**Дніпровський національний університет імені Олеся Гончара**

**ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ**

**КАФЕДРА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ**

**Звіт**

**з навчальної практики: комп’ютерно-технічної**

на тему:«Візуальне програмування в середовищі MicrosoftVisualStudio»

Студентки I курсу ПА-19-2 групи

Мовсісян Лаури Ростомівни

спеціальності: Прикладна математика

керівник: ас. Сірик Світлана Федорівна

Кількість балів\_\_\_\_\_\_\_\_\_\_\_\_

Національна шкала \_\_\_\_\_\_\_\_

Оцінка ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Гук Н.А\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_СафроноваІ.А.\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Сердюк М.Є.\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

м. Дніпро – 2020р.

**Зміст**

Вступ………………………………………………………………………….……… 3

Постановка задачі…………………………………………………………………… 4

Підготовка середовища програмування…………………………………………… 5

1. Початок роботи…………………………………………………………………....6

2. Макетування……………………………………………….………………………7

3. Реалізація графічного інтерфейсу програми…………………………………….8

3.1.Панель інтервалу…………………………………………………………...…...14

3.2. Панель вибору функції…..……………………………………….…….……...14

3.3. Панель вибору………...……………………………………………………..…14

3.4. Графік………………………………………………………………….……..…15

3.5. Кнопка початку роботи програми………………………………………...…. 15

4. Етапи розробки графіків..………………………………………………………..16

4.1.Додання декількох графікі……………………………………………………..14

4.2.Стрілки на осях ………………..………………………………………… ……14

4.3. Початок системи координат……………………………………………..….…17

4.4. Розмітка головних осей…………………………………………………….…..17

4.5.Назва координатних осей…………………………………………………….. .17

4.6. Стиснення та розтягнення……………………………………………………..17

4.7.Маштабування …….………………….………………………………………...17

5. Додаткові форми…………………………………………………………………18

5.1.Інформація про програму………………………………………………………18

5.2.Формування файла………………………………………...……………………18

5.3.Індивідуальне завдання…………………………………………………………19

Висновки…………………………………………………………………………….20

Список використаних джерел………………………………..…………………….21

Додатки…………………………………………………...…………………………22

**Вступ**

З 15 по 26 червня 2020-го року, я, студентка групи ПА-19-2 Мовсісян Лаура, пройшла навчальну практику з теми «Візуальне програмування в середовищі Microsoft Visual Studio» під керівництвом Сірик Світлани Федорівни.

Під час практики я повинна була навчитися працювати з візуальним інтерфейсом програми та зафіксувати знання, які я отримала за 1 курс навчання. Нашою метою була розробка дизайну для цієї програми, її програмування та тестування роботи програми.

У якості завдання нашій групі дали реалізувати побудову графіків функції. Більш уваги на завдання ми надамо у пункті «Постановка задачі».

В ході роботи кожен мав змогу обирати для свого проекта комфортну для нього мову програмування.Я обрала мову С#, оскільки завдяки неї, легше реалізувати візуальний інтерфейс програми без зайвих налаштувань.

Кожен день,під час проходження практики, ми вели письмовий звіт у форматі щоденника , в якому містилися записи:

-хід роботи над програмою;

-що було зроблено за день;

-які труднощі виникли.

# **Постановка задачи**

Зміст роботи:

1. Вивчити базові можливості візуального програмування [1].

2. Вивчити базові можливості програмування графіки [2,4,3,5].

3. Написати програму для виведення на екран графіка функції, а також програму для підготовки даних (номер варіанта Індивідуального завдання збігається з номером в студентському журналі)

4. Підготувати звіт про виконання завдання для навчальної практики: комп'ютерно – технологічної.

Технічні вимоги:

1. вид графіка

2. осі (розмітка осей, написи на осях, стрілочки)

3. можливість виведення декількох графіків

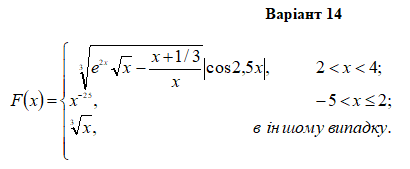
4. вивести графік функції, заданої в текстовому файлі поточено, тобто значеннями функції в точках).

5. масштабування

6. стиснення / розтягнення щодо осі абсцис / ординат.

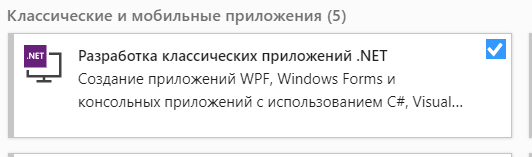
Примітка: функціональні можливості повинні бути реалізовані в тексті програми (засобами мови) і в інтерфейсі користувача (засобами візуальних компонент)

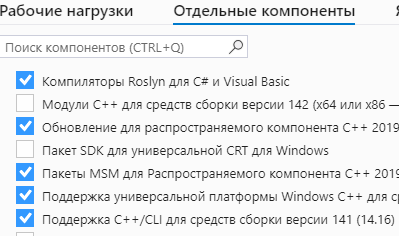
Індивідуальне завдання:

****

# **Підготовка середовища програмування**

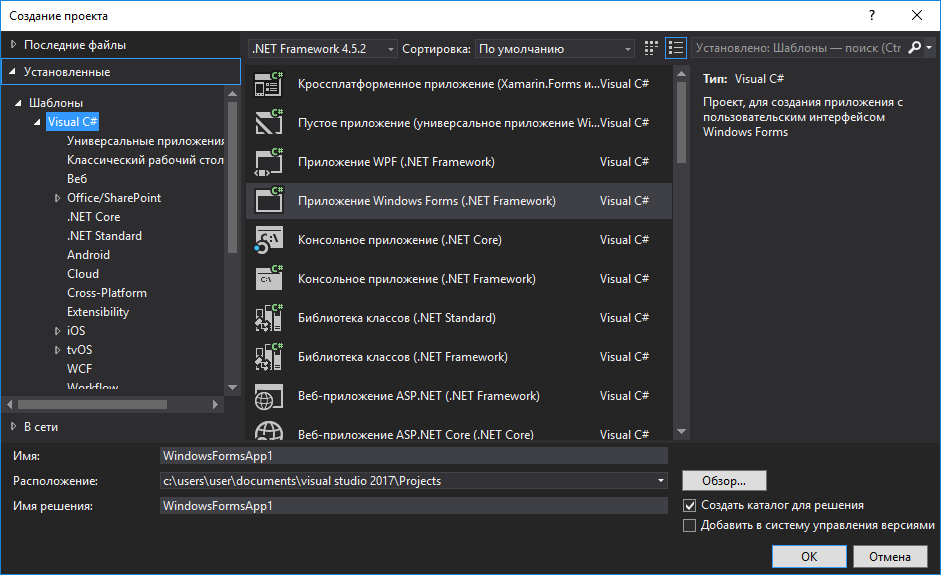
Перед тим, як починати виконувати завдання в середовищі Visual Studio 2019, я мала доповнити підтримку мови програмування С++/CLI. Для цього я запустила VS Inctaller та завантажила додаткову окрему підтримку та відповідні розширення :





1. **Початок роботи**

Спершу під час роботи з графічним інтерфейсом в MicrosoftVisualStudio 2017 необхідно в створенні проекту обрати «Додаток WindowsForm (.NETFramework). Для зручності можна відсортувати можливі для створення проекти по мовам, в моєму випадку це C#. Даємо назву проекту та розпочинаємо роботу.

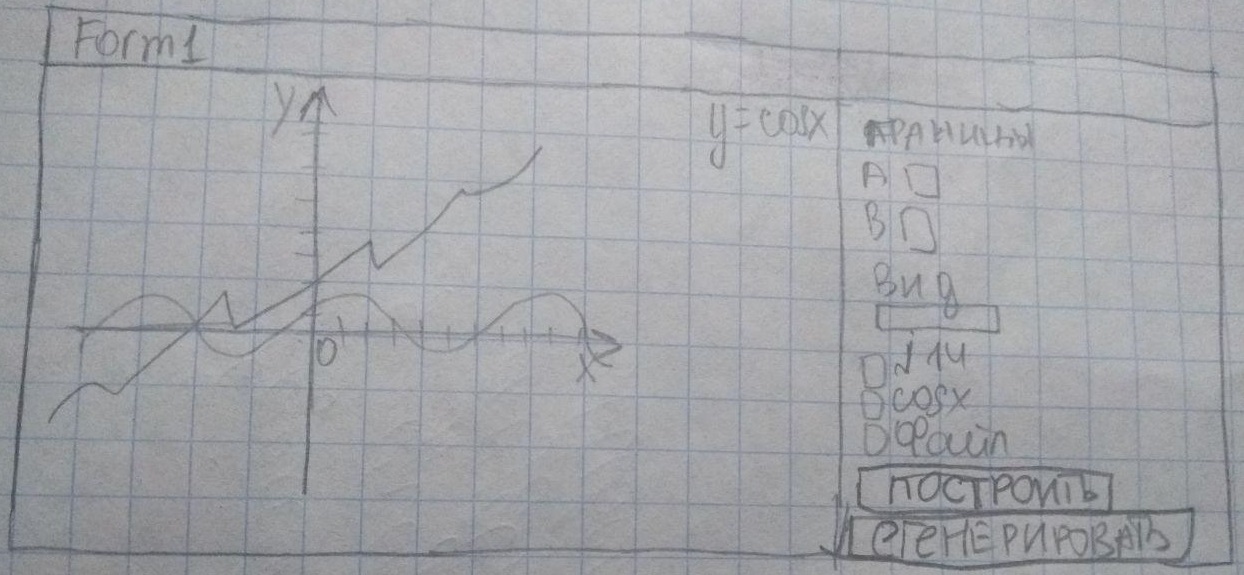
*Створення проекту в середовищі MicrosoftVisualStudio 2017 рис.1.1*

# 

# **2. Макетування**

**Малюємо приблизний макет інтерфейсу майбутньої програми. Він має такий вигляд:**

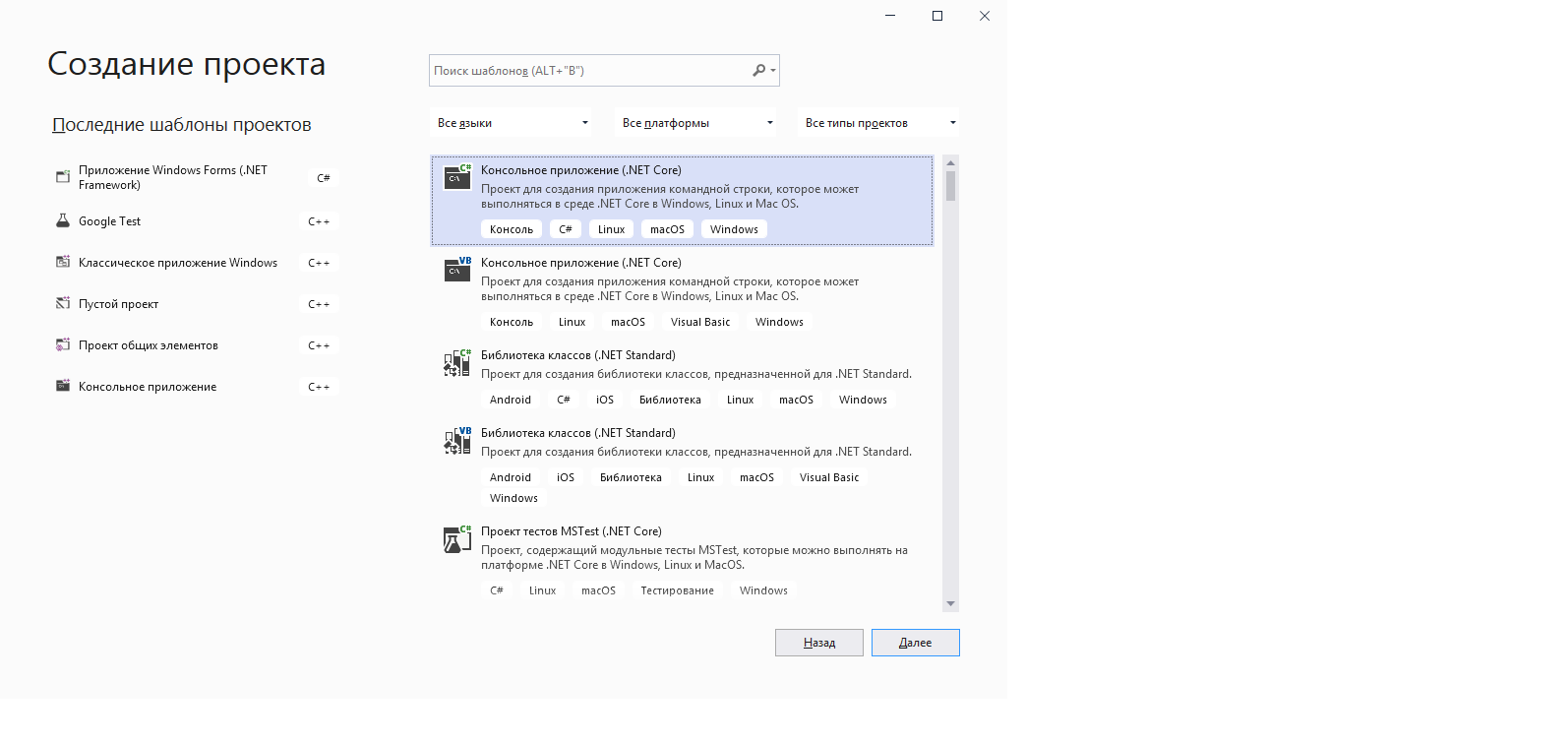
***Макет інтерфейса рис. 2.1***



На макеті зображені усі основні компоненти, які буде містити наша програма . Подалі у неї будуть внесені деякі зміни, тому вона не повністю співпадатиме з макетом, але загальна картина залишатиметься такою ж і зміни не вплинуть на фундамент макета програми.

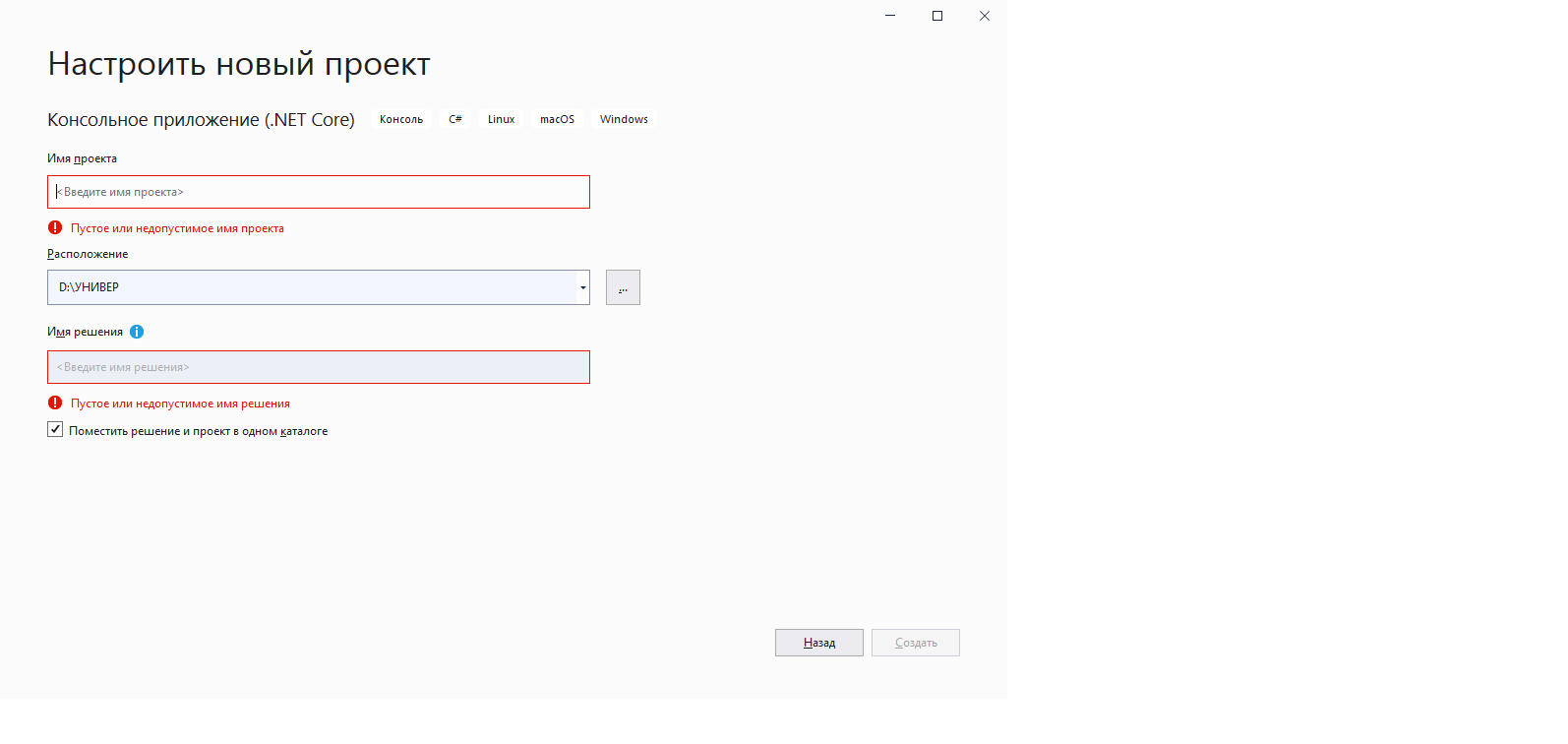
**3.** **Реалізація графічного інтерфейсу програми**

Запускаємо VS та створюємо нову Windows Forms (C#):



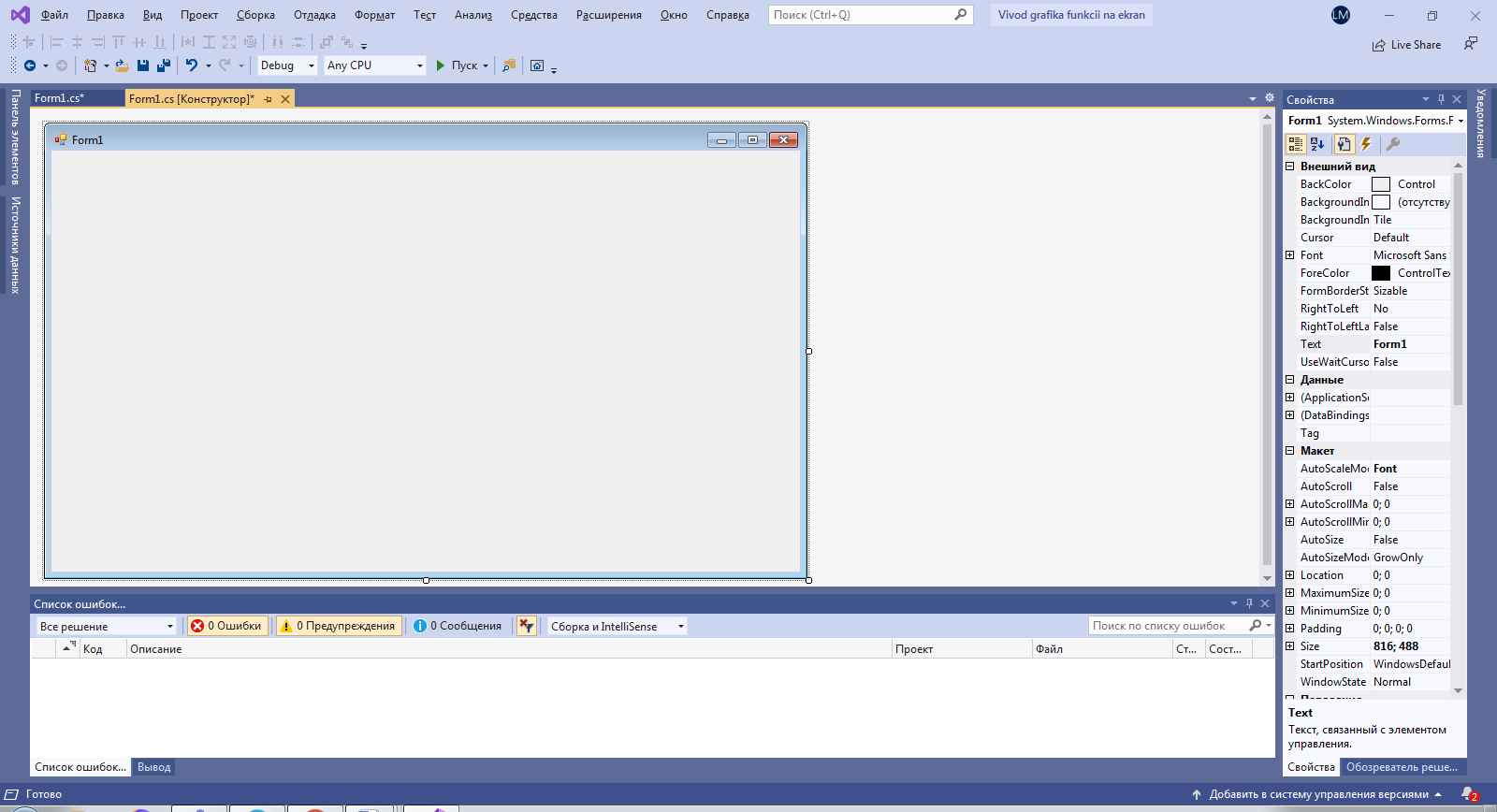
*Створення проекту*

Даємо назву проекту:



*Назва проекту*

Тиснемо «Создать» і отримуємо порожній проект та форму:

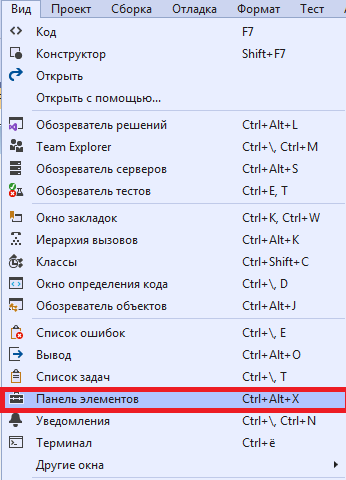


*Пустий проект*

Проект готовий для подальшого наповнення змістом.

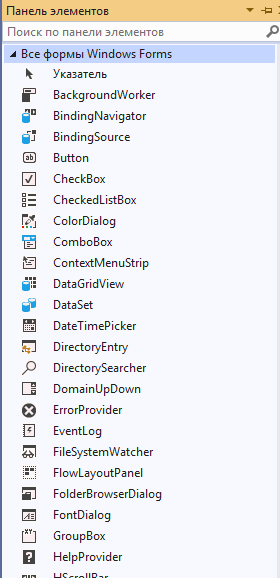
Після того,створення проекту, додаємо до нього елементи зToolbox. Для її пошуку, треба знайти верхню позначку «Вид»,після натиснути на «Панель элементов».

*«Панель элементов» рис 3.1*

**

В результаті ми отримуємо вікно, в якому лежать необхідні інструменти:

*Вікно з інструментами для роботи з графічним інтерфейсом рис.3.2*

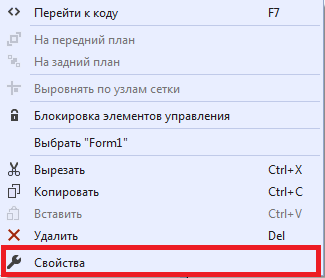


У роботі використано елементи типу:

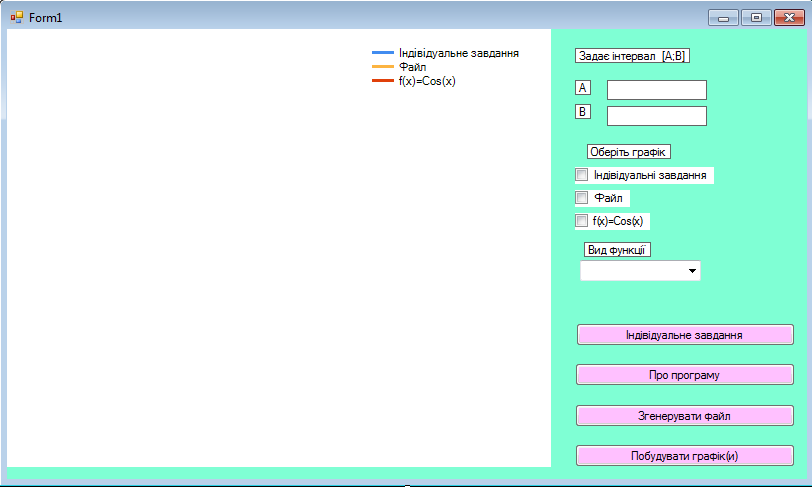
* Chart –побудова вікна з графіком.
* Bottom – побудова кнопок.
* CheckBox – вибір графіків для виведення на екран.
* ComboBox – вибір одного з варіантів виводу графіка.
* Label – написи.
* TextBox – зміна тексту.
* PictureBox– вставка фото.

Для редагування властивості елемента, наприклад, зміни напису на кнопці, треба натиснути на той елемент лівою клавішею миші та натиснути на «Свойства», після з’явиться вікно. В ньому є можливість редагування властивостей елемента. Більше уваги редагуванню властивостей ми надамо нижче.

*Властивості елемента. рис.3.3*



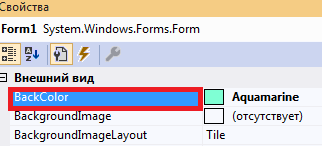
Після усіх маніпуляцій, кінцевий інтерфейс виглядає так:

Готовий інтерфейс рис.3.4

Отже відповідно макету,я приєднала кнопки, додала для зручності користувача написи, які підказують, що потрібно робити, обновили написи на кнопках та фон програми. Більше про елементи побачимо далі.

Для зміни фону програми, у властивостях знаходимо елементи,що відповідають за зовнішній вигляд програми, і змінюємо фон у BackgroundImage, на один з кольорів на вибір.

*Зміна фону рис.3.5*

**

**3.1. Панель інтервалу**

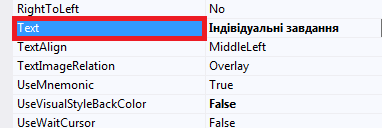
Для того, щоб користувач мав змогу вводити інтервал, я використовувала у своїй роботі такі елементи, як Label та Text Box. У вікні Label, ми написали користувачеві, що він має вводити в Text Box.

Якщо користувач неправильно введе границі інтервалу [А,B],таким чином, що,В більше за А, то програма поміняє їх місцями.

### **3.2.****Панель вибору функції**

Для реалізації вибору функції я додала три елементи Check Box, кожний із назвою необхідного графіка. Програма почне малювати графіки, залежно від того які з компонентів будуть відмічені користувачем, після чого виведе на екран відповідні графіки.

*Зміна назви CheckBox рис.3.6*



### **3.3. Панель вибору**

Для реалізація вибору типу графіка використовуємо ComboBox. Користувач має обрати один із запропонованих варіантів, після чого програма змінює вид графіка. В іншому випадку автоматично програма малює лінійний графік.

### 

### **3.4.** **Графік**

Для реалізація графіків використовуємо елемент Chart. Має бути можливість відображення одночасно декількох графіків. Для кожного графіка обираємо окремий колір, щоб швидше розрізняти, після додаємо розмітку осей, написи на них та направляючі стрілки. Прибираємо сітку. Важливо задати початок системи координат в точці (0.0).

Дозволяємо користувачеві масштабувати графік за власним бажанням.

Більш уваги звернемо на це у пункті «Етапи розробки графіків».

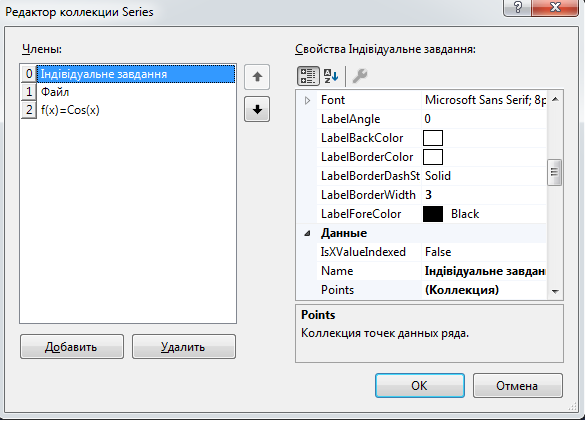
### **3.5.****Кнопка початку роботи програми**

Для реалізації кнопки запуску програми ми використовували елемент Button. Ми перейшли до коду програми двічі натиснувши на елемент. Коли користувач натискає кнопку перевіряються помилкові ситуації.Якщо користувач ввів усі параметри, то програма запускається і будує відповідний графік вказаного типу на вказаному інтервалі.

**4. Етапи розробки графіків:**

**4.1.** **Додання декількох графіків**

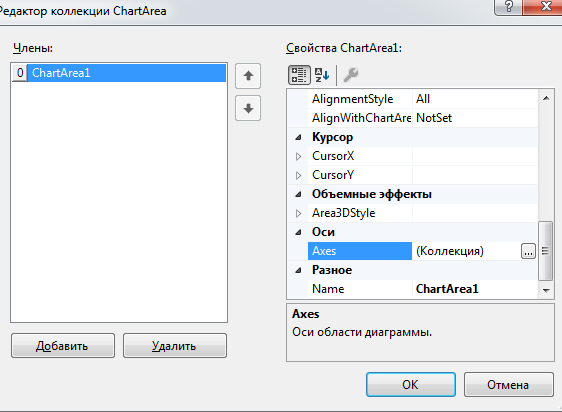
Для того, щоб одночасно намалювати декілька графіків, у властивостях елементу Chart знаходимо колекцію Series, натискаємо на кнопку «Добавить»,додаємо графіки та налаштовуємо

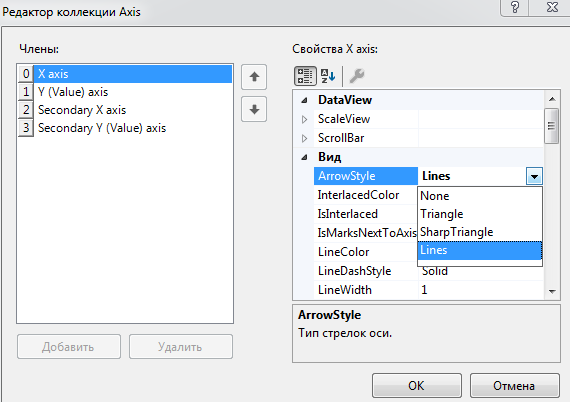
Додавання декількох функцій рис.4.1

**4.2. Стрілки на осях**

В колекції ChartAreas знаходимо колекцію Axes.Вона відповідає за властивості координат осей. Для відображення стрілочок на координатних осях міняємо властивість ArrowStyle на Lines для осі і X ,і Y.

*Осі Х та У Рис.4.2, рис 4.3*

****



**4.3. Початок системи координат**

Для створення початку координат у властивості Scale змінемо властивість Crossing з Auto на 0 по координаті і Х ,і У.

**4.4. Розмітка головних осей**

Для відображення розміток на координатних осях змінимо властивість IsMarginVisible на True по осі Х та У.

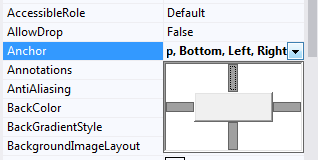
**4.5. Назва координатних осей**

Для додання написів на координатних осях додаємо назву осей в властивості Title для кожної осі окремо.

**4.6. Стиснення та розтягнення**

Для стиснення та розтягнення графіків використовуємо властивість Anchor для кожного компонента, та налаштовуємо її в залежності від того, де знаходиться компонент і як він має розтягуватись:

*Зміна Аnchorрис.4.4*



**4.7. Маштабування**

Для масштабування в колекції ChartAreas потрібно змінити властивість IsUserEnabled та IsUserSelection на true для Cursorx та Cursory.

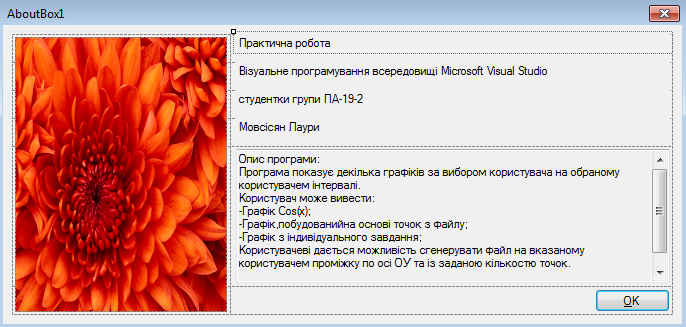
**5. Додаткові форми**

В процесі проектування я додала до основної форми ще 2 форми:

**5.1.Інформація про програму**

При натисненні на кнопку «Опис програми» програма виводить на екран ще одну форму, що містить інформацію про програму та її автора:

*Інформація про програму рис.5.1*



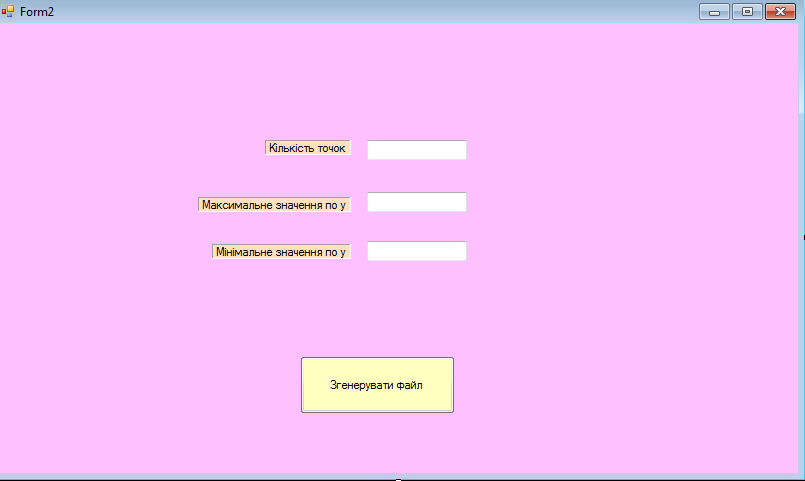
Я використали PictureBox щоб,урізноманітнити та вставила фото квітки

При натисненні кнопки «ОК» вікно з описом закриється.

**5.2.Формування файла**

При виборі графіка побудованого на основі файлу, користувач також має згенерувати файл, натиснувши на кнопку «Згенерувати файл». В формі,яка з'явиться користувач повинен заповнити інформацію для подальшої роботи, а саме кількість точок та максимальне і мінімальне значення, в якому будуть генеруватися точки по осі ОУ.

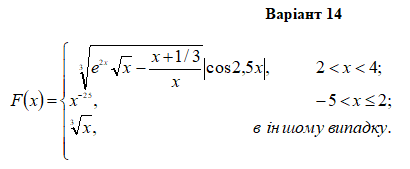
Форма для генерації файла рис.5.2



Після натиснення кнопки «Згенерувати файл»  з’явиться ця форма, яку користувач повинен заповнити. Якщо якесь з полів буде пустим, то програма видасть відповідне повідомлення. Якщо дані введені некоректно (кількість точок < 0 або нижня границя більша за верхню) то програма змінить дані на коректні сама (від’ємну кількість змінить на додатню або змінить місцями).

Потім програма згенерує файл test.txt, в якому будуть знаходитися координати точок, по яким буде побудований графік.

**Приклад змісту файла для 5 елементів, min = -4, max = 6. Інтервал : [-4,6]:**

******

# **Висновки**

Під час виконання роботи по навчальній практиці ми навчилися:

-базовим можливостям візуального програмування;

-можливості програмування графіка.

Ознайомилися з базовими елементами їх властивостями. Під час проходження практики, я ознайомилася та получила практичні навички в програмуванні на С#.

Навчилися додавати до проекту декілька форм та виводити їх на екран.

Розглянули елемент Chart та його базові можливості, а саме:

-масштабування,

-редагування осей графіка,

-додавання декількох графіків одночасно.

Використовували його складові для реалізації нашого проекту. Реалізували вивід декількох графіків одночасно та окремо, завдяки СhackBox.Ми навчилися створювати простий дизайн для програм.

-розвили навички командної праці.

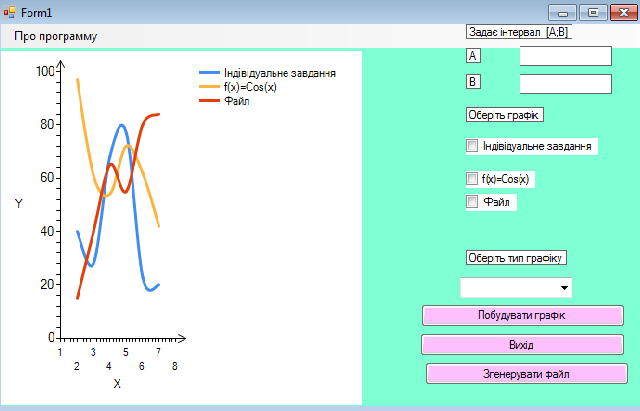
Робота з візуальним інтерфейсом була дуже цікавою.Я здобула новий досвід в програмуванні, який неодмінно стану у пригоді в подальшому навчанні.

**Список використаних джерел**

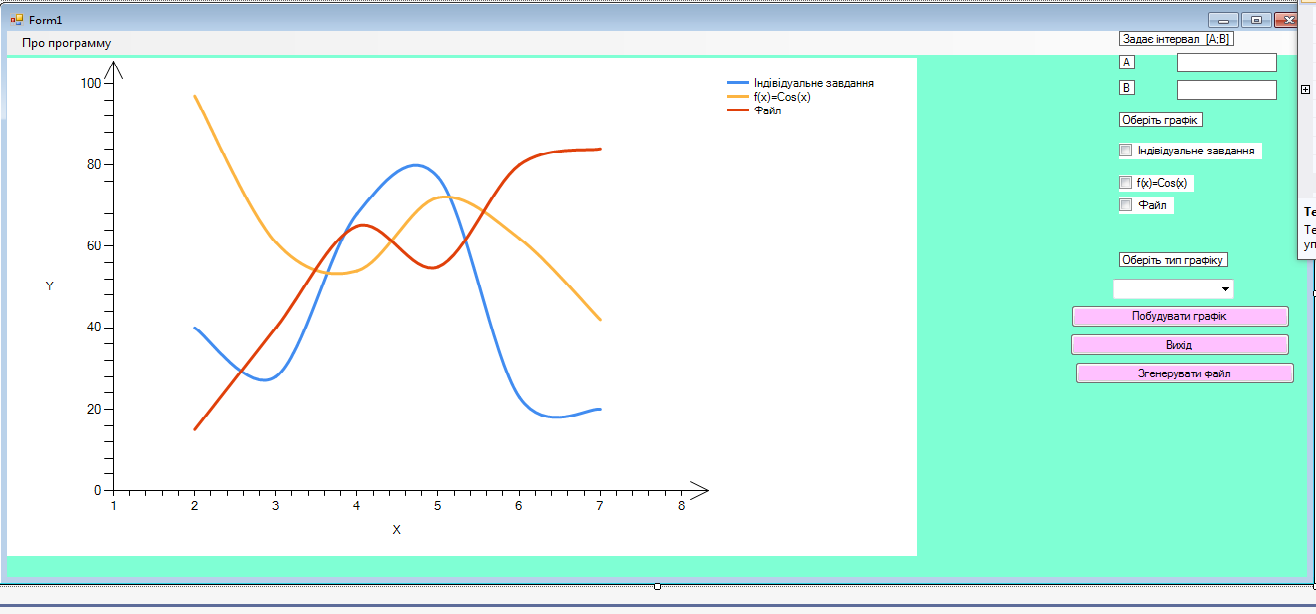
1. Борис Пахомов. C/C++ и MS Visual C++ 2010 для начинающих. - БХВ-Петербург, 2011. - 736 с.
2. КлассGraphics[Електронний ресурс].–Режим доступу:http://msdn.microsoft.com/ru-ru/library/system.drawing.graphics.aspx
3. Работа с графикой в платформе .NET Framework на языке C++[Електронний ресурс ]. – Режим доступу:http://msdn.microsoft.com/ru-ru/library/
4. Построение графика функции на промежутке с определенным шагом [Електронний ресурс]. – Режим доступу: <https://otherreferats.allbest.ru/programming/00423204_0.html>
5. http://msdn.microsoft.com/ru-ru/library/system.drawing.graphics.aspx

# **Додатки**

*СтисненняДодаток 8*

**

*Розтягнення Додаток 9*

**

Код Основной программы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace Temp

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Button1\_Click(object sender, EventArgs e)

{

Cursor = Cursors.WaitCursor;

try

{

var inputs = new List<string>();

foreach (var textBox in Controls.OfType<TextBox>())

{

inputs.Add(textBox.Text);

}

// проверка заполнения полей

if (inputs.Any(string.IsNullOrEmpty))

{

MessageBox.Show("Не всі поля заповнені");

}

else

{

var a = Convert.ToDouble(textBox1.Text);

var b = Convert.ToDouble(textBox2.Text);

// проверка нижней и верхней границы

if (a > b)

{

var tmp = a;

a = b;

b = tmp;

}

// підключення усіх графіків

Series graphic1 = chart1.Series["Вариант№14"];

Series graphic2 = chart1.Series["y = log(x)"];

Series graphic3 = chart1.Series["from file"];

// очистка

graphic1.Points.Clear();

graphic2.Points.Clear();

graphic3.Points.Clear();

if (checkBox1.Checked)

{

for (var i = a; i <= b; i += 0.1)

{

var x = i;

var y = Function(x);

graphic1.Points.AddXY(x, y);

}

}

if (checkBox2.Checked)

{

for (var i = a; i <= b; i += 0.1)

{

var x = i;

var y = Log(x);

graphic2.Points.AddXY(x, y);

}

}

if (checkBox3.Checked)

{

// считываем инф из файла в виде массива строк

var fileContent = File.ReadAllLines(@"c:/temp/test.txt");

// разделяем стоку х и у

var stringOfX = fileContent[0];

var stringOfY = fileContent[1];

// разделяем каждую строку через пробелы

var listOfX = stringOfX.Split(' ');

var listOfY = stringOfY.Split(' ');

// преобразование каждого х и у в double

for (var i = 0; i < listOfX.Length; i++)

{

var x = Convert.ToDouble(listOfX[i]);

var y = Convert.ToDouble(listOfY[i]);

graphic3.Points.AddXY(x, y);

}

}

}

}

catch

{

}

Cursor = Cursors.Default;

}

// індивідуальне завдання

public double Function(double x)

{

try

{

if (x == 0.0)

{

return 0;

}

else if (x > -3 && x <= 2)

{

return Math.Pow(x, -2);

}

else if (x > 5 && x < 10)

{

return Math.Pow(Math.Exp(-2 + x), 1 / 5) \* (1 / Math.Sqrt(Math.Pow(x, 2) + x + Math.Log(Math.Abs(x - 3.14))));

}

else

{

return Math.Pow(x, 2);

}

}

catch

{

return 0;

}

}

public double Log(double x)

{

return Math.Log(x);

}

private void ComboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

// преобразование объекта типа object к типу combobox

var comboBox = (ComboBox)sender;

// достать выбраный елемент

var selectedType = comboBox.SelectedItem.ToString();

// пройтись по всем графикам

foreach (var series in chart1.Series)

{

switch (selectedType)

{

case "Line":

{

series.ChartType = SeriesChartType.Line;

break;

}

case "Point":

{

series.ChartType = SeriesChartType.Point;

break;

}

}

}

}

private void Button2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

// подсоединяем кнопку генерации файла с кодом для файла

private void Button3\_Click(object sender, EventArgs e)

{

try

{

var a = Convert.ToDouble(textBox1.Text);

var b = Convert.ToDouble(textBox2.Text);

var childForm = new Form2

{

A = a,

B = b

};

childForm.Show();

}

catch

{

}

}

// подсоединяем кнопку "Про програму" з кодом

private void ПроПрограмуToolStripMenuItem\_Click(object sender, EventArgs e)

{

var about = new AboutProgramm();

about.Show();

}

}

}

Код (О программе)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Temp

{

partial class AboutProgramm : Form

{

public AboutProgramm()

{

InitializeComponent();

this.Text = String.Format("О программе {0}", AssemblyTitle);

this.labelProductName.Text = AssemblyProduct;

this.labelVersion.Text = String.Format("Версия {0}", AssemblyVersion);

this.labelCopyright.Text = AssemblyCopyright;

this.labelCompanyName.Text = AssemblyCompany;

this.textBoxDescription.Text = AssemblyDescription;

}

#region Методы доступа к атрибутам сборки

public string AssemblyTitle

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttribute), false);

if (attributes.Length > 0)

{

AssemblyTitleAttribute titleAttribute = (AssemblyTitleAttribute)attributes[0];

if (titleAttribute.Title != "")

{

return titleAttribute.Title;

}

}

return System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().CodeBase);

}

}

public string AssemblyVersion

{

get

{

return Assembly.GetExecutingAssembly().GetName().Version.ToString();

}

}

public string AssemblyDescription

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescriptionAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyDescriptionAttribute)attributes[0]).Description;

}

}

public string AssemblyProduct

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyProductAttribute)attributes[0]).Product;

}

}

public string AssemblyCopyright

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;

}

}

public string AssemblyCompany

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCompanyAttribute)attributes[0]).Company;

}

}

#endregion

private void OkButton\_Click(object sender, EventArgs e)

{

Close();

}

}

}

код генерации файла

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Temp

{

public partial class Form2 : Form

{

public double A;

public double B;

public Form2()

{

InitializeComponent();

}

private void Button1\_Click(object sender, EventArgs e)

{

// взводимсчетчик

var random = new Random();//srand(time(0))

var inputs = new List<string>();

// собираемвсе inputs

foreach (var textBox in Controls.OfType<TextBox>())

{

inputs.Add(textBox.Text);

}

// проверяем на заполненость

if (inputs.Any(string.IsNullOrEmpty))

{

MessageBox.Show("Не всі поля заповнені");

}

else

{

var countOfPoint = Convert.ToInt32(textBox1.Text);

var lowerBorder = Convert.ToInt32(textBox2.Text);

var upperBorder = Convert.ToInt32(textBox3.Text);

if (countOfPoint < 0)

{

countOfPoint = Math.Abs(countOfPoint);

}

if (lowerBorder > upperBorder)

{

var tmp = lowerBorder;

lowerBorder = upperBorder;

upperBorder = tmp;

}

// создать пустые списки чисел для х и у

var listOfX = new List<double>();

var listOfY = new List<int>();

// считаем шаг

var step = (B - A) / (countOfPoint - 1);

// перемещаемся от нижнего значения до верхнего с шагом

for (var i = A; i <= B; i += step)

{

listOfX.Add(i);

// у - рандомное число, сгенерированное на заданом диапазоне

var y = lowerBorder + RandomNumber(random, 0, (upperBorder - lowerBorder));

listOfY.Add(y);

}

// удаляем старый файл

File.Delete(@"c:/temp/test.txt");

// создаем новый

var file = File.CreateText(@"c:/temp/test.txt");

// список х и у, разделенных пробелами

file.WriteLine(string.Join(" ", listOfX));

file.WriteLine(string.Join(" ", listOfY));

// закрыть файл для записи

file.Close();

Close();

}

}

public int RandomNumber(Random random, int min, int max)

{

// рандомное число в заданом диапазоне

return random.Next(min, max);

}

}

}