

Лабораторна робота №3. Windows Forms. Перетворення типів даних

Тема: Перетворення типів даних.

Мета заняття: навчитись працювати з елементом TextBox у середовищі Microsoft Visual Studio; набути практику перетворення типів даних для введення даних користувачем, їх обробки та виводу результатів з допомогою Windows Forms мовою C#; навчитись працювати з типами даних string, Array, ArrayList.

1. Теоретичні відомості

1.1 Елемент TextBox Windows Forms

C # TextBox використовується для отримання даних від користувачів або також може бути використаний для відображення деяких значень для користувача. Текстове поле є контейнером для текстових блоків, ви можете робити введення або показувати текст, як потрібно, у формі абзаців.

Перш за все, вам потрібно перетягнути TextBox з панелі інструментів і відрегулювати його у формі, як ви хотіли. Назва текстового поля за замовчуванням – «textBox1», яку можна змінити на панелі на вкладці властивостей. Також можна змінити властивість Text всередині текстового поля на панелі властивостей, але якщо потрібно змінити текст динамічно, тоді потрібно ввести код, як показано нижче.

```
textBox1.Text = "TheEngineeringProjects.com";
```

Тепер, коли будете виконуватись код, текстове поле з текстом з'явиться автоматично, як показано на рис. 1.1.

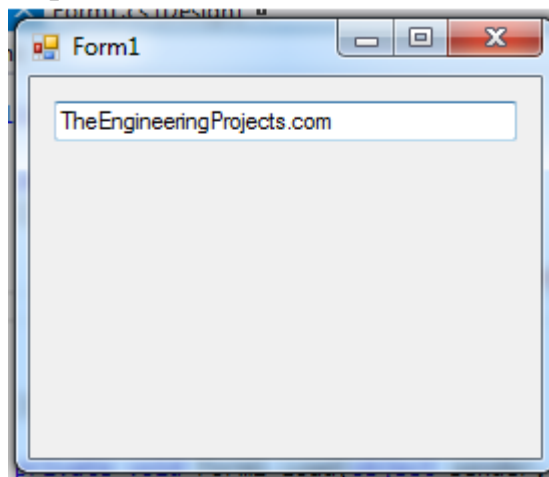


Рисунок 1.1 – Елемент TextBox

Значення текстового поля можна надати за допомогою змінних, наприклад, зберегти рядок у будь-якій змінній, а потім передати його в текстове поле:

```
string var = "TheEngineeringProjects.com";  
textBox1.Text = var;
```

Якщо необхідно зчитати значення з текстового поля, тоді можна використовувати такий код:

```
string var;  
var = textBox1.Text;
```

Для зміни текстового поля з панелі властивостей використовується комбінація F4 . Змінити висоту та ширину текстового поля можна на вкладці дизайнера, просто перетягнувши координати текстового поля або для динамічного змінення можна скористуватись кодом

```
textBox1.Width = 250;  
textBox1.Height = 50;
```

Для зміни кольору тла та тексту користуємось таким кодом (рис. 1.2):

```
textBox1.BackColor = Color.Blue;  
textBox1.ForeColor = Color.White;
```

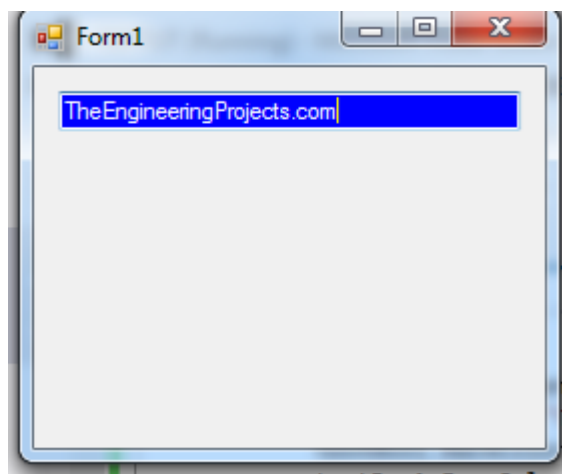


Рисунок 1.2 – Результат встановлення властивостей BackColor ForeColor для елемента textBox1

Існує три типи меж, які можна застосувати: FixedSingle, Fixed3D, None. Для зміни стилю меж для текстового поля використовується код:

```
textBox1.BorderStyle = BorderStyle.Fixed3D;  
textBox1.BorderStyle = BorderStyle.FixedSingle;  
textBox1.BorderStyle = BorderStyle.None;
```

Для встановлення максимальної довжини тексту всередині текстового поля використовується властивість.

```
textBox1.MaxLength = 40;
```

Для заборони введення або редагування тексту елемента textBox1 використовується властивість ReadOnly:

```
textBox1.ReadOnly = true;
```

Для того щоб зробити текстове поле багаторядковим текстовим полем використовується властивість Multiline:

```
textBox1.Multiline = true;
```

Якщо потрібно, щоб текстове поле використовувалось як поле для введення пароля, тоді використовується такий код:

```
textBox1.PasswordChar = '*';
```

Для того щоб користувач вводив будь-які дані з нового рядка, тоді можна використати один з двох способів:

```
// Перший метод
textBox1.Text += "ваш текст" + "\ r \ n";

// Другий метод
textBox1.Text += "ваш текст" + Environment.NewLine;
```

Число, що введено textBox1 являється текстом, тому для роботи з числом його необхідно перетворити в ціле число або дійсне число

```
int i;
i = int.Parse (textBox1.Text);
// String to Float перетворення
float i;
i = float.Parse (textBox1.Text);
//String to Double conversion
double i;
i = float.Parse (textBox1.Text);
```

Усі властивості, які були розглянуті, додані до фрагменту коду (рис. 1.3):

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            textBox1.Width = 250;
            textBox1.Height = 50;
            textBox1.Multiline = true;
            textBox1.BackColor = Color.Blue;
            textBox1.ForeColor = Color.White;
            textBox1.BorderStyle = BorderStyle.Fixed3D;
        }

        private void button1_Click(object sender, EventArgs e)
        {

```

```

        string var;
        var = textBox1.Text;
        MessageBox.Show(var);
        label1.Text = var;
    }
}

```

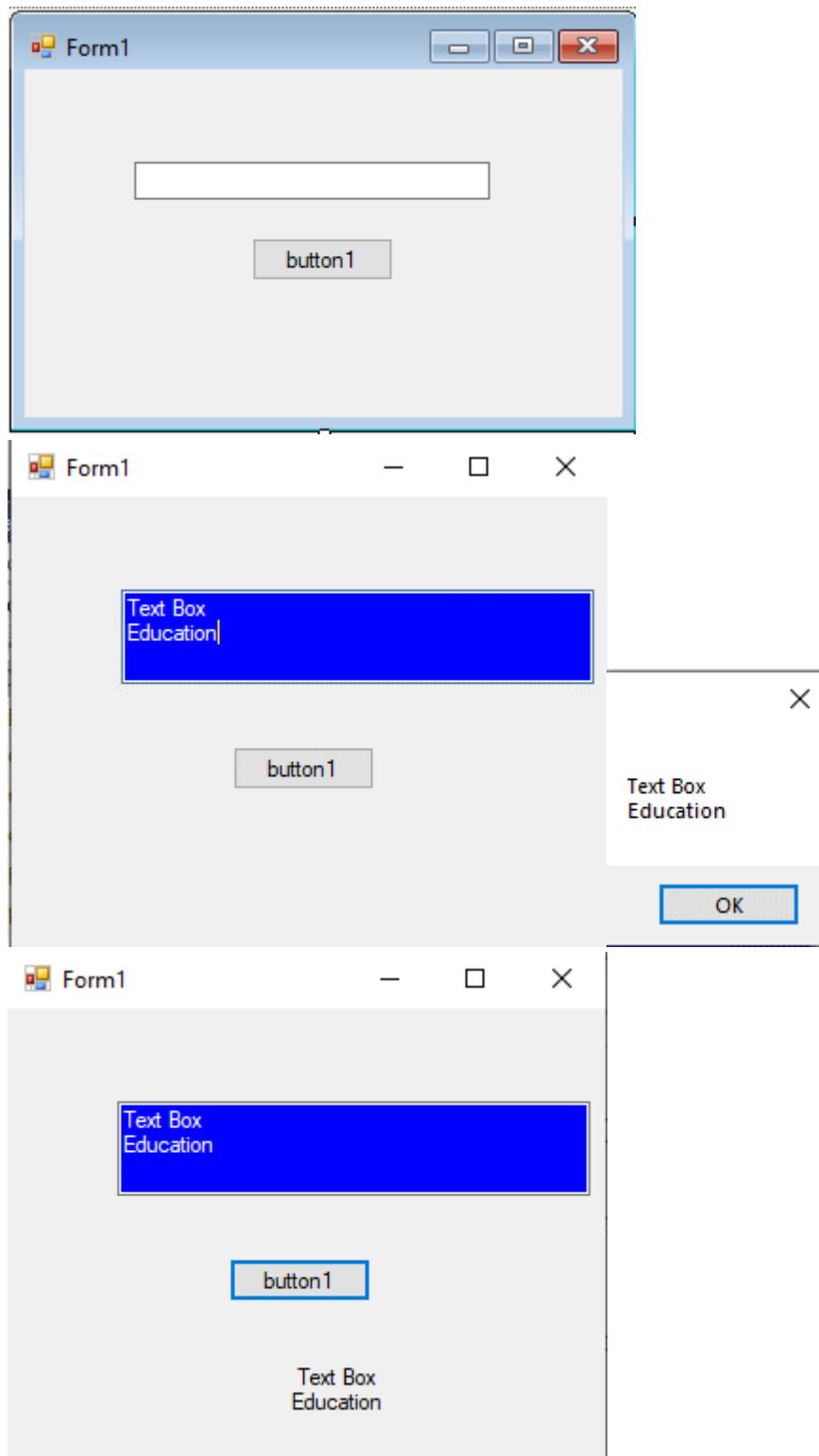


Рисунок 1.3 – Windows Forms, що показують особливості використання елемента textBox

1.2 Як використовувати у C# String змінні

Розробимо невеликий проект, щоб показати, як працює C# String.

Створюємо простий проект C# з однією кнопкою та одним текстовим полем, як показано на рис. 1.4

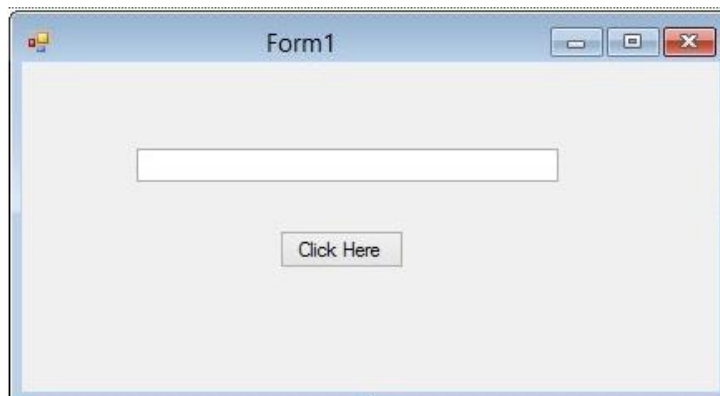


Рисунок 1.4 – Проект C# для вивчення особливостей роботи C# String

Змініть текст кнопки на «Click Here» та властивість «Name» на «ClickHere» .
Аналогічно змінюємо «Name» текстового поля на txtClick.
До проекту додаємо код, що представлений на рис. 1.5.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Variables
{
    public partial class Form1 : Form
    {
        string webBlog;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private void ClickHere_Click(object sender, EventArgs e)
        {
            webBlog = "www.TheEngineeringProjects.com";
            txtClick.Text = webBlog;
        }
    }
}
```

Рисунок 1.5 – Фрагмент коду проекту

У коді створена рядкова змінна **webBlog**, який присвоєне значення. Це значення відображено у текстовому полі (рис. 1.6).



Рисунок 1.6 – Результат роботи програми, що наведений на рис. 1.5

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Variables
{
    public partial class Form1 : Form
    {
        string webBlog1, webBlog2;

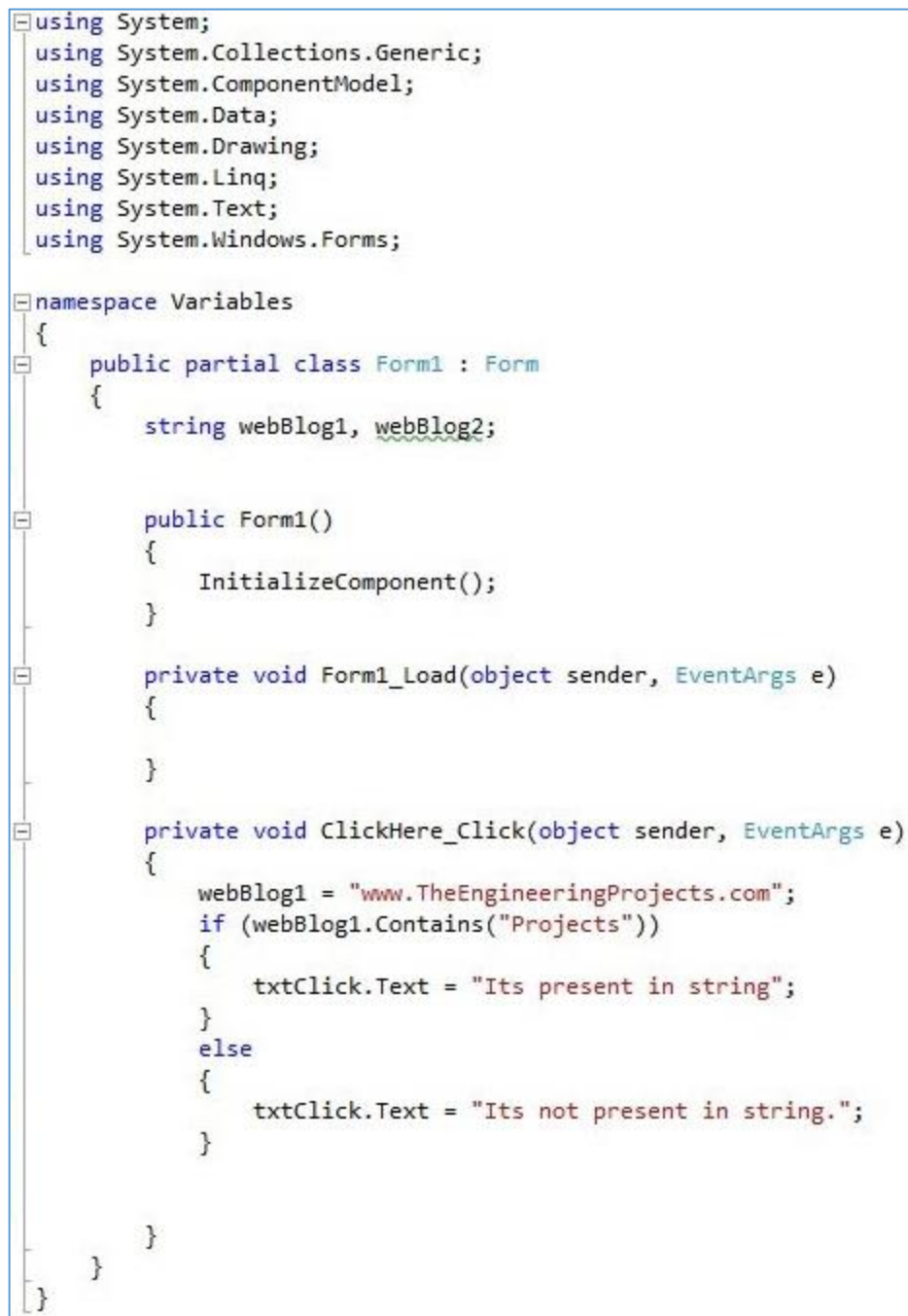
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private void ClickHere_Click(object sender, EventArgs e)
        {
            webBlog1 = "www.TheEngineeringProjects.com";
            webBlog2 = "www.TheEngineeringProjects.com";
            if (String.Compare(webBlog1, webBlog2) == 0)
            {
                txtClick.Text = "Both are equal";
            }
            else
            {
                txtClick.Text = "Both are not equal";
            }
        }
    }
}
```

Рисунок 1.7 – Фрагмент коду для порівняння двох рядків

Фрагмент коду, де порівнюються два рядки наведений на рис. 1.7. На рис. 1.8 наводиться фрагмент програми, яка показує як здійснювати пошук фрагменту тексту у рядку.

The image shows a screenshot of a C# code editor with a solution explorer on the left. The code is as follows:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Variables
{
    public partial class Form1 : Form
    {
        string webBlog1, webBlog2;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private void ClickHere_Click(object sender, EventArgs e)
        {
            webBlog1 = "www.TheEngineeringProjects.com";
            if (webBlog1.Contains("Projects"))
            {
                txtClick.Text = "Its present in string";
            }
            else
            {
                txtClick.Text = "Its not present in string.";
            }
        }
    }
}
```

Рисунок 1.8 – Фрагмент коду для пошуку фрагменту тексту у рядку

Припустимо, в якомусь проекті у вас дуже довгий рядок C #, і потрібна частина цього рядка C #. Тоді потрібно використовувати рядкову команду (рис. 1.9):

`webBlog1.Substring (4).`

У наведеному коді створені рядкові змінні C # String та задані їх значення. Другу змінна пропускає 4 символи першої змінної (рис. 1.10).

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Variables
{
    public partial class Form1 : Form
    {
        string webBlog1, webBlog2;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private void ClickHere_Click(object sender, EventArgs e)
        {
            webBlog1 = "www.TheEngineeringProjects.com";
            webBlog2 = webBlog1.Substring(4);
            txtClick.Text = webBlog2;
        }
    }
}

```

Рисунок 1.9 – Фрагмент коду для вивчення поділу рядка на частини



Рисунок 1.10 – Результат виконання програми, що наведена на рис. 1.9

1.3 Перетворення типів даних у C

При роботі над якоюсь програмою, керованою даними, завжди потрібно перетворити одну форму даних в іншу, і саме там потрібні перетворення даних.

Невелика кількість перетворень не потребує жодного методу чи класу, тобто якщо ми хочемо перетворити ціле число в float, то компілятор може це легко зробити. Такі перетворення називаються неявними (рис. 1.11).

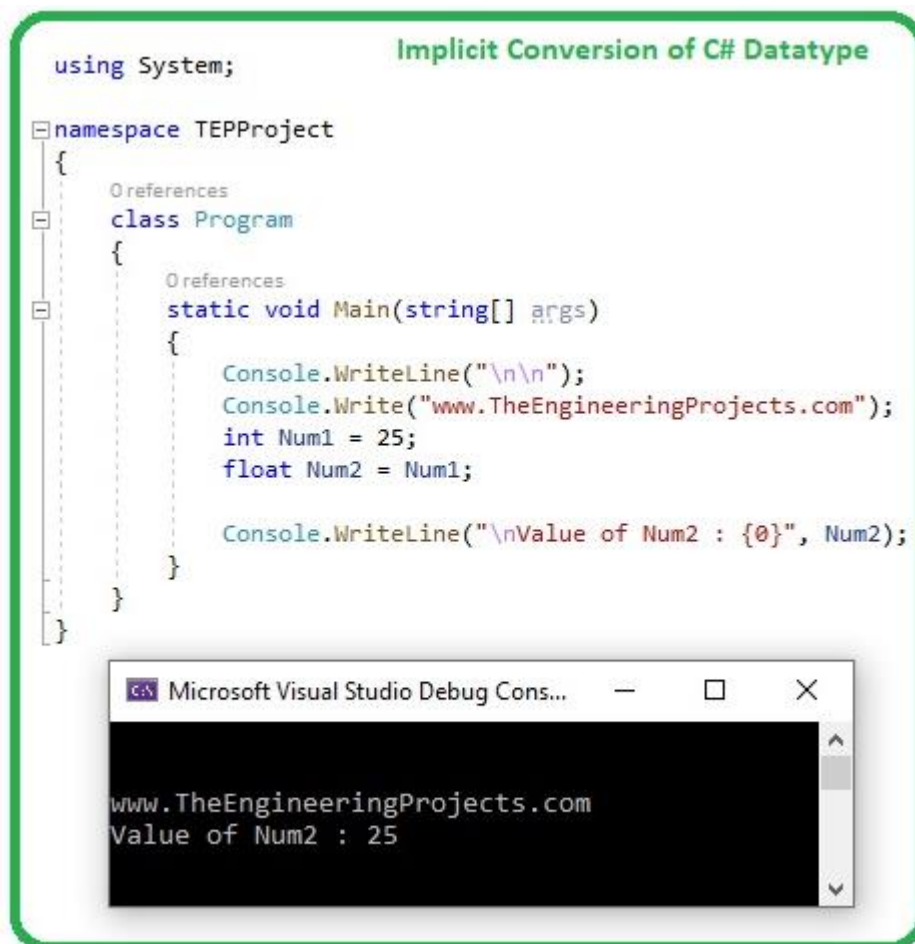


Рисунок 1.11 – Неявне перетворення даних типу Int у Float

Але якщо потрібно перетворити float (скажімо, 313.56) у ціле число, то виникає помилка компілятора, тобто виняток переповнення. Для здійснення таких перетворень необхідно використовувати метод явних перетворень, попередньо визначений у C #. Є два варіанти явних перетворень: Cast Operator та клас Convert у C #. Способи перетворення даних наведені на рис. 1.12. Оператор Cast (int) бере основне значення і ігнорує десяткову / дробову частину. Клас перетворення (Convert.ToInt32) округлює число і не ігнорує десяткової частини.

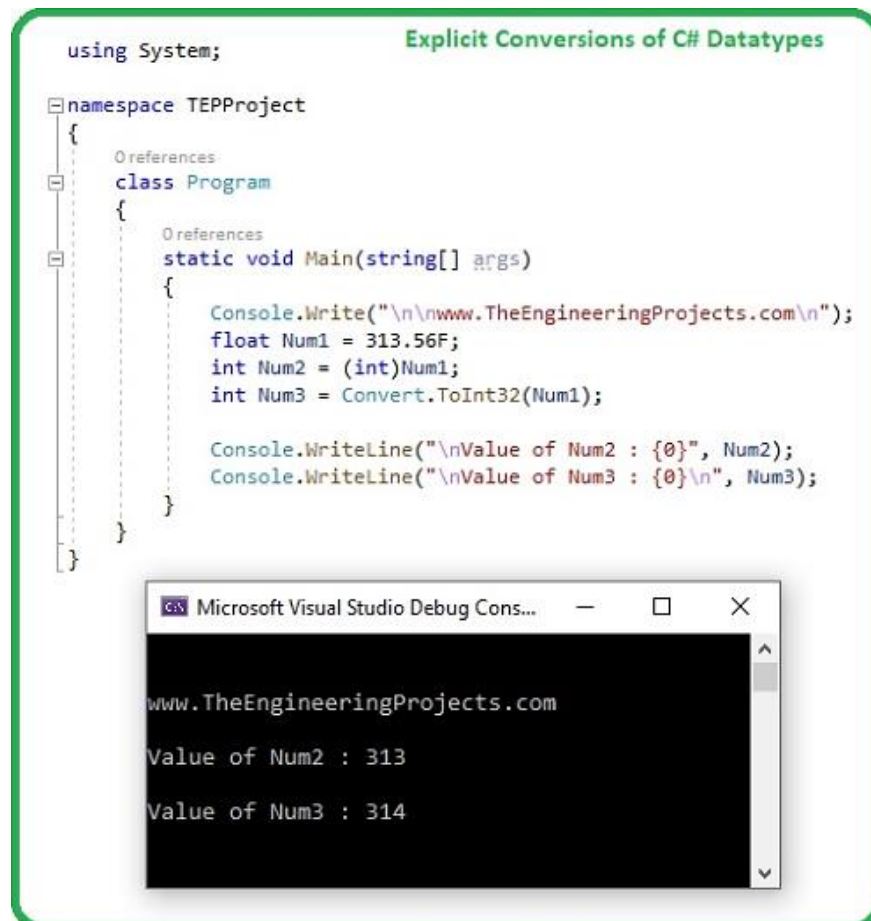


Рисунок 1.12 – Явне перетворення даних типу Float у Int

Для string перетворень в C # є два варіанти: Parse, TryParse. Приклад реалізація методу Parse для перетворення рядка у цілочисельне значення наведено на рис. 1.13.

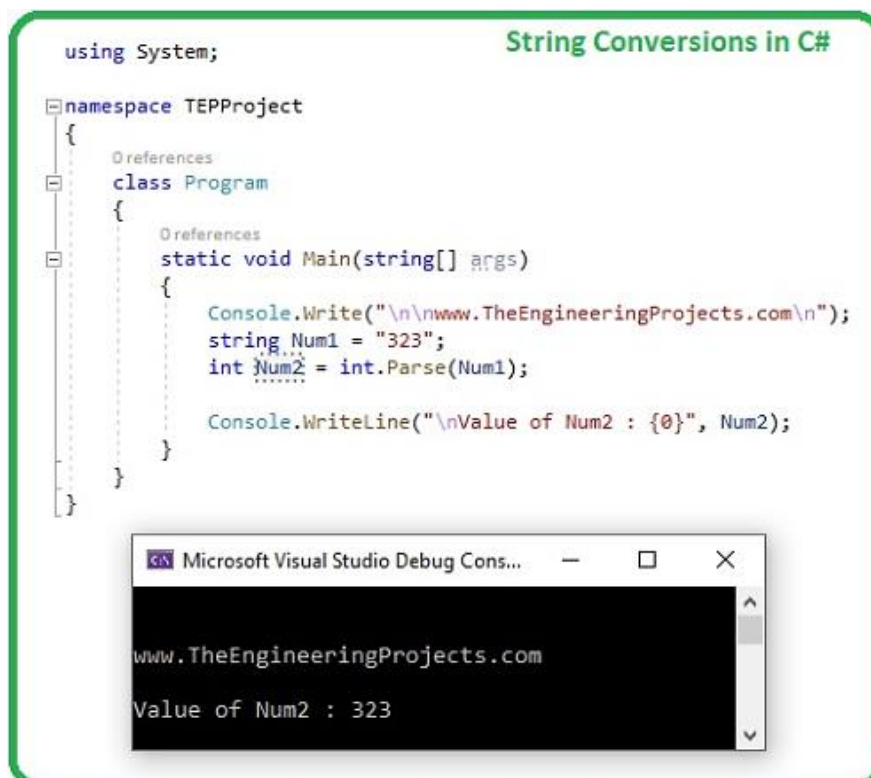


Рисунок 1.13 – Застосування методу Parse перетворення String в Int

Метод Parse має недолік: якщо рядок містить букви або спеціальні символи, то це створить помилку. Щоб уникнути помилок необхідно використовувати метод TryParse (рис. 1.14):

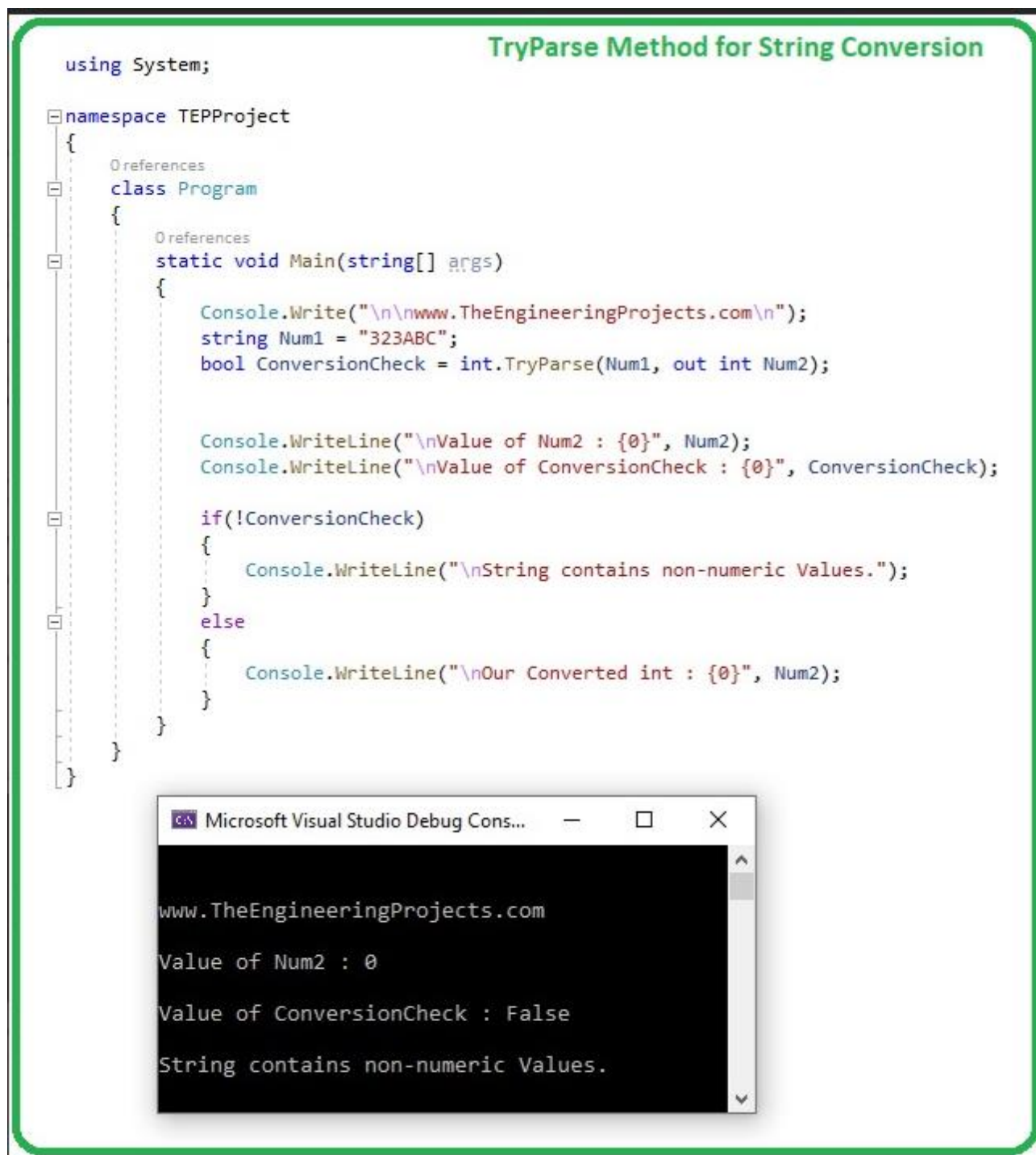


Рисунок 1.14 – Застосування методу TryParse перетворення String в Int

Метод TryParse приймає два параметри: рядок, який перетворюється та ціле число, в якому значення буде збережено. Метод TryParse також повертає булевий тип, який буде: True, якщо конверсія пройшла успішно та False, якщо перетворення не вдалося. У прикладі рядок "323ABC" містить букви, тому ConversionCheck помилковий, і не отримуємо int значення у Num2 .

1.4 Як користуватися масивом C # ?

Масив має вигляд:

```
FirstArray [3] = {Елемент1, Елемент2, Елемент3};
```

Масив має ім'я FirstArray, і він може містити максимум три члени, що відображається в [] дужках.

Встановлені значення трьох його членів, які розділені комами. Для отримання доступу до окремих членів цього масиву C # необхідно викликати їх за їх індексами. Отже, перший елемент кожного масиву C # завжди є 0.

```
FirstArray [0] = Елемент1;  
FirstArray [1] = Елемент2;  
FirstArray [2] = Елемент3;
```

Створемо простий проект C # на якій додаємо елементи керування: кнопку ClickHere та текстове поле txtClick (рис. 1.15).

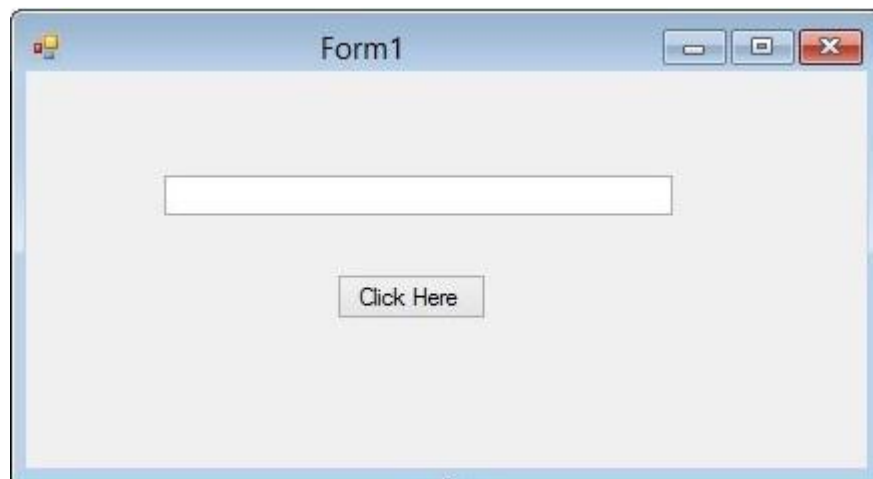


Рисунок 1.15 – Вікно форми з кнопкою та текстовим полем

Додамо масив рядків C # у проект. Спочатку потрібно оголосити масив String C #. Для цього використовується такий код код:

```
// C# Array Initializing  
String[] students = new String[5];  
// Initializing Complete
```

Ім'я масиву C # **students** і використання оператора **new** для створення нового екземпляра масиву C #. Задана довжина масиву – 5 елементів. Можна не задавати довжину масиву ,як показано нижче:

```
// C# Array Initializing  
String[] students = new String[];  
// Initializing Complete
```

Один із способів додавання значень до масиву C # такий:

```
// Adding values to C# Array.  
students[0] = "Zain";  
students[1] = "Nasir";
```

```
students[2] = "Kamraan";  
students[3] = "John";  
students[4] = "Jack";  
// Values added.
```

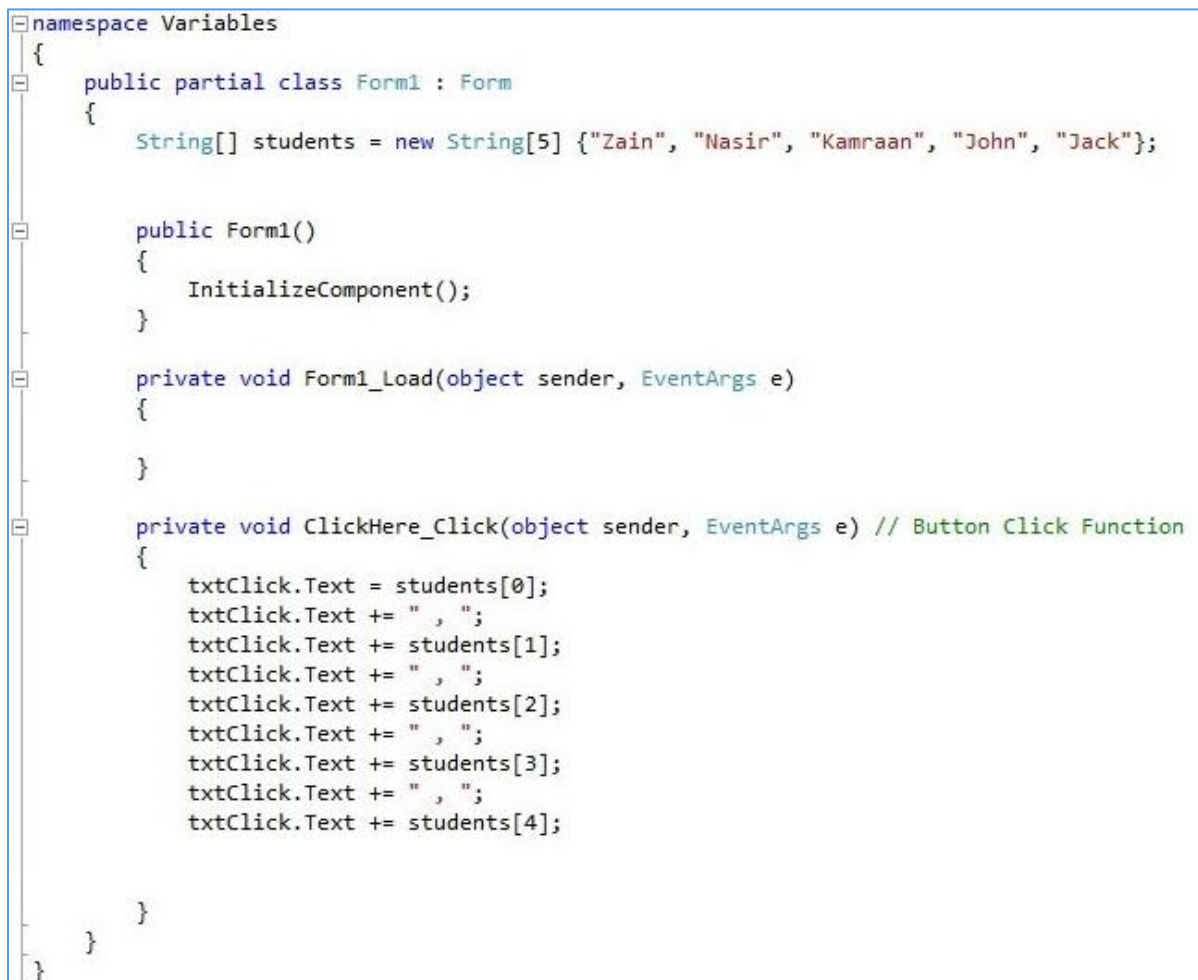
Ще один із способів призначення значень масиву C #:

```
// Adding values to C# Array.  
String[] students = new String[5] {"Zain", "Nasir", "Kamraan",  
"John", "Jack"};  
// Values added.
```

Тепер давайте отримаємо елементи з цього масиву та відобразимо їх у текстовому полі. Для цього використовуйте наведений нижче код:

```
txtClick.Text = students[0];  
txtClick.Text += " , ";  
txtClick.Text += students[1];  
txtClick.Text += " , ";  
txtClick.Text += students[2];  
txtClick.Text += " , ";  
txtClick.Text += students[3];  
txtClick.Text += " , ";  
txtClick.Text += students[4];
```

Повний код наведений на рис. 1.16



```
namespace Variables  
{  
    public partial class Form1 : Form  
    {  
        String[] students = new String[5] {"Zain", "Nasir", "Kamraan", "John", "Jack"};  
  
        public Form1()  
        {  
            InitializeComponent();  
        }  
  
        private void Form1_Load(object sender, EventArgs e)  
        {  
        }  
  
        private void ClickHere_Click(object sender, EventArgs e) // Button Click Function  
        {  
            txtClick.Text = students[0];  
            txtClick.Text += " , ";  
            txtClick.Text += students[1];  
            txtClick.Text += " , ";  
            txtClick.Text += students[2];  
            txtClick.Text += " , ";  
            txtClick.Text += students[3];  
            txtClick.Text += " , ";  
            txtClick.Text += students[4];  
        }  
    }  
}
```

Рисунок 1.16 – Код програми з використанням елементів масиву



Рисунок 1.17 – Результати роботи програми, що наведена на рис. 1.16

1.5 Як користуватися елементом ArrayList?

Елемент ArrayList використовується для зберігання в ньому даних так само, як C # масиви, але між ними є незначна різниця. У C # ArrayList можна будь-коли додавати або видаляти дані, ArrayList налаштовується автоматично.

Додавання або видалення даних у C # ArrayList здійснюється за допомогою індексів цих даних. Отже, коли додаються дані, то ArrayList автоматично розтягується і створює вхідні дані в новому індексі. Аналогічно, коли видаляються дані з ArrayList, вони зменшуються і відповідно коригують дані. Для ініціалізувати ArrayList використовується код:

```
// .... Ініціалізація C # ArrayList ....  
    ArrayList TEP = new ArrayList ();  
// .... Закінчується тут .....
```

Створений ArrayList має ім'я TEP. Важливим є те, що необхідно додати до простору імен **using System.Collections**.

Для додавання даних використовується команда **TEP.Add ()**, де TEP – це ім'я ArrayList, а дані подають у дужках. Додамо дані до ArrayList, використовуючи такий код:

```
// .... Adding Data in ArrayList ....  
    TEP.Add("The");  
    TEP.Add("Engineering");  
    TEP.Add("Projects");  
// .... Data added in ArrayList ....
```

Додано три значення у TEP ArrayList. Виведемо ці значення у поле txtClick за подією Click кнопки ClickHere. До функції Click додаємо код:

```
// ... Displaying values ....  
    txtClick.Text = TEP[0].ToString();  
    txtClick.Text += " , ";  
    txtClick.Text += TEP[1].ToString();
```



```

        txtClick.Text += " , ";
        txtClick.Text += TEP[2].ToString();
    // ... Values Displayed .....

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Collections;

namespace Variables
{
    public partial class Form1 : Form
    {
        ArrayList TEP = new ArrayList();

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            TEP.Add("The");
            TEP.Add("Engineering");
            TEP.Add("Projects");
        }

        private void ClickHere_Click(object sender, EventArgs e) // Button Click Function
        {
            txtClick.Text = TEP[0].ToString();
            txtClick.Text += " , ";
            txtClick.Text += TEP[1].ToString();
            txtClick.Text += " , ";
            txtClick.Text += TEP[2].ToString();
        }
    }
}

```

Initialization

Adding Data

Displaying Data

Рисунок 1.18 – Код програми для демонстрації роботи з ArrayList

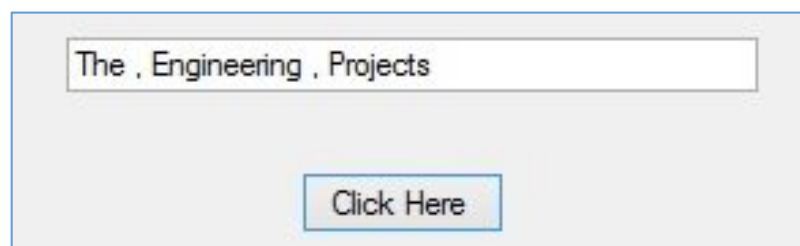


Рисунок 1.19 –Результат виконання коду програми для демонстрації роботи з ArrayList

Розглянемо, як підрахувати загальну кількість елементів у ArrayList. Для цього вам потрібно додати TEP.Count до події Click кнопки ClickHere, як показано нижче:

```
// ..... Display Values .....  
txtClick.Text = TEP[0].ToString();  
txtClick.Text += " , ";  
txtClick.Text += TEP[1].ToString();  
txtClick.Text += " , ";  
txtClick.Text += TEP[2].ToString();  
txtClick.Text += " , ";  
txtClick.Text += TEP.Count;  
// ..... Values Displayed .....
```

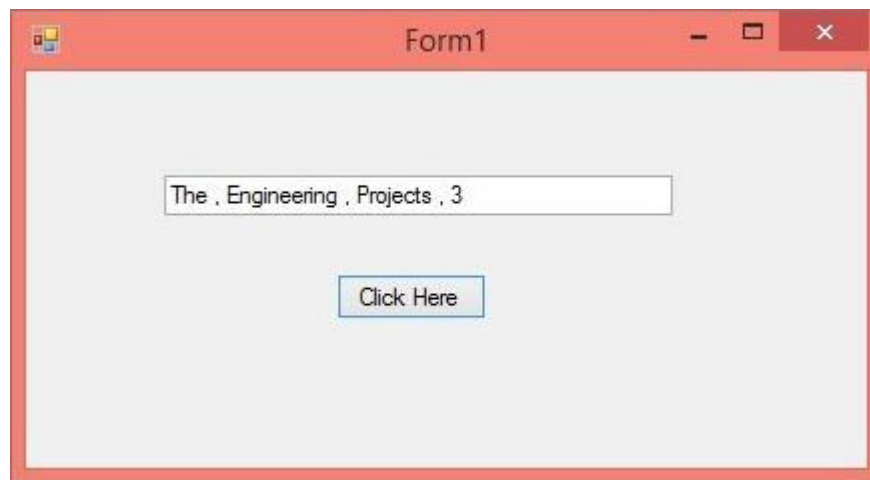


Рисунок 1.20 – Результат виконання коду програми з підрахунком загальної кількості елементів у ArrayList

Для очищення ArrayList використовується команда TEP.Clear(). У наведеному нище коді відображені значення ArrayList та виводиться загальна кількість елементів ArrayList; далі очищається ArrayList командою TEP.Clear () та знову виводиться загальна кількість елементів. Так як ArrayList очищений, то в ньому немає елементів і буде виведено значення 0 (рис. 1.21):

```
txtClick.Text = TEP[0].ToString();  
txtClick.Text += " , ";  
txtClick.Text += TEP[1].ToString();  
txtClick.Text += " , ";  
txtClick.Text += TEP[2].ToString();  
txtClick.Text += " , ";  
txtClick.Text += TEP.Count;  
TEP.Clear();  
txtClick.Text += " , ";  
txtClick.Text += TEP.Count;
```

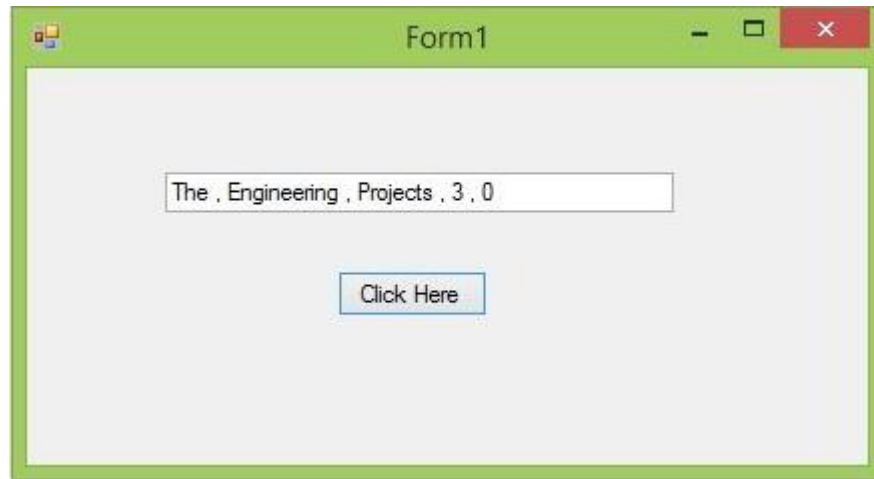



Рисунок 1.21 – Демонстрація роботи команди TEP.Count

2. Практичне завдання



У процесі виконання завдань лабораторної роботи необхідно формувати набори тестових даних для перевірки правильності виконання програмного коду. Створений код і результати перевірки його роботи потрібно помістити у звіт. Тестувати роботу програми рекомендується після додання чи зміни кожного оператора виведення.

Завдання 1. Створити проект для розв'язання задачі.

Два потяги виїхали одночасно назустріч один одному. Потрібно знайти, через який час вони зустрінуться, якщо задано значення відстані між ними в момент початку руху та швидкості руху кожного потяга (рис. 2.1).

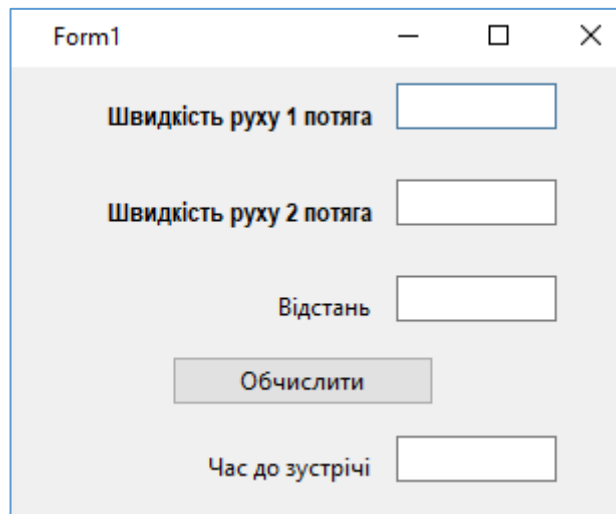


Рисунок 2.1 – Вікно форми для завдання 1

Завдання 2. Скласти програму для знаходження значення функції при різних значеннях x .

1. $y = \sqrt{|x - 1|} + \sin x$

2. $y = 1 + \frac{1}{x} + \frac{1}{x^2}$

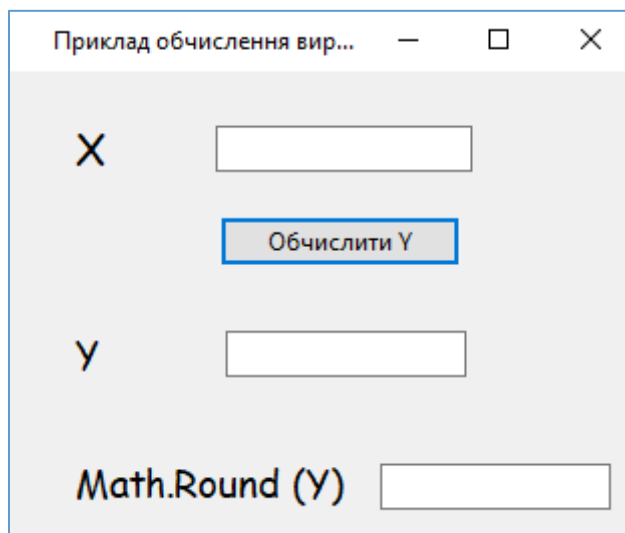


Рисунок 2.2 – Вікно форми для завдання 2

Завдання 3. Скласти програму для визначення найбільшого з трьох чисел а, b, c. Значення змінних а, b і c мають бути впорядковані за зростанням (поміняти місцями а, b і c так, щоб виявилось $a \leq b \leq c$).

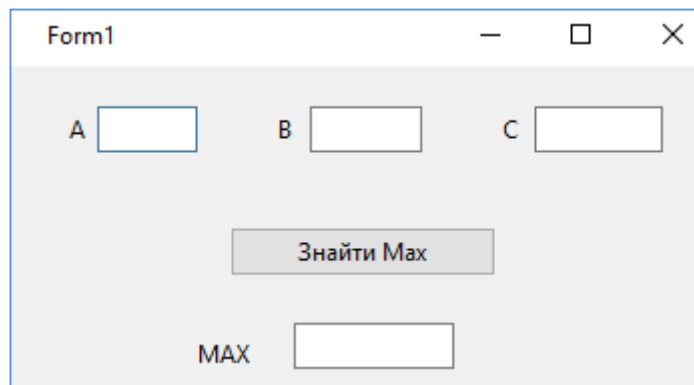


Рисунок 2.3 – Вікно форми для завдання 3

3. Контрольні запитання

1. Для чого використовується елемент TextBox ?
2. Як ввести та вивести дані через TextBox ?
3. Наведіть властивості елемента TextBox. Як їх можна попередньо встановити та змінити під час роботи програми?
4. Як задати максимальну довжини тексту та задати багаторядковий текстове поле?
5. Які існують методи перетворення типів числових даних між собою?
6. Які існують методи перетворення рядкових даних у числові та навпаки?
7. Як порівняти між собою рядкові дані, як здійснювати пошук фрагменту тексту у рядку?
8. Як оголосити масив рядків та задати йому значення?
9. Які переваги та особливості застосування елемента ArrayList?

Література

1. Евдокимов П. В. С# на примерах. СПб.: Наука и Техника, 2019. 320 с.
2. Маки А. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов; пер. с англ. М. : ООО ИД "Вильямс", 2010. 416 с
3. С# 7.0. Справочник. Полное описание языка.: Пер. с англ. СПб.: ООО "Альфа-книга", 2018. 1024 с.
4. Троелсен, Эндрю, Джепикс, Филипп. Язык программирования С# 7 и платформы .NET и .NET Core. СПб. : ООО "Диалектика", 2018. 1328 с.
5. Офіційний сайт компанії Microsoft щодо технологій WPF та Windows Forms [Електронний ресурс]. – Режим доступу : <http://window-sclient.net>.
6. С#. Теорія та практика. URL: https://www.bestprog.net/uk/sitemap_ua/c-3
7. Сажин А. Справочник по языку программирования С#. URL:

<https://brainoteka.com/blogs/c-spravochnik>.

8. C# Tutorial URL <https://www.theengineeringprojects.com>.

9. Уроки C#. URL: <https://itproger.com/course/csharp>.

10. Полное руководство по C# 8 и .NET Core. URL: <https://metanit.com/sharp/>