

Лабораторна робота №4 Windows Forms. Елемент ListBox

Тема: Елемент ListBox.

Мета заняття: навчитись працювати з елементом ListBox у середовищі Microsoft Visual Studio; набути практику додавання, видалення даних, зміни властивостей, їх обробки та виводу результатів з допомогою Windows Forms мовою C#; навчитись використовувати події (events) при роботі з елементом ListBox.

1. Теоретичні відомості

1.1 Додавання елементів до ListBox

Елемент ListBox використовується у формах реєстрації та додатках з продажу товару. Для роботи елементом з ListBox потрібно з панелі інструментів перетягнути його на форму Form1. Для встановлення значень всередині Listbox потрібно написати назву поля списку, а після цього написати **Items.Add ("YourData")**, наприклад (рис. 1.1):

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

namespace TEPTUT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            listBox1.Items.Add("Pakistan");
            listBox1.Items.Add("United States");
            listBox1.Items.Add("United Kingdom");
            listBox1.Items.Add("India");
        }
    }
}
```



Рисунок 1.1 – Додавання елементів до ListBox

Можно додати значення в `ListBox` за допомогою масивів та змінних типу `string`. Приклад коду, який демонструє використання для вставки значень елементу `ListBox` у `C #` рядкових змінних:

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

namespace TEPTUT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            String var1 = "Lahore";
            String var2 = "Multan";
            String var3 = "Islamabad";
            String var4 = "Karachi";
            listBox1.Items.Add(var1);
            listBox1.Items.Add(var2);
            listBox1.Items.Add(var3);
            listBox1.Items.Add(var4);
        }
    }
}
```

Приклад коду, який демонструє використання для вставки значень елементу `ListBox` у `C #` масиву:

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

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

            String[] city = new String[] { "Lahore", "Multan",
            "Islamabad", "Karachi" };

            for (int x = 0; x < city.Length; x++)
            {
                listBox1.Items.Add(city[x]);
            }
        }
    }
}
```

Приклад коду, який демонструє читання даних з TextBox та їх копіювання до елементу ListBox у C # (рис. 1.2)

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

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

        private void button1_Click(object sender, EventArgs e)
        {
            listBox1.Items.Add(textBox1.Text);
        }
    }
}
```

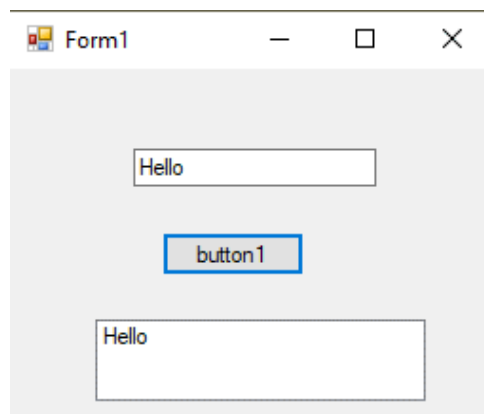


Рисунок 1.2 – Читання даних з TextBox та їх копіювання до елементу ListBox

Розглянемо як вибрати будь-який елемент з ListBox. Для отримання вибраного значення елемента використовується властивість SelectedItem.

У коді, що наведений нижче, після вибору значення з ListBox та натисненні кнопки, виникає подія, яка створить спливаюче повідомлення про вибране значення елемента.

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

namespace TEPTUT
```

```

{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            listBox1.Items.Add("A");
            listBox1.Items.Add("B");
            listBox1.Items.Add("C");
            listBox1.Items.Add("D");
        }

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show(listBox1.SelectedItem.ToString());
        }
    }
}

```

1.2 Вибір кількох елементів *ListBox*

Для вибору кількох варіантів одночасно необхідно у властивості **SelectionMode** встановити значення **SelectionMode.MultiSimple**.

У прикладі коду (рис. 1.3) використовується властивість **SelectionMode.MultiSimple** та цикл **foreach**, щоб показати спливаюче повідомлення зі значенням вибраних елементів.

```

namespace LIST_BOX
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            listBox1.Items.Add("Sunday");
            listBox1.Items.Add("Monday");
            listBox1.Items.Add("Tuesday");
            listBox1.Items.Add("Wednesday");
            listBox1.Items.Add("Thursday");
            listBox1.Items.Add("Friday");
            listBox1.Items.Add("Saturday");
            listBox1.SelectionMode = SelectionMode.MultiSimple;
        }
        private void button1_Click(object sender, EventArgs e)
        {
            //listBox1.Items.Add(textBox1.Text);
            //MessageBox.Show(listBox1.SelectedItem.ToString());
            foreach (Object obj in listBox1.SelectedItems)
            {
                MessageBox.Show(obj.ToString());
            }
        }
    }
}

```

Рисунок 1.3 – Демонстраційний код використання властивості **SelectionMode**

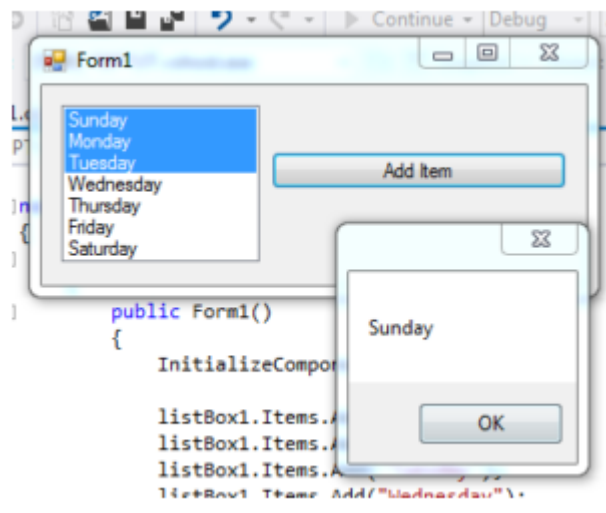


Рисунок 1.4 – Демонстрація роботи форми з вибору кількох елементів ListBox

З рис. 1.4 видно, що вибрані перші три індекси, і при натисненні кнопки генерується спливаюче повідомлення лише з Sunday.

Так як у програмі (рис. 1.3) використаний цикл foreach, то кожна ітерація циклу містить лише один вивід. Коли буде закрито спливаюче вікно, у полі повідомлень відображається наступне вибране значення індексу. Цикл працює, поки не залишиться вибраних індексів.

1.3 Очищення та видалення значень ListBox

Для видалення всіх значень зі списку використовується метод `Items.Clear`. У коді, що наведений нижче встановлюються значення у Listbox, і коли користувач натисне на кнопку, очищається весь список:

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

namespace TEPTUT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            listBox1.Items.Add("A");
            listBox1.Items.Add("B");
            listBox1.Items.Add("C");
            listBox1.Items.Add("D");
        }
    }
}
```

```

        private void button1_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
        }
    }
}

```

Для видалення певного індексу або значення елемента використовують методи `Remove` або `RemoveAt`. Метод `RemoveAt` використовується для видалення значення за індексом. Демонстраційний код для методу `RemoveAt`:

```

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

namespace TEPTUT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            listBox1.Items.Add("A");
            listBox1.Items.Add("B");
            listBox1.Items.Add("C");
            listBox1.Items.Add("D");
        }

        private void button1_Click(object sender, EventArgs e)
        {
            listBox1.Items.RemoveAt(0);
        }
    }
}

```

У коді користувач натискає кнопку, нульовий індекс видаляється зі списку. Коли нульовий індекс видаляється, наступний індекс автоматично стає нульовим. Наступний код демонструє реалізацією методу `Remove`, який використовується для видалення значення за точним формулюванням:

```

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

namespace TEPTUT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            listBox1.Items.Add("A");
            listBox1.Items.Add("B");
        }
    }
}

```

```

        listBox1.Items.Add("C");
        listBox1.Items.Add("D");
    }

    private void button1_Click(object sender, EventArgs e)
    {
        listBox1.Items.Remove("A");
    }
}

```

У наведеному коді видаляється лише точне відповідне значення «А» за натисненням кнопки. При наступному натисканні кнопки не видалиться жоден індекс.

1.4 Зміна властивостей тексту ListBox

Для зміни розміру тексту у ListBox використовується властивість **Font**. У демонстраційному коді після натиснення кнопки користувачем збільшиться розмір тексту C # ListBox (рис. 1.5).

```

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

namespace TEPTUT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            listBox1.Items.Add("A");
            listBox1.Items.Add("B");
            listBox1.Items.Add("C");
            listBox1.Items.Add("D");
        }

        private void button1_Click(object sender, EventArgs e)
        {
            listBox1.Font = new Font(Font.FontFamily, 12);
        }
    }
}

```

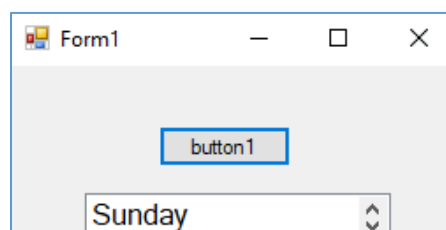


Рисунок 1.5 – Демонстрація використання властивості Font елементу ListBox

Для зміни кольору тексту або кольору фону тексту використовується властивість `BackColor`. У демонстраційному коді змінюється колір тексту тексту списку поля.

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

namespace TEPTUT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            listBox1.Items.Add("A");
            listBox1.Items.Add("B");
            listBox1.Items.Add("C");
            listBox1.Items.Add("D");
            listBox1.BackColor = Color.LightCyan;
        }
    }
}
```

Для зміни кольору тексту або кольору переднього плану використовують властивість `ForeColor`. У демонстраційному коді змінюється колір тексту на білий, а `BackColor` - на чорний (рис. 1.6).

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

namespace TEPTUT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            listBox1.Items.Add("The");
            listBox1.Items.Add("Engineering");
            listBox1.Items.Add("Projects");
            listBox1.Font = new Font(Font.FontFamily, 15);
            listBox1.BackColor = Color.Black;
            listBox1.ForeColor = Color.White;
        }
    }
}
```

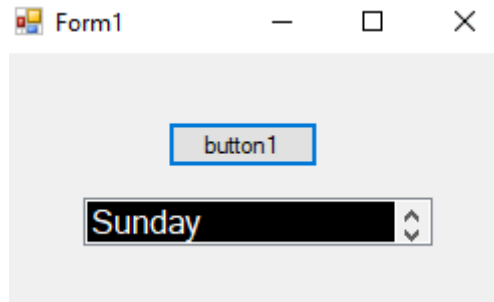


Рисунок 1.6 – Демонстрація роботи властивостей ForeColor та BackColor

1.5 Події ListBox у C

Елемент ListBox може використовувати такі події (Events): Click, DoubleClick, ForeColorChanged, BackColorChanged, SelectedIndexChanged, MouseHover, MouseLeave,

Подія ListBox Click використовується, коли потрібно виконати будь-яку дію у відповідь одним клацанням на Listbox, наприклад, показати будь-яке повідомлення попередження у вигляді спливаючого вікна (рис. 1.7):

```
private void listBox1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Warn ! Click is disable..");
}
```

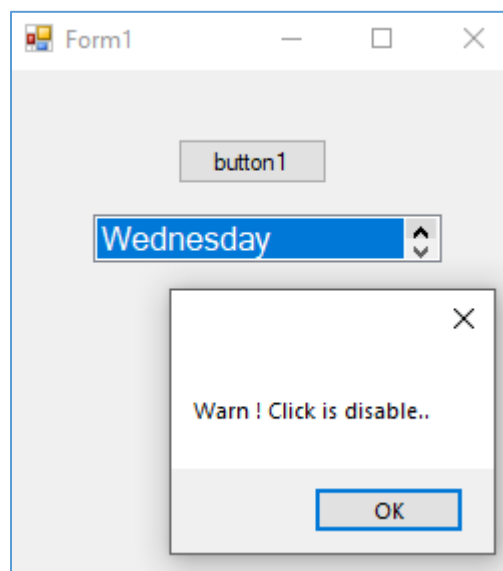


Рисунок 1.7 – Демонстрація реакції на подію ListBox Click

Ця подія DoubleClick відбувається, коли користувач два рази натисне на Listbox. У демонстраційному коді змінюється колір списку ListBox, коли користувач натисне двічі на елемент ListBox.

```
private void listBox1_DoubleClick(object sender, EventArgs e)
{
    listBox1.BackColor = Color.Red;
}
```

Подія `MouseHover` відбувається, коли користувач наведе курсор миші на `Listbox`. У демонстраційному коді змінюється властивість `foreColor` при наведенні курсору миші:

```
private void listBox1_MouseHover(object sender, EventArgs e)
{
    listBox1.ForeColor = Color.Bisque;
}
```

Подія `MouseLeave` відбувається, коли користувач відведе курсор від списку. У демонстраційному коді змінюється фоновий колір на чорний, коли користувач відведе курсор миші.

```
private void listBox1_MouseHover(object sender, EventArgs e)
{
    listBox1.ForeColor = Color.Bisque;
}

private void listBox1_MouseLeave(object sender, EventArgs e)
{
    listBox1.ForeColor = Color.Black;
}
```

Подія `BackColorChanged` відбувається, коли користувач змінить колір фону списку. У демонстраційному коді буде змінюватись колір тла, коли користувач наведе мишу на елемент `Listbox`. Коли колір зміниться, відбудеться подія `BackColorChanged`, у якій буде показано спливаюче повідомлення.

```
private void listBox1_MouseHover(object sender, EventArgs e)
{
    listBox1.BackColor = Color.Bisque;
}

private void listBox1_BackColorChanged(object sender, EventArgs e)
{
    MessageBox.Show("BackGround Color is changed !!");
}
```

Подія `ForeColorChanged` відбувається, коли користувач змінить колір переднього плану списку. У демонстраційному коді використовується подія наведення миші, щоб змінити `foreColor` та викликати подію `ForeColorChanged`. `ForeColorChanged` подія викликає спливаюче повідомлення, що колір переднього плану змінений.

```

private void listBox1_MouseHover(object sender, EventArgs e)
{
    listBox1.ForeColor = Color.Bisque;
}

private void listBox1_ForeColorChanged(object sender, EventArgs e)
{
    MessageBox.Show("ForeGround Color is changed !!");
}

```

Подія `SelectedIndexChanged` відбувається, коли користувач вибере будь-який елемент у списку. У демонстраційному коді використовуються два списки (рис. 1.8). Коли користувач вибере будь-яке значення з першого списку, відповідно це значення відобразиться у другому списку (рис. 1.9).

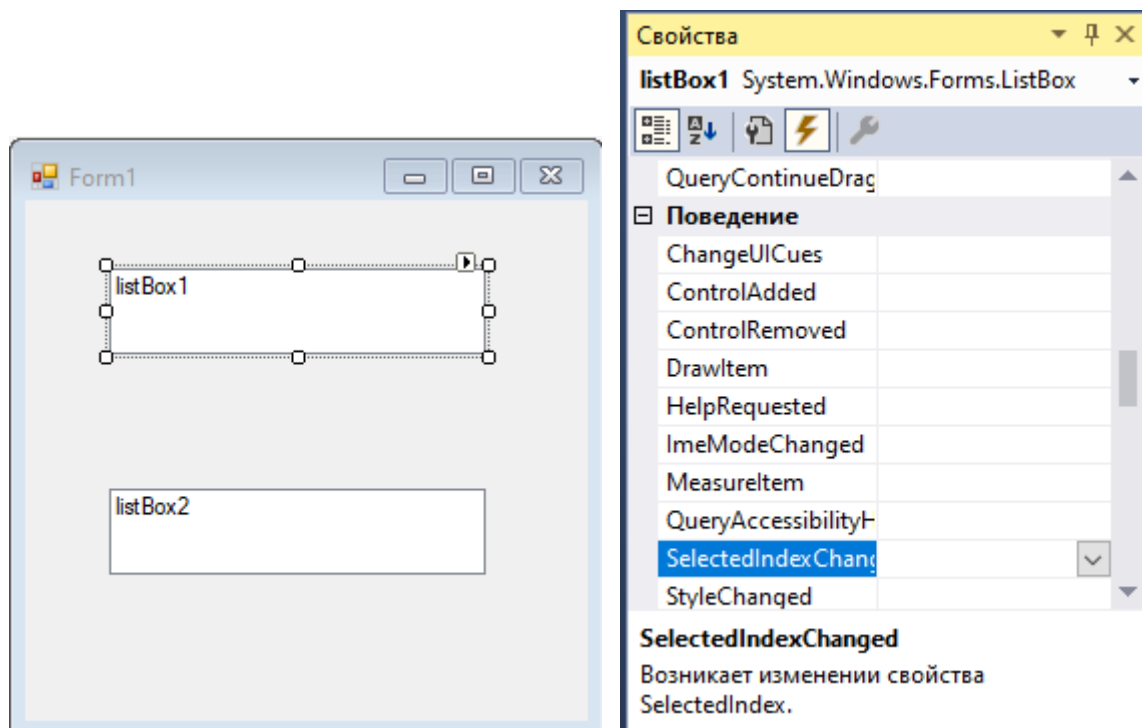
```

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

namespace TEPTUT
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            listBox1.Items.Add("The Engineering Projects");
            listBox1.Items.Add("C# Tutorials");
        }

        private void listBox1_SelectedIndexChanged(object sender,
            EventArgs e)
        {
            if (listBox1.SelectedIndex == 0)
            {
                listBox2.Items.Clear();
                listBox2.Items.Add("Arduino Projects");
                listBox2.Items.Add("Proteus Projects");
                listBox2.Items.Add("Matlab Projects");
            }
            else
            {
                if (listBox1.SelectedIndex == 1)
                {
                    listBox2.Items.Clear();
                    listBox2.Items.Add("C# Label");
                    listBox2.Items.Add("C# TextBox");
                    listBox2.Items.Add("C# ComboBox");
                }
            }
        }
    }
}

```



```

namespace LIST_BOX2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            listBox1.Items.Add("The Engineering Projects");
            listBox1.Items.Add("C# Tutorials");
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (listBox1.SelectedIndex == 0)
            {
                listBox2.Items.Clear();
                listBox2.Items.Add("Arduino Projects");
                listBox2.Items.Add("Proteus Projects");
                listBox2.Items.Add("Matlab Projects");
            }
            else
            {
                if (listBox1.SelectedIndex == 1)
                {
                    listBox2.Items.Clear();
                    listBox2.Items.Add("C# Label");
                    listBox2.Items.Add("C# TextBox");
                    listBox2.Items.Add("C# ComboBox");
                }
            }
        }
    }
}

```

Рисунок 1.8 – Форма та код програми для демонстрації події SelectedIndexChanged

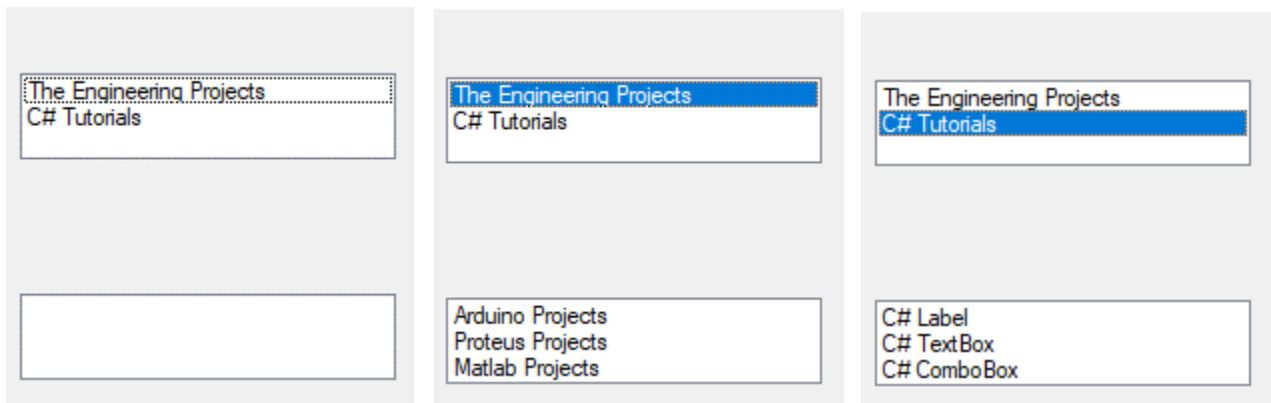


Рисунок 1.9 – Демонстрація роботи програми, що використовує подію SelectedIndexChanged

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.

Дано послідовність із N цілих чисел ($N \leq 100$). Визначити кількість парних чисел, які мають парні індекси.

1) Створіть новий проект. Розмістіть на формі елементи Label1, Label2, Button1 (Name - bT1), Button2 (Name – bT2), ListBox (Name – listB1), TextBox1 (Name – texB1) TextBox2 (Name – texB2) і налаштуйте їхні властивості згідно з рис.2.1.

Рисунок 2.1 – Зовнішній вигляд форми для реалізації завдання 1

2) Для зберігання елементів використаємо масив. Оскільки кількість елементів послідовності $N \leq 100$,

опишіть масив: `int[] Num_list = new int[100];`

опишіть глобальні змінні: `Random rand = new Random();` N – кількість елементів послідовності; K —кількість парних чисел, що мають парні індекси; i — індекс поточного елемента масиву.

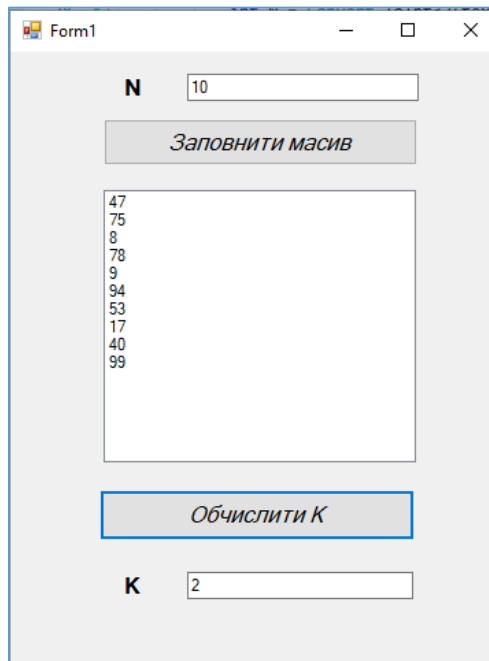
3) Створіть процедуру обробки події Click для кнопки «Заповнити масив». Запишіть оператори для введення значення N і заповнення елемента ListBox випадковими числами:

```
int N = Convert.ToInt32(texB1.Text);
for (int i=0; i < N; i++)
{
    Num_list[i] = rand.Next(100);
    listB1.Items.Add(Convert.ToString(Num_list[i]));
}
```

4) Створіть процедуру обробки події OnClick для кнопки «Обчислити K». У циклі перебираємо елементи ListBox; якщо `listB1.Items[i]` відповідає умові, збільшуємо значення K на 1:

```
int k = 0;
for (int i = 0; i < listB1.Items.Count; i++)
{
    int temp = Convert.ToInt32(listB1.Items[i]);
    if ((temp % 2 == 0) && (i % 2 == 0))
    {
        k = k + 1;
    }
}
```

Перевірте роботу програми для різних значень N. Поясніть результати роботи програми, наведені на рис. 2.2



The screenshot shows a Windows application window titled "Form1". Inside the window, there is a text box labeled "N" containing the number "10". Below it is a button labeled "Заповнити масив". Under the button is a list box containing the following numbers: 47, 75, 8, 78, 9, 94, 53, 17, 40, and 99. Below the list box is a button labeled "Обчислити K". At the bottom of the window is a text box labeled "K" containing the number "2".

Рисунок 2.2 – Результат роботи програми до завдання 1

Завдання 2. Створити проект для розв'язання задачі.

Дано послідовність із N цілих чисел ($N \leq 100$). Визначити кількість непарних чисел, які мають непарні індекси.

```
namespace Lab4
{
    public partial class Form1 : Form
    {
        int[] Num_list = new int[100];
        Random rand = new Random();

        public Form1()
        {
            InitializeComponent();
            listB1.Items.Clear();
        }

        private void bT1_Click(object sender, EventArgs e)
        {
            int N = Convert.ToInt32(texB1.Text);
            for (int i=0; i < N; i++)
            {
                Num_list[i] = rand.Next(100);
                listB1.Items.Add(Convert.ToString(Num_list[i]));
            }
        }

        private void bT2_Click(object sender, EventArgs e)
        {
            int k = 0;
            for (int i = 0; i < listB1.Items.Count; i++)
            {
                int temp = Convert.ToInt32(listB1.Items[i]);
                if ((temp % 2 == 0) && (i % 2 == 0))
                {
                    k = k + 1;
                }
            }

            textB2.Text = Convert.ToString(k);
        }
    }
}
```

Рисунок 2.3 – Код програми для реалізації завдання 1

3. Контрольні запитання

1. Для чого використовується елемент ListBox ?
2. Як ввести та вивести дані через ListBox ?
3. Наведіть властивості елемента ListBox. Як їх можна попередньо встановити та змінити під час роботи програми?
4. Як додавати, видаляти, очищати елементи ListBox?

5. Як визначити кількість елементів в ListBox?
6. Які існують події при роботі з ListBox? Як їх налаштувати?

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 Win-dows 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. С# Tutorial URL <https://www.theengineeringprojects.com>.
9. Уроки С#. URL: <https://itproger.com/course/csharp>.
10. Полное руководство по С# 8 и .NET Core. URL: <https://metanit.com/sharp/>