

Лабораторна робота №11. Проект MyUtilites. Робота з елементом NumericUpDown. Генератор

Тема: Windows Forms. Проект MyUtilites. Робота з елементом NumericUpDown.

Мета заняття: навчитись працювати з елементом NumericUpDown для створення діапазону значень генератора випадкових чисел, напрацювати навички з обробником подій Click елементів Button, checkBox та виводом інформації у елемент TextBox з допомогою Windows Forms мовою C# у середовищі Microsoft Visual Studio.

1. Теоретичні відомості

Елемент **NumericUpDown** надає користувачу вибір числа з певного діапазону. Для визначення діапазону чисел NumericUpDown має дві властивості: **Minimum** (задає мінімальне число) і **Maximum** (задає максимальне число). Саме значення елемента зберігається у властивості **Value** :

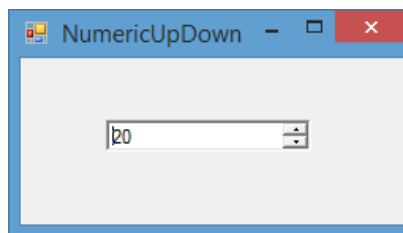


Рисунок 1 – Властивість Value елемента NumericUpDown

За замовчуванням елемент відображає десяткові числа. Однак якщо встановимо його властивість **Hexadecimal** рівним true, то елемент буде відображати всі числа в шістнадцятковій системі. Навіть якщо в коді встановимо десяткове значення, то елемент все одно відобразить його в шістнадцятковій системі.

```
numericUpDown1.Value = 66;
```

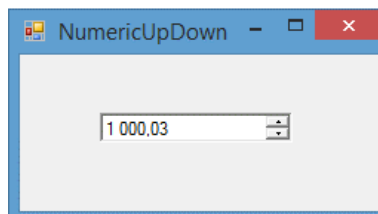


Рисунок 1.2 – Властивості **DecimalPlaces** та **ThousandsSeparator** елемента NumericUpDown

Якщо необхідно відображати в поле дробові числа, то можна використовувати властивість **DecimalPlaces**, яка вказує, скільки знаків після коми повинно відображатися. За замовчуванням ця властивість дорівнює нулю.

Також можна задати відображення роздільника тисяч. Для цього властивість **ThousandsSeparator** необхідно встановити у значення true. Наприклад, при коді елемент **NumericUpDown** буде мати вигляд (рис. 1.2):

```
numericUpDown при Value=1000,03,  
DecimalPlaces=2  
ThousandsSeparator=true:
```

За замовчуванням при натисканні на стрілочки вгору-вниз на елементі значення буде збільшуватися, або зменшуватися на одиницю. Але за допомогою властивості **Increment** можна задати інший крок збільшення, в тому числі і дробовий.

При роботі з **NumericUpDown** слід враховувати, що його властивість **Value** (як і властивості **Minimum** і **Maximum**) зберігає значення **decimal**. Тому в коді також повинні з ним працювати як з **decimal**, а не як з типом **int** або **double**.

Елемент **DomainUpDown** призначений для введення текстової інформації. Він має текстове поле для введення рядка і дві стрілки для переміщення за списком рядків (рис. 1.3)

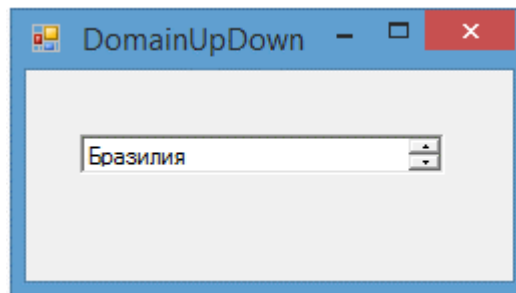


Рисунок 1.3 – Елемент **DomainUpDown**

Список для **DomainUpDown** задається за допомогою властивості **Items**. Список можна відразу впорядкувати за алфавітом. Для цього треба властивості **Sorted** привласнити значення true.

Щоб можна було циклічно переміщатися по списку, тобто при досягненні кінця або початку списку його перегляд починався з першого або останнього елемента, треба встановити для властивості **Wrap** значення true.

У коді вибране значення в **DomainUpDown** доступно через властивість **Text**. Наприклад, додамо програмно список рядків в **DomainUpDown** і опрацюємо зміну вибору в списку:

```
public partial class Form1 : Form  
{  
    public Form1()  
    {  
        InitializeComponent();  
  
        List<string> states = new List<string>  
        {  
            "Аргентина", "Бразилія", "Венесуела", "Колумбія", "Чили"  
        };  
    }  
};
```

```

        // додаємо список елементів
        domainUpDown1.Items.AddRange(states);
        domainUpDown1.TextChanged += domainUpDown1_TextChanged;
    }
    // обробка зміни тексту в елементі
    void domainUpDown1_TextChanged(object sender, EventArgs e)
    {
        MessageBox.Show(domainUpDown1.Text);
    }
}

```

Для обробки зміни тексту можна використовувати подію TextChanged, в обробнику якого виводимо обраний текст в повідомлення.

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.

1. Запускаємо Visual Studio. Відкриваємо проект MyUtilites (продовження проекту, створеного у лабораторній роботі 10).

2. Вибираємо властивість TabPages «Collection». У вікні, що відкрилось, міняємо властивість Text у tabPage2 на «Генератор» (рис. 2.1).

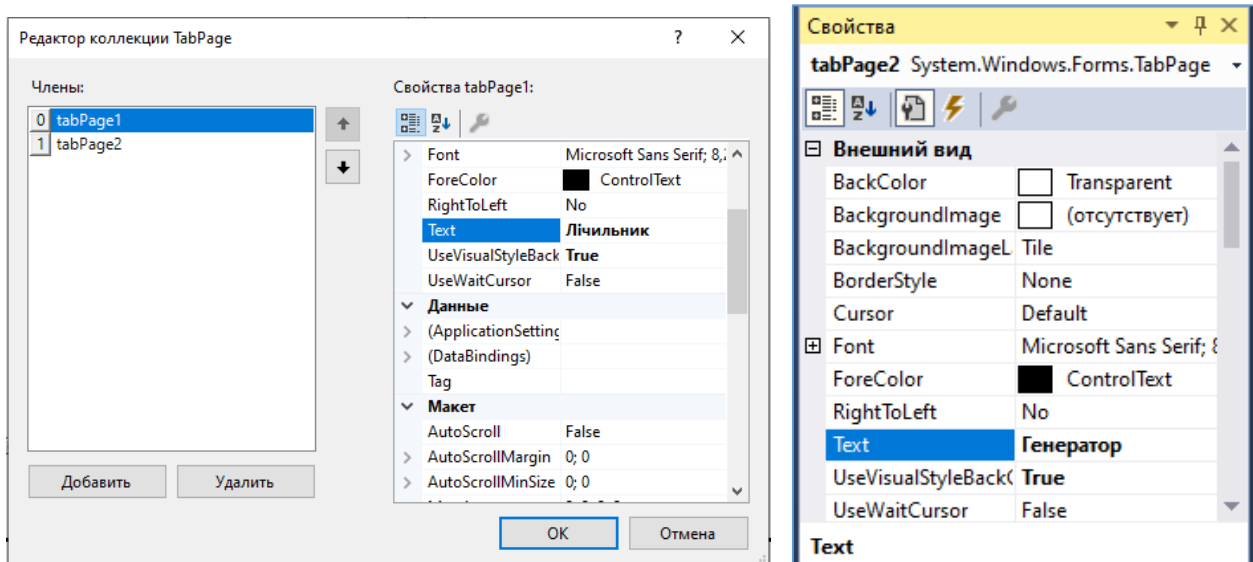


Рисунок 2.1 – Переименовування властивості Text у tabPage2

3. До проекту додаємо кнопку, 3 мітки та 2 елемента NumericUpDown (рис. 2.2).

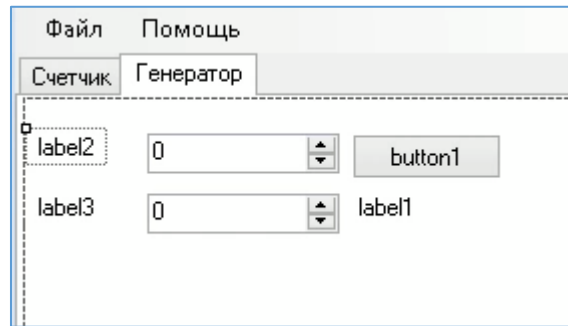


Рисунок 2.2 – Проектування дизайну MyUtilites. Вкладка «Генератор»

4. Перейменувати елементи (властивість «Text») відповідно до рис. 2.3. Властивість Name кнопки змінити на – btnRandom, Name мітки Label1 на lblRandom. Задати величину за замовчуванням для елементів NumericUpDown значення Value – 1 та 6.

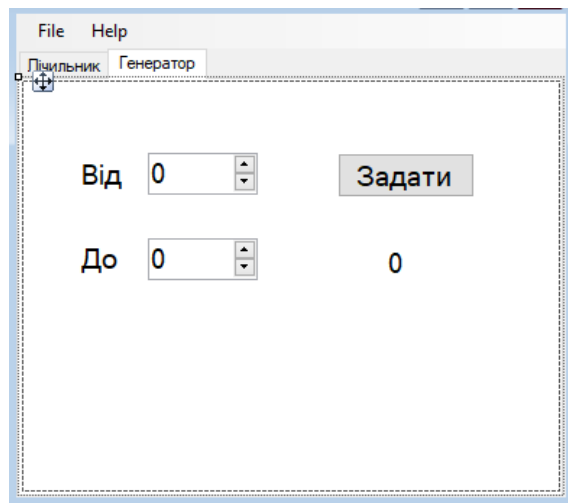


Рисунок 2.3 – Дизайн додатку MyUtilites. Вкладка «Генератор»

5. Створюємо елемент rnd класу Random (рис. 2.4).

```
public partial class MainForm : Form
{
    int count=0;
    Random rnd;

    public MainForm()
    {
        InitializeComponent();
        rnd = new Random();
    }
}
```

Рисунок 2.4 – Створення елементу rnd класу Random

6. Додаємо обробник подій для кнопки «Задати» (рис. 2.5)

```
private void btnRandom_Click(object sender, EventArgs e)
{
    int n;
    n = rnd.Next(Convert.ToInt32(numericUpDown1.Value), Convert.ToInt32(numericUpDown2.Value)+1);
    lblRandom.Text = n.ToString();
}
```

Рисунок 2.5 – Код обробника події Click кнопки «Задати»

8. Додаємо на форму елемент TextBox та задаємо властивість Name tbRandom. Встановлюємо режим MultiLine та властивість ScrollBars - Vertical (рис. 2.6)

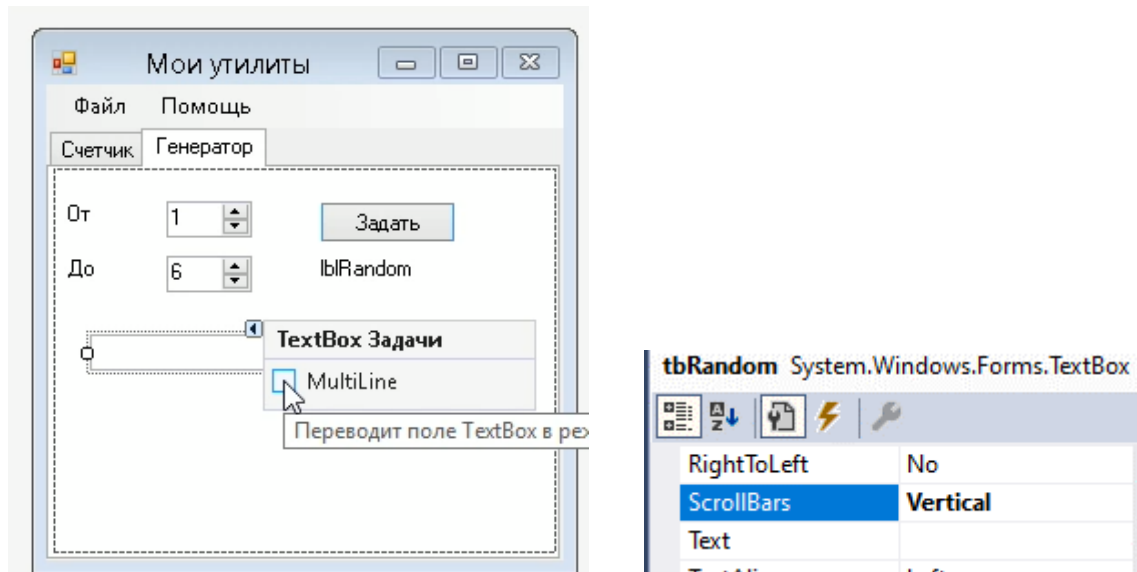


Рисунок 2.6 – Налаштування властивостей елемента TextBox

9. Додаємо код виводу в TextBox» (рис. 2.7).

```
private void btnRandom_Click(object sender, EventArgs e)
{
    int n;
    n = rnd.Next(Convert.ToInt32(numericUpDown1.Value), Convert.ToInt32(numericUpDown2.Value)+1);
    lblRandom.Text = n.ToString();
    tbRandom.AppendText(n + "\r\n");
}
```

Рисунок 2.7 – Код обробника події Click кнопки btnRandom

10. Додаємо кнопку «Очистити» (рис. 2.9). Name – btnRandomClear, Text – «Очистити». Створюємо обробник події (рис. 2.8)

```
private void btnRandomClear_Click(object sender, EventArgs e)
{
    tbRandom.Clear();
}
```

Рисунок 2.8 – Код обробника події Click кнопки «Очистити»

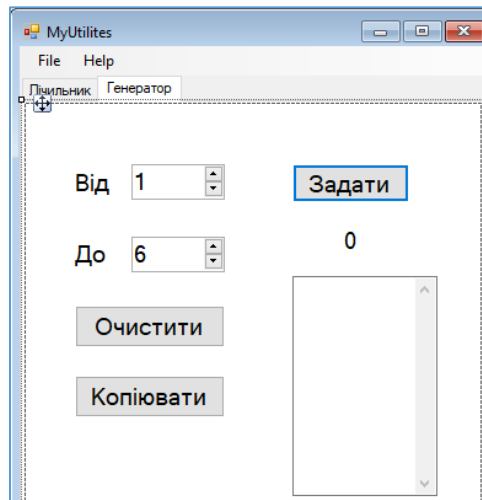


Рисунок 2.9 – Оновлений дизайн додатку MyUtilites. Вкладка «Генератор»

11. Додаємо кнопку копіювання в буфер обміну (рис. 2.9). Name – btnRandomCopy, Text – «Копіювати». Створюємо обробник події (рис. 2.10)

```
private void btnRandomCopy_Click(object sender, EventArgs e)
{
    Clipboard.SetText(tbRandom.Text);
}
```

Рисунок 2.10 – Код обробника події Click кнопки «Копіювати»

12. Організуємо додавання чисел до textBox, які раніше не зустрічались. Для цього в обробник події кнопки btnRandom дописуємо код (рис. 2.11).

```
private void btnRandom_Click(object sender, EventArgs e)
{
    int n;
    n = rnd.Next(Convert.ToInt32(numericUpDown1.Value), Convert.ToInt32(numericUpDown2.Value)+1);
    lblRandom.Text = n.ToString();
    if (tbRandom.Text.IndexOf(n.ToString())==-1)
        tbRandom.AppendText(n + "\r\n");
}
```

Рисунок 2.11 – Оновлений код обробника події Click кнопки btnRandom

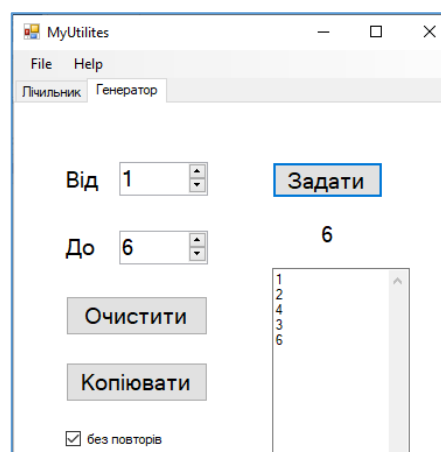


Рисунок 2.12 – Додавання елементу checkBox до дизайну додатку MyUtilites.

13. Додаємо елемент checkBox (рис. 2.12), який буде визначати додавати з повторами значення або ні до textBox. Властивість Text перейменуємо «Без повторів», Name в chRandom. Змінюємо код обробника події Click кнопки btnRandom (рис. 2.13)

```
private void btnRandom_Click(object sender, EventArgs e)
{
    int n;
    n = rnd.Next(Convert.ToInt32(numericUpDown1.Value), Convert.ToInt32(numericUpDown2.Value)+1);
    lblRandom.Text = n.ToString();
    if (cbRandom.Checked)
    {
        while (tbRandom.Text.IndexOf(n.ToString()) != -1)
            n = rnd.Next(Convert.ToInt32(numericUpDown1.Value), Convert.ToInt32(numericUpDown2.Value) + 1);
        tbRandom.AppendText(n + "\r\n");
    }
    else tbRandom.AppendText(n + "\r\n");
}
```

Рисунок 2.13 – Код обробника події Click кнопки btnRandom, що враховує стан елементу checkBox

14. Змінюємо код обробника події Click кнопки btnRandom. Якщо числа усі вибрані, то більше нічого не додаємо. Для цього додаємо змінну i, яка підраховує кількість натиснень кнопки. Якщо більше 1000, то виходимо з циклу while, якщо ні то пробуємо додати число (рис. 2.14).

```
private void btnRandom_Click(object sender, EventArgs e)
{
    int n;
    n = rnd.Next(Convert.ToInt32(numericUpDown1.Value), Convert.ToInt32(numericUpDown2.Value)+1);
    lblRandom.Text = n.ToString();
    if (cbRandom.Checked)
    {
        int i = 0;
        while (tbRandom.Text.IndexOf(n.ToString()) != -1)
        {
            n = rnd.Next(Convert.ToInt32(numericUpDown1.Value), Convert.ToInt32(numericUpDown2.Value) + 1);
            i++;
            if (i > 1000) break;
        }
        if (i < 1000) tbRandom.AppendText(n + "\r\n");
    }
    else tbRandom.AppendText(n + "\r\n");
}
```

Рисунок 2.14 – Оптимізований код обробника події Click кнопки btnRandom

15. Перевірте роботу елементів меню додатку MyUtilites (Ctrl+F5). Оформіть звіт.

3. Контрольні запитання

1. Призначення елементів NumericUpDown та DomainUpDown.
2. Як задати мінімальне та максимальне число елементу NumericUpDown?
3. Яке призначення властивостей Hexadecimal, DecimalPlaces та ThousandsSeparator?
4. Яке призначення властивості Increment елементу NumericUpDown?
5. Як задати діапазон значень елементу DomainUpDown?
6. Як задати циклічне перебирання значень елементу DomainUpDown?
7. Як створити випадкове число в діапазоні значень?

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. 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/>
11. С#. Мини-программы. URL: <https://geekbrains.ru/chapters/972>