.5.1), демонструє збіжність до кореня за 3 ітерації (всього 26 ітерацій при послідовному обчисленні 9 коренів).

Досягнута точність при обчисленні кореня виявилася вищою, ніж була «запланована».

Схема Ньютона (л.1.5.2)–(л.1.5.3) показала «в середньому» в 1.8 рази більше ітерацій на кожному з коренів, а загальна кількість їх склала 47.

При цьому довжина інтервалу $[x_{i-1}; x_i]$, на якому здійснювалося уточнення коренів, була одна й та сама для обох схем.

Метод дихотомії, який був протестований нами на цьому ж рівнянні, показав в середньому по 33 ітерацій на корінь при тій же довжині інтервалу $[x_{i-1}; x_i]$.

Це більше ніж в 10 разів у порівнянні із схемою (л.1.5.1) методу дотичних, і більше ніж в 6 разів по відношенню до схеми (л.1.5.2)–(л.1.5.3).

Отже, нові методи наближеного розв'язування нелінійного рівняння в виді $F(x) = 0$ налагоджені нами та протестовані.

Лише після того, як тестові обчислення дали Вам шуканий результат, Вам слід виконати аналогічні обчислення для Вашого персонального варіанту лабораторної роботи 1.5.

Різниця буде не тільки в математичному виді самої функції $F(x)$ та двох її похідних $F'(x_*)$ і $F''(x_*)$.

На інтервалі $x \in [x_0; x_n]$, який буде Вам запропонований, функція $F(x)$ буде мати тільки один нуль x_*.

Для знайденого значення цього кореня x_* слід додатково обчислити значення функцій $F'(x_*)$ та $F''(x_*)$, а потім звести ці результати в «єдину таблицю».

Рекомендація. При реалізації індивідуального завдання слід ***обов'язково зберегти*** код основного додатку, який був розроблений для тестових рівнянь.

Таблиця рівнянь та їх параметрів за варіантами

V	рівняння $F_V(x)=0$	x_0	x_n	n	V.v
1	$F_1(x)=3\cos\left(\sqrt{11}\cdot x\right)-(x-1)^2=0$	+0.0	+1.0	310	01.1
		+1.0	+2.0	400	01.2
		+2.0	+3.0	250	01.3
2	$F_2(x)=\sin(2x+1)-\frac{1}{4}\sqrt{5-x}=0$	-2.8	-1.3	320	02.1
		-1.3	+0.3	390	02.2
		+0.3	+1.8	260	02.3
3	$F_3(x)=\ln\left[e\cdot(x+\pi)\right]+4\sin(2x+1)=0$	-1.5	-0.5	330	03.1
		+1.0	+2.0	380	03.2
		+2.0	+3.5	270	03.3
4	$F_4(x)=\cos(\sqrt{7}x-1)+1-e^{-x/\pi}=0$	-0.5	+0.5	340	04.1
		+0.5	+1.5	370	04.2
		+1.5	+2.5	280	04.3
5	$F_5(x)=3\sin(\pi x+2)-\frac{2x+1}{x+1}=0$	+1.0	+2.0	350	05.1
		+2.0	+2.5	360	05.2
		+3.0	+4.0	290	05.3

V	рівняння $F_V(x)=0$	x_0	x_n	n	V.v
6	$F_6(x)=0.5\cos(2x)-\frac{x}{1+x^2}=0$	-4.5	-3.5	360	06.1
		-3.5	-1.5	350	06.2
		-1.5	+0.0	300	06.3
7	$F_7(x)=\frac{\sqrt{x+7}}{\pi}+e\cdot\sin(3x+4)=0$	+0.5	+1.5	370	07.1
		+1.5	+2.5	340	07.2
		+2.5	+3.5	310	07.3
8	$F_8(x)=1+e^{0.3x}+\pi\cos(e\cdot x-1)=0$	-6.5	-5.5	380	08.1
		-5.5	-4.5	330	08.2
		-4.0	-3.0	300	08.3
9	$F_9(x)=\frac{1}{\pi}\ln(2e-x)+\sin(2x-e)=0$	-5.5	-4.0	390	09.1
		-4.0	-2.5	320	09.2
		-2.5	-1.0	290	09.3
10	$F_{10}(x)=e^{x/4}\sin(4x)-\frac{x+7}{\pi^2}=0$	-4.3	-3.5	400	10.1
		-3.5	-2.7	310	10.2
		-2.7	-2.0	280	10.3

Таблиця тестових результатів за варіантами

V	x_0	x_n	n	x_*	$F'(x_*)$	$F''(x_*)$	k
1	-1.0	+0.0	500	-0.2948170325	+10.8407514890	-20.4420626232	3
2	-4.0	-2.8	500	-3.2413502326	+1.4362598922	-2.8681335034	2
3	-2.0	-1.5	500	-1.9182410779	-6.8130836176	+4.1381910539	2
4	-2.0	-1.0	500	-1.7083236113	-1.2809090226	-5.2320204836	2
5	0.0	+0.5	500	+0.2336191178	-9.3095498696	-10.6733444222	2
6	-6.0	-5.0	500	-5.3117016429	-0.8996535022	+0.7380203744	2
7	-0.5	+0.0	500	-0.1825604740	-7.7033701597	+7.4757373378	2
8	-7.5	-6.5	500	-7.2775323739	+8.0199835365	+8.2317486310	3
9	-7.0	-5.5	500	-6.0498602121	-1.2865964753	+3.1057762651	3
10	-5.0	-4.3	500	-4.4880759060	+0.7747457417	-3.6500514413	3

Зразок файлу результатів (для тестового рівняння (л.1.5.4)–(л.1.5.5))

```
Main_LW_1_5 Petrenko_I_M.txt  data : 16.03.2022  time : 23:01

---------------------- LW_1_5_v00 -------------------------
  xo =  1,0  xn =  2,0  n = 500
  0  xr =   1,642743997228   err =  9,3E-013  K =  2
    d1f =  -5,463668814212   d2f =   9,020450874276

```

Не забудьте додати до файлу результатів код основної програми !

Методика виконання самостійного завдання № 1.5

«Обчислення коренів нелінійного рівняння методом Ньютона (дотичних)»

Постановка завдання

Побудувати алгоритм та налагодити консольний **Windows**–додаток, призначений для розв'язування наступної практичної задачі (простий чисельний експеримент).

Нехай нам відомі рівняння траєкторій $y_1 = f_1(x)$ і $y_2 = f_2(x)$ для двох матеріальних частинок, які одночасно здійснюють рух в площині Oxy. Треба:

1. з'ясувати, чи існують точки перетину цих траєкторій на заданому інтервалі зміни просторової координати $x \in [x_0, x_n]$ (етап виділення коренів);

2. за допомогою метода Ньютона (схема (1.5.1) з лабораторної роботи 1.5) із абсолютною похибкою $\varepsilon \sim 10^{-10}$ визначити наближені розв'язки $\tilde{x}_i$ рівняння $f_1(x) = f_2(x)$ на частинних інтервалах $[x_{i-1}, x_i]$, які були виявлені на попередньому етапі;

3. обчислити ординати $\tilde{y}_i = f_1(\tilde{x}_i) = f_2(\tilde{x}_i)$ точок перетину траєкторій та записати числові результати в таблицю із десятьма цифрами після десятинної точки.

Для розв'язування поставленої задачі слід скористатися вже існуючими програмними компонентами з бібліотеки **MAC_DLL** та за необхідністю поповнити їх.

Попередні зауваження

Перед тим як приступати до безпосереднього програмування, нам слід привести дану математичну задачу к такій формі, яка допускає застосування чисельного методу дотичних (схема (л.1.5.1)) до її розв'язування.

Отже, задане рівняння для траєкторій точок $f_1(x) = f_2(x)$, які перетинаються, замінюємо на еквівалентне рівняння $F(x) = f_1(x) - f_2(x) = 0$.

Тепер ми можемо аналізувати таблицю функції $F(x)$ на інтервалі $[x_0, x_n]$ на предмет наявності інтервалів $[x_{i-1}, x_i]$, які містять прості нулі цієї функції.

Оскільки нам належить застосовувати метод дотичних за схемою (л.1.5.1), слід попередньо аналітичним шляхом отримати математичні вирази (формули) для похідних $F'(x)$ і $F''(x)$.

Вочевидь, що $F'(x) = f_1'(x) - f_2'(x)$ та $F''(x) = f_1''(x) - f_2''(x)$.

Вимоги до програмних компонентів

Основна обчислювальна програма повинна бути розроблена в виді окремого проекту **`MAC_CheckTask_1_5`** консольного додатка **`Win32`** в існуючому робочому просторі **`MAC_Petrenko`**.

Власно створення таблиці функції $F(x)$, слід реалізувати в екземплярі нового класу **`MyExtendedTableOfFunction`** (розширена таблиця функції), який є нащадком класу **`MyTableOfFunction`**.

В конструкторі нового класу повинні бути два додаткові формальні параметри – узагальнені делегати типа **`Func<double,double>`**, які призначені для передачі «лінків» на функції (від англійського **`link`**), які мають обчислювати значення першої та другої похідних від основної функції $F(x)$.

Метод уточнення нулів – це новий, перевантажений метод **`Tangent()`**, в класі **`My_Equations`**, який реалізується за схемою (л.1.5.1) та сигнатура якого надалі буде вказана.

Методика виконання варіанту завдання

В існуючому робочому просторі **`MAC_Petrenko`** генеруємо (додаємо) новий проект **`MAC_CheckTask_1_5`** консольного додатку. Додаємо до переліку **`References`** проекту **`MAC_CheckTask_1_5`** посилання на проект **`MAC_DLL`**.

Нехай ми маємо наступну математичну постановку задачі:

Рівняння траєкторій:

$$y_1 = f_1(x) = \text{th}(x + a)$$
$$y_2 = f_2(x) = 2\cos(\sqrt{10} \cdot x + b)$$

Таблиця параметрів задачі:

x_0	x_n	a	b
`0.0`	`4.0`	`0.1`	`-0.1`

Функція $F(x)$, яка досліджується, представляє собою різницю $F(x) = f_1(x) - f_2(x)$ двох заданих функцій з параметрами.

Теж саме буде стосуватися першої $F'(x)$ і другої $F''(x)$ похідних.

Знайдемо математичні вирази для цих похідних:

$$f_1'(x) = \text{ch}^{-2}(x + a), \qquad f_1''(x) = 2 \cdot \text{th}(x + a) \cdot \text{ch}^{-2}(x + a),$$
$$f_2'(x) = -2\sqrt{10} \cdot \sin(\sqrt{10}x + b), \quad f_2''(x) = -20 \cdot \cos(\sqrt{10}x + b).$$

Параметри задачі a і b слід декларувати в основній програмі як статичні змінні (та одночасно дати їм числові значення, які зазначені в таблиці параметрів), а також створити коди відповідних математичних функцій.

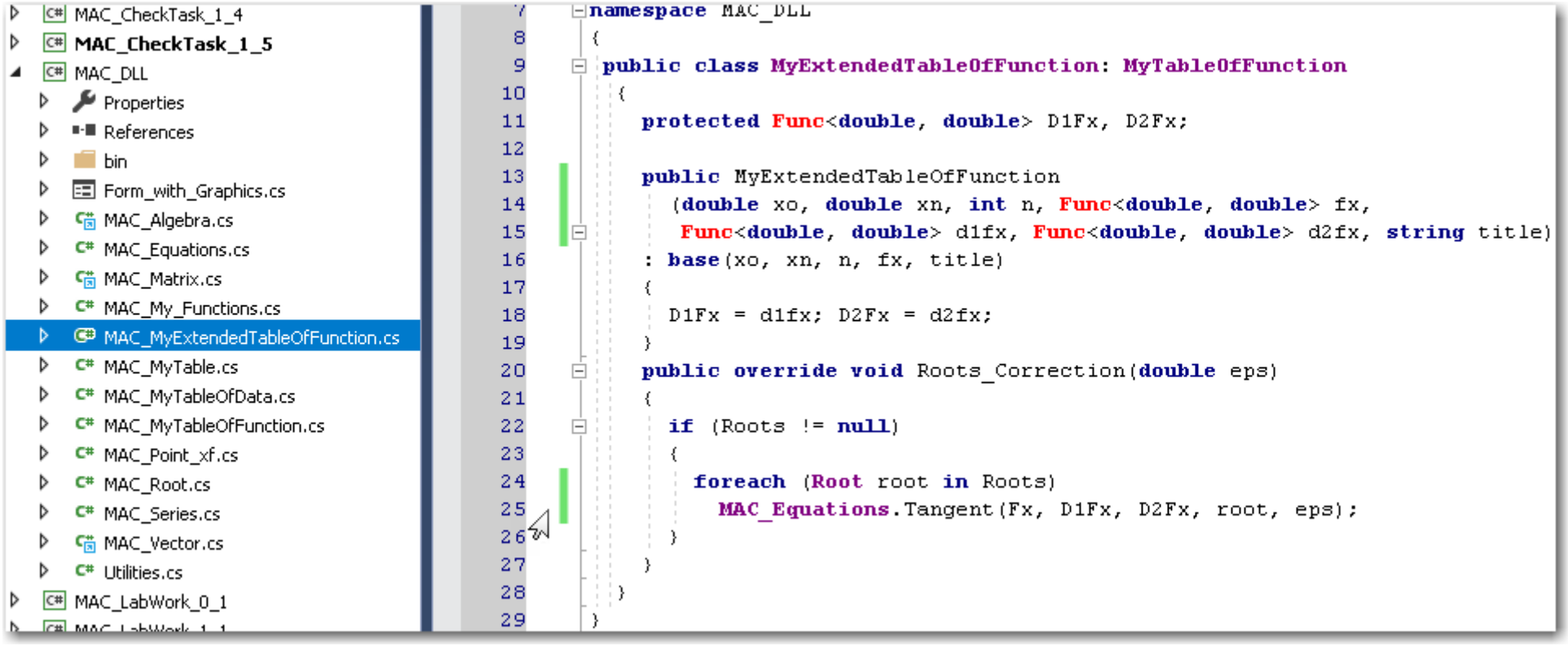
Отже, первісна структура додатку **MAC_CheckTask_1_5** може бути наступною:

```
6   using EToF = MAC_DLL.MyExtendedTableOfFunction;
7   using UTL = MAC_DLL.Utilities;
8   using System.IO;
9
10  namespace MAC_CheckTask_1_5
11  {
12    class Main_CT_1_5
13    {
14      static double a = 0.1, b = -0.1, s = Math.Sqrt(10.0);
15
16      static void Main(string[] args)
17      {
18        using (StreamWriter SW = UTL.ResultWriter("CT_1_5"))
19        {
20          <-- Розв'язування завдання -->
36        }
37      }
38      public static double Fx(double x) { return f1(x) - f2(x); }
39      public static double d1Fx(double x) { return d1f1(x) - d1f2(x); }
40      public static double d2Fx(double x) { return d2f1(x) - d2f2(x); }
41
42
43      public static double f1(double x) { return Math.Tanh(x + a); }
44
45      public static double d1f1(double x) { return 1.0 / Math.Cosh(x + a) / Math.Cosh(x + a); }
46
47      public static double d2f1(double x) { return -2.0 * Math.Tanh(x + a) / Math.Cosh(x + a) / Math.Cosh(x + a); }
48
49      public static double f2(double x) { return 2.0 * Math.Cos(x * s + b); }
50
51      public static double d1f2(double x) { return -2.0 * s * Math.Sin(x * s + b); }
52
53      public static double d2f2(double x) { return -2.0 * s * s * Math.Cos(x * s + b); }
54    }
55  }
```

Тепер займемося формуванням нового класу **MyExtendedTableOfFunction** – розширеної таблиці функції.

Новий клас **MyExtendedTableOfFunction** є нащадком класу **MyTableOfFunction** і розміщується в існуючій бібліотеці класів **MAC_DLL**.

В новому класі перевизначимо віртуальний метод **Roots_Correction()** уточнення нулів функції із застосуванням нового перевантаженого метода Ньютона:



Перевантажений метод **Tangent()** в класі **My_Equations** (на ділянку коду із його сигнатурою вказує стрілка миші) Вам слід запрограмувати самостійно, взявши за основу вже налагоджений аналогічний метод на основі алгоритму (л.1.5.1).

Якщо цей етап роботи викликає у Вас певні труднощі, то код нового методу **`Tangent()`** буде розташовано в кінці даного методичного матеріалу.

Після того, як ця частина роботи буде вже Вами виконана, можна скористатися розробленими компонентами та за їх допомогою розв’язати поставлену задачу:

```
17      static double a = 0.1, b = -0.1, s = Math.Sqrt(10.0);
18
19      static void Main(string[] args)
20      {
21        using (StreamWriter SW = UTL.ResultWriter("CT_1_5"))
22        {
23          #region    <-- Розв'язування завдання -->
24
25          EToF ET_Fx = new EToF(0.0, 4.0, 200, Fx, d1Fx, d2Fx, " - Newton -");
26          ET_Fx.Roots_Correction(1.0E-12);
27
28          SW.WriteLine(ET_Fx.Table_of_Roots("--- All Roots ---"));
29
30          SW.WriteLine("\r\n  Точки перетину траєкторій: \r\n");
31          FTN xy;
32          for (int i = 0; i < ET_Fx.Roots.Count; i++)
33          {
34            xy = new FTN(ET_Fx.Roots[i].X, f1(ET_Fx.Roots[i].X));
35            SW.WriteLine($"{i,3}" + xy.ToPrint());
36          }
37          #endregion <-- Розв'язування завдання -->
38        }
39      }
```

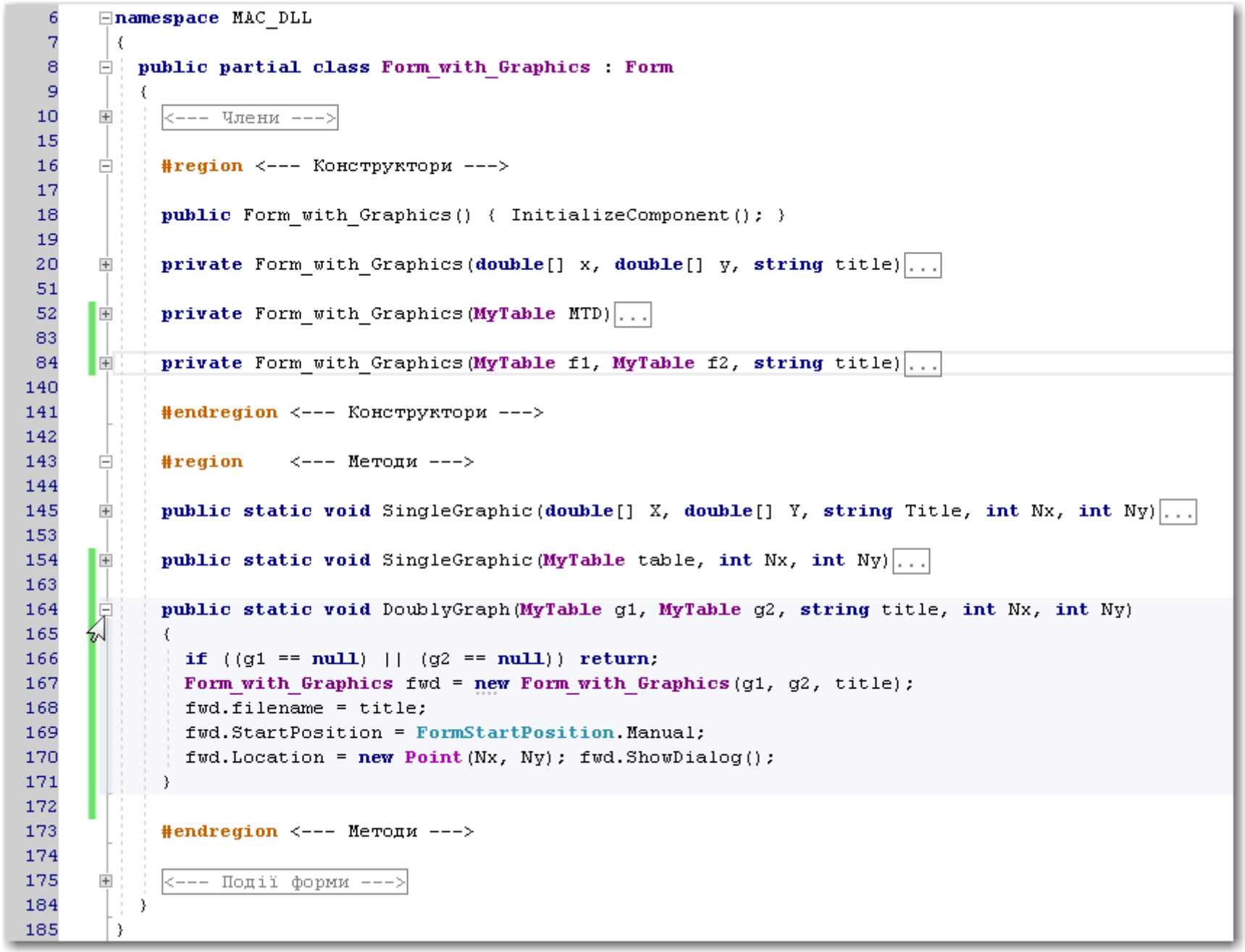
Результати обчислень зберігаються в текстовому файлі **`CT_1_5 Petrenko_I_M.txt`**:

Не забудьте потім додати до цього файлу код основної програми.

Для більш переконливого аналізу результатів нам не вистачає візуального спостереження точок перетину графіків двох функцій $y = f_1(x)$ і $y = f_2(x)$.

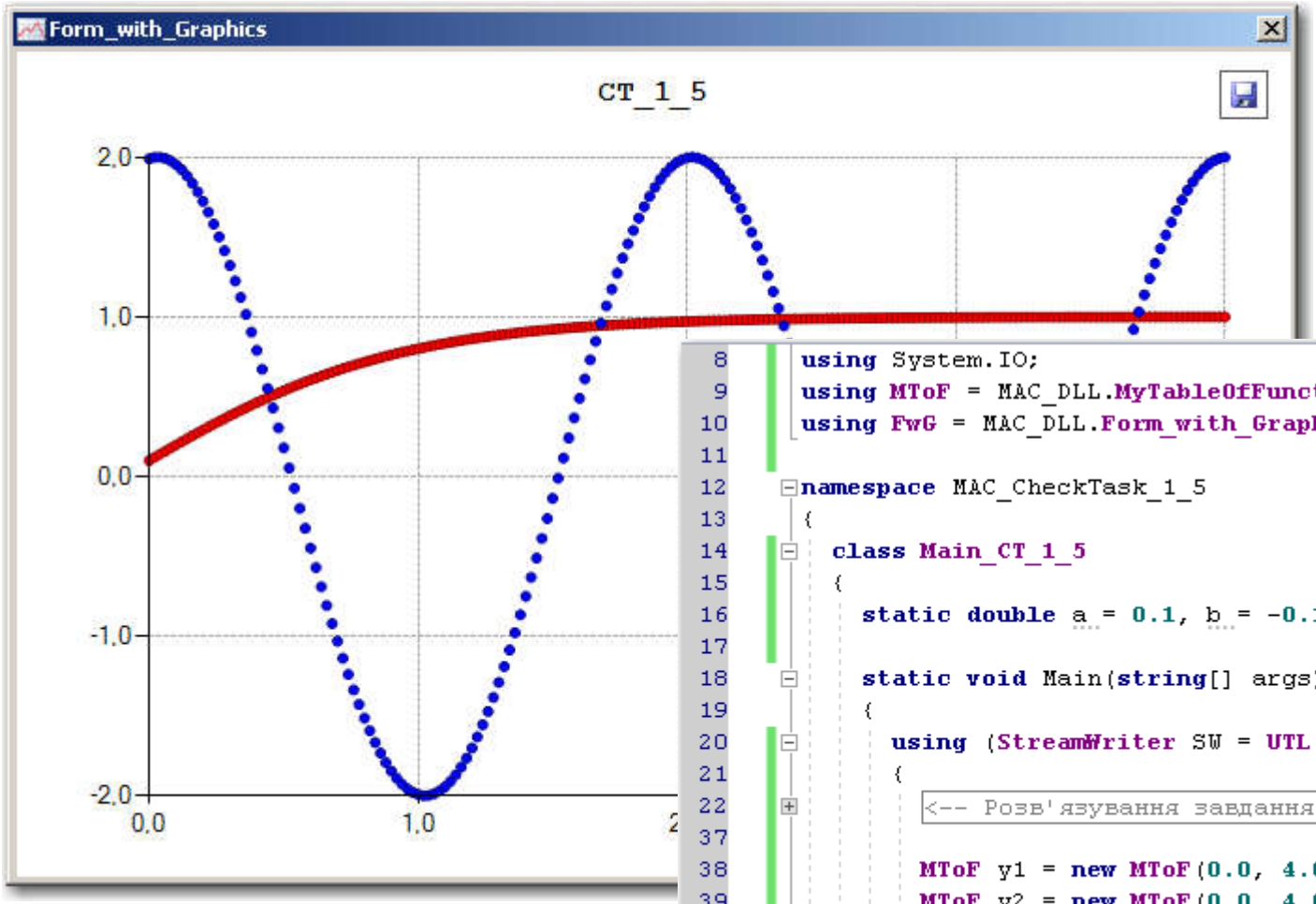
До даного самостійного завдання вам додаються коди нового конструктора **`Form_with_Data()`** та відповідного нового метода **`DoublyGraph()`**, які ви повинні додати до відповідних розділів класу **`Form_with_Graphics`** в бібліотеці **`MAC_DLL`**.

Зазначені компоненти розміщені в файлі **`to CT_1_5.rar`** на Гугл-диску в директорії «Програмні коди».



```
6   namespace MAC_DLL
7   {
8     public partial class Form_with_Graphics : Form
9     {
10      <--- Члени --->
15
16      #region <--- Конструктори --->
17
18      public Form_with_Graphics() { InitializeComponent(); }
19
20      private Form_with_Graphics(double[] x, double[] y, string title)...
51
52      private Form_with_Graphics(MyTable MTD)...
83
84      private Form_with_Graphics(MyTable f1, MyTable f2, string title)...
140
141     #endregion <--- Конструктори --->
142
143     #region    <--- Методи --->
144
145     public static void SingleGraphic(double[] X, double[] Y, string Title, int Nx, int Ny)...
153
154     public static void SingleGraphic(MyTable table, int Nx, int Ny)...
163
164     public static void DoublyGraph(MyTable g1, MyTable g2, string title, int Nx, int Ny)
165     {
166       if ((g1 == null) || (g2 == null)) return;
167       Form_with_Graphics fwd = new Form_with_Graphics(g1, g2, title);
168       fwd.filename = title;
169       fwd.StartPosition = FormStartPosition.Manual;
170       fwd.Location = new Point(Nx, Ny); fwd.ShowDialog();
171     }
172
173     #endregion <--- Методи --->
174
175     <--- Події форми --->
184   }
185 }
```

Крім того, не забудьте приєднати до переліку посилань **References** нашого проекту **MAC_CheckTask_1_5** системну бібліотеку **System.Windows.Forms**. Отже, незначні зміни в основній програмі дозволяють нам переглянути графіки функцій $y = f_1(x)$ та $y = f_2(x)$, та їх точки перетину, та зіставити їх із отриманими результатами обчислень.




```
8   using System.IO;
9   using MToF = MAC_DLL.MyTableOfFunction;
10  using FwG = MAC_DLL.Form_with_Graphics;
11
12  namespace MAC_CheckTask_1_5
13  {
14    class Main_CT_1_5
15    {
16      static double a = 0.1, b = -0.1, s = Math.Sqrt(10.0);
17
18      static void Main(string[] args)
19      {
20        using (StreamWriter SW = UTL.ResultWriter("CT_1_5"))
21        {
22          <-- Розв'язування завдання -->
37
38          MToF y1 = new MToF(0.0, 4.0, 200, f1, "f1");
39          MToF y2 = new MToF(0.0, 4.0, 200, f2, "f2");
40          FwG.DoublyGraph(y1, y2, "CT_1_5", 300, 500);
41        }
42      }
```

Якщо Ви успішно виконали дане програмне проектування і отримали представлені вище числові та графічні результати, то Ви можете приступати до безпосереднього виконання свого індивідуального варіанту даного самостійного завдання.

Для цього скористайтеся вже існуючим проектом **MAC_CheckTask_1_5**.

Нижче наведено код перевантаженого метода дотичних **Tangent()**, який належить класу **My_Equations**, та який є задіяним в методі **Roots_Correction()** в класі **MyExtendedTableOfFunction**:

```
 9   public class MAC_Equations
10   {
11     public static double Dichotomy(double a, double b, double eps, Func<double, double> f, out int K)...
23
24     public static void Dichotomy(Func<double, double> f, Root root, double eps)...
37
38     public static double Tangent(Func<double, double> Fx, Func<double, double> D1Fx, Func<double, double> D2Fx,
39                                  double xL, double xR, double eps, out int K)...
49
50     public static double Tangent(Func<double, double> Fx, double xL, double xR, double eps, out int K)...
60
61     public static void Tangent(Func<double, double> Fx, Func<double, double> D1Fx,
62                                Func<double, double> D2Fx, Root root, double eps)
63     {
64       root.X = (root.XR + root.XL) * 0.5; root.Iters = 0;
65       if ((Fx(root.XL) * D2Fx(root.XL)) > 0) root.X = root.XL;
66       if ((Fx(root.XR) * D2Fx(root.XR)) > 0) root.X = root.XR;
67       while (Math.Abs(Fx(root.X)) > eps)
68       {
69         root.X = root.X - Fx(root.X) / D1Fx(root.X); root.Iters++;
70         if (root.Iters > 15) break;
71       }
72       root.Err = Math.Abs(Fx(root.X));
73     }
74   }
```

Таблиця функцій та їх параметрів за варіантами

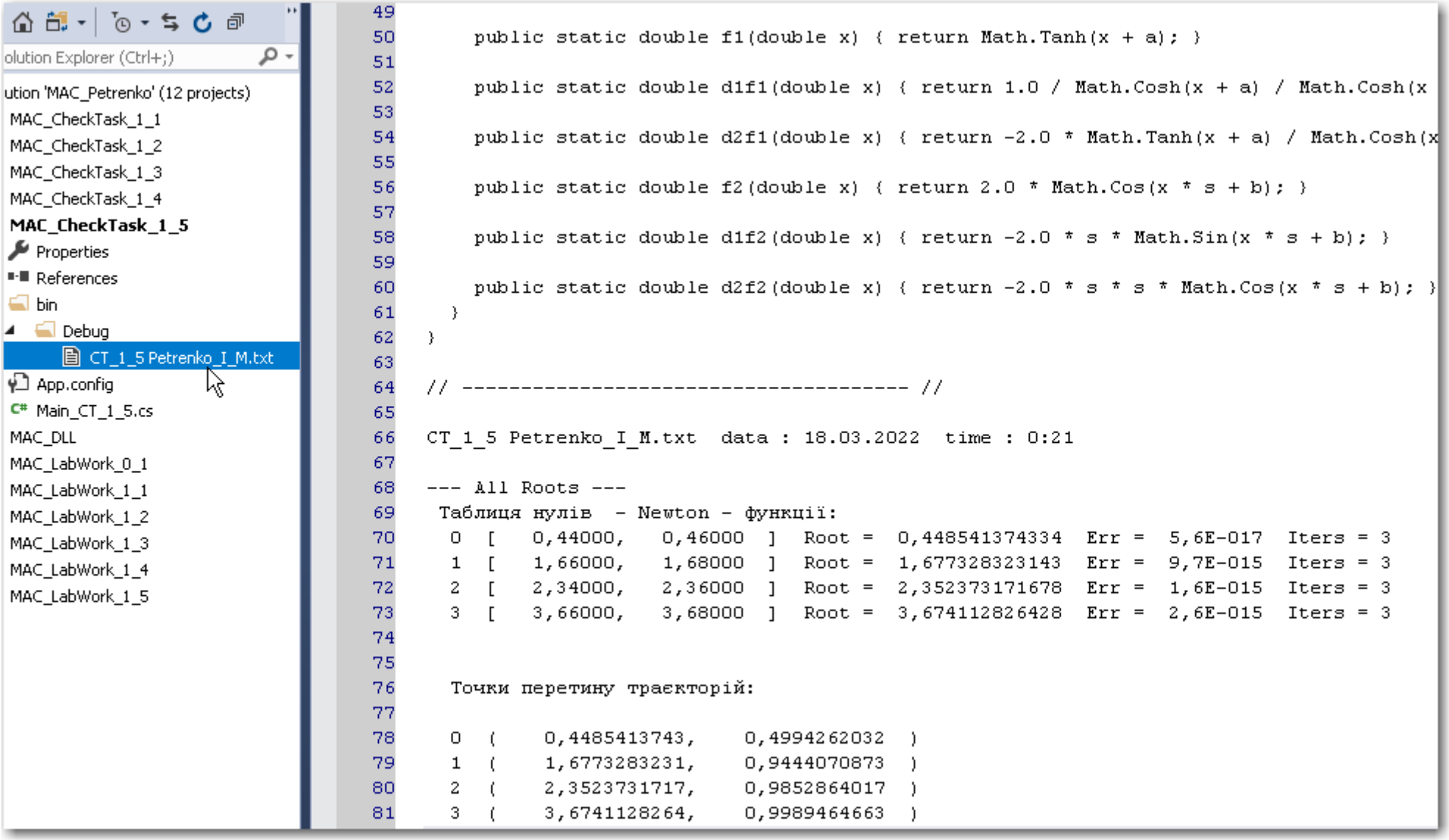
V		x_0	x_n	a	b	n	V.v
1	$f_1(x) = \lg\left(\sqrt{9x + a}\right)$ $f_2(x) = 4 \cdot \sin\left(5x + b\right)$	+2.0	+4.5	+7.0	-1.0	350	01.1
		+1.8	+4.2	+9.0	-1.5	330	01.2
		+2.4	+4.9	+8.5	-1.3	370	01.3
2	$f_1(x) = 4 \cdot \cos\left(\sqrt{17}x + a\right)$ $f_2(x) = -0.3 \cdot \sqrt{11x + b}$	+2.1	+5.3	+0.3	+0.1	290	02.1
		+2.7	+5.6	+0.5	-0.4	270	02.2
		+3.5	+6.8	+0.7	+0.3	380	02.3
3	$f_1(x) = 3\sqrt{2} \cdot \sin\left(\pi x + a\right)$ $f_2(x) = b \cdot \sqrt{\pi - 2x}$	-3.5	+0.5	+0.2	+1.0	280	03.1
		-2.5	+1.5	+0.5	+0.6	260	03.2
		-2.1	+0.9	+0.7	+1.3	340	03.3
4	$f_1(x) = a \cdot x^3 - 0.5$ $f_2(x) = \left(x + b\right)^2 \cdot \sin\left(4\pi x - 1\right)$	0.0	+1.0	+1.2	+1.0	270	04.1
		+0.4	+1.5	+1.1	+1.3	250	04.2
		+0.5	+1.4	+1.5	+1.8	350	04.3
5	$f_1(x) = 0.1 \cdot x^4 - a$ $f_2(x) = 2\sqrt{\pi} \cdot \cos\left(\pi x + b\right)$	-2.7	+1.3	-0.7	-1.4	260	05.1
		-2.8	+1.7	-0.6	-1.3	270	05.2
		-2.1	+1.4	-1.1	-1.8	320	05.3

V		x_0	x_n	a	b	n	V.v
6	$f_1(x) = e \cdot \sin(2\pi x + a)$ $f_2(x) = 0.25\pi \cdot (x - 1) - b \cdot \sqrt{x}$	+2.3	+4.2	+0.3	+1.0	310	06.1
		+2.1	+4.3	+0.5	+0.9	340	06.2
		+2.2	+4.1	+0.7	+0.6	370	06.3
7	$f_1(x) = 1.2 - \ln(x + a)$ $f_2(x) = 3 \cdot \cos(e \cdot x - b)$	+3.3	+8.5	-0.7	+0.2	280	07.1
		+4.4	+9.3	-0.4	+0.6	360	07.2
		+4.7	+8.8	-0.8	-0.5	310	07.3
8	$f_1(x) = \ln(\sqrt{x} + a)$ $f_2(x) = e \cdot \sin(4x - b)$	+3.1	+5.7	+3.2	+0.4	330	08.1
		+3.4	+6.5	+2.8	-0.7	290	08.2
		+4.2	+7.3	+3.1	+0.6	340	08.3
9	$f_1(x) = x^4 - 5x^2 + a$ $f_2(x) = \ln(0.5x + b)$	-2.8	+2.5	+3.1	+4.9	370	09.1
		-2.6	+2.7	+2.9	+4.8	340	09.2
		-2.3	+2.4	+4.3	+4.7	380	09.3
10	$f_1(x) = (x^2 + a)^{-1}$ $f_2(x) = 0.5 \cdot \sin(b \cdot x)$	-3.3	+2.4	+2.1	+2.5	350	10.1
		-1.1	+4.5	+2.8	+2.3	330	10.2
		+0.2	+4.3	+2.7	+2.4	420	10.3

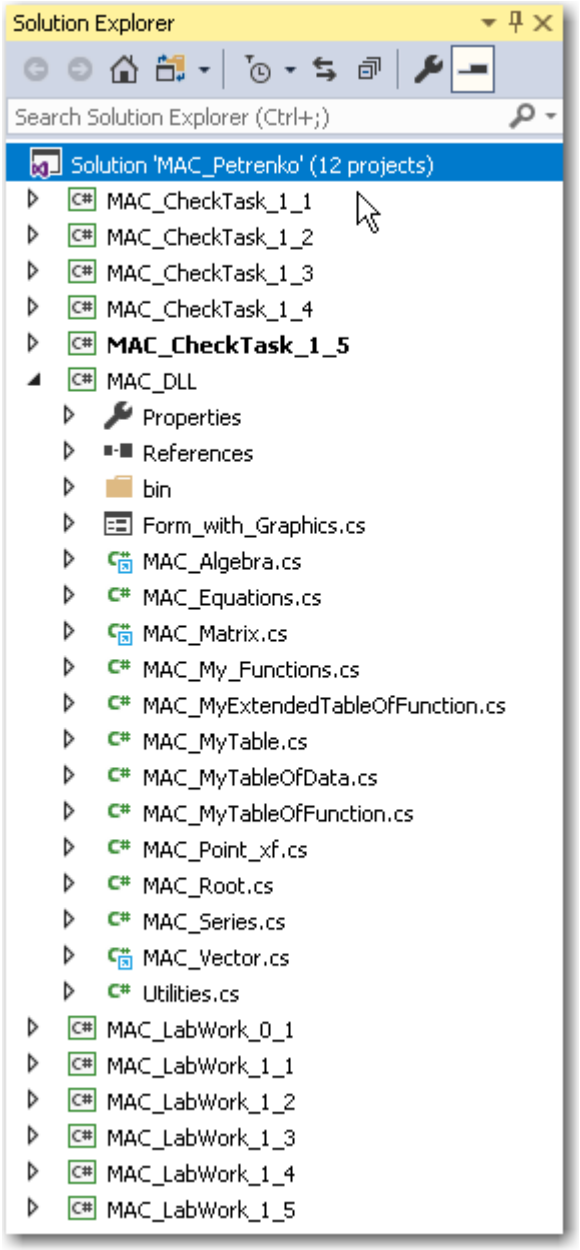
Таблиця тестових результатів за варіантами

V	x_0	x_n	a	b	n	$\tilde{x}_1; y(\tilde{x}_1)$	$\tilde{x}_2; y(\tilde{x}_2)$	$\tilde{x}_3; y(\tilde{x}_3)$	$\tilde{x}_4; y(\tilde{x}_4)$
1	+2.2	+4.8	+6.5	-0.5	300	+2.6505451982	+3.2026452687	+3.9106956916	+4.4562102464
						+0.7411144518	+0.7740337542	+0.8100485574	+0.8342204132
2	+2.0	+5.0	-0.2	+0.2	300	+2,0419234149	+2,6145619828	+3,5967532940	+4,1107028094
						-1,4281121037	-1,6144399533	-1,8917678930	-2,0217803494
3	-5.0	-0.5	-0.2	+0.8	300	-3,7260073280	-3,1326335892	-1,7741691859	-1,0792742831
						+2,6038257772	+2,4536483636	+2,0691920782	+1,8417628459
4	-0.1	+0.7	+0.9	+0.8	300	-0,0721016445	+0,0108089159	+0,3574025644	+0,5650801538
						-0,5003373479	-0,4999988634	-0,4589119527	-0,3376049926
5	-3.0	+1.0	-0.5	-1.0	300	-1,9850961597	-1,2489334576	-0,1366399574	+0,7700709903
						+2,0528381662	+0,7433084546	+0,5000348587	+0,5351660066
6	+1.8	+4.0	-0.3	+1.2	300	+1,9931607082	+2,5881803904	+3,0183271780	+3,5625426956
						-0,9141255280	-0,6831847615	-0,4996095643	-0,2523475762
7	+2.0	+6.5	-0.4	-0.4	300	+2,6968701867	+3,8917141678	+5,0962387513	+6,1417982838
						+0,3684525922	-0,0503927812	-0,3467619227	-0,5477724511
8	+2.0	+5.0	+3.5	-0.2	300	+2,1485678080	+3,2568212074	+3,7091186224	+4,8360865357
						+1,6025742866	+1,6685869065	+1,6911851151	+1,7403100919
9	-2.6	+2.6	+3.0	+5.0	300	-2,1560672245	-0,5555073769	+0,5327707910	+2,1791474557
						+1,3665931576	+1,5522845998	+1,6613442440	+1,8065780838
10	-2.2	+3.6	+2.5	+2.0	300	-1,7541831560	+0,4216776169	+1,3264024520	+3,2196331364
						+0,1793027743	+0,3734392091	+0,2347779671	+0,0777240077

Зразок файлу результатів, який надсилається на перевірку:



По завершенню всіх залікових робіт першого Модуля даної навчальної дисципліни Ваш робочий простір **MAC_Petrenko** повинен мати наступну загальну структуру проектів:



Навчальне видання

ЧИСЕЛЬНІ МЕТОДИ

Модуль 1. Елементарні обчислення

ЕЛЕКТРОННИЙ МЕТОДИЧНИЙ ПОСІБНИК

до лабораторних робіт студентам факультету математики, фізики та інформаційних технологій
першого (бакалаврського) рівня освіти, спеціальності 122 «Комп'ютерні науки», 126 «Інформаційні системи і технології»

Електронне практичне видання

Укладачі:
Царенко Олексій Павлович
Недєва Ольга Анатоліївна

В авторській редакції

Затвердж. авт. 18.09.2023. Шрифт Times New Roman.
Системні вимоги: операційна система сумісна з програмним забезпеченням для читання файлів формату PDF.
Обсяг 8,3 МБ. Зам. № 2653.

Видавець і виготовлювач
Одеський національний університет імені І. І. Мечникова
Свідоцтво суб'єкта видавничої справи ДК № 4215 від 22.11.2011 р.
65082, м. Одеса, вул. Єлісаветинська, 12, Україна, тел.: (048) 723 28 39, e-mail: druk@onu.edu.ua