

Лабораторна робота №5. Windows Forms. Елемент CheckBox

Тема: Елемент CheckBox.

Мета заняття: навчитись працювати з елементом RadioButton, CheckBox у середовищі Microsoft Visual Studio; навчитись використовувати події (events) при роботі з елементом RadioButton, CheckBox з допомогою Windows Forms мовою C#.

1. Теоретичні відомості

1.1 Елемент RadioButton

C # RadioButton використовують як перемикач для отримання конкретних вхідних значень. RadioButton дозволяє користувачеві одночасно перевіряти лише одну опцію, а інші параметри залишаються неперевіреними або невибраними. Для додавання RadioButton необхідно знайти RadioButton у ToolBox і перетягнути його до своєї програми. На рис. 1.1 додано 8 RadioButtons та вибрано лише одну, тому що не можна вибрати більше однієї RadioButton одночасно.

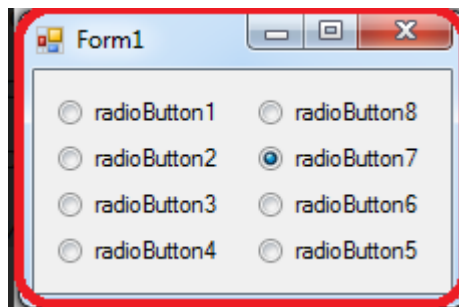


Рисунок 1.1 – Робота з RadioButtons у Windows Forms

Назва перемикачів - це імена за замовчуванням. Для зміни їх назви є два варіанти. Перший варіант - змінити властивість Text у розділі властивості (рис. 1.2).

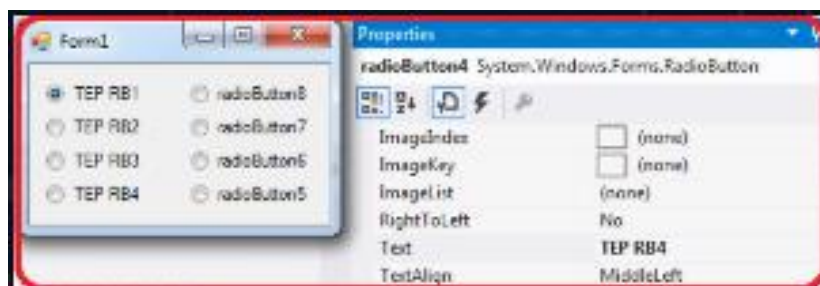


Рисунок 1.2 – Налаштування властивості RadioButton.Text

Другий спосіб зміни назви радіо кнопки - це програмно. У вихідному коді потрібно встановити ім'я перемикача за допомогою властивості «Text» перемикача. У коді, що наведений нижче, показано, як встановити значення властивості тексту перемикача.

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

namespace TEP
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            radioButton5.Text = "TEP Code RB5";
            radioButton6.Text = "TEP Code RB6";
            radioButton7.Text = "TEP Code RB7";
            radioButton8.Text = "TEP Code RB8";
        }
    }
}
```

Якщо потрібно змінити стан RadioButton, то це дозволяє властивість Checked. За замовчуванням кожна RadioButton не встановлена, тому можна змінити значення властивості Checked на true, як показано у коді нижче:

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

namespace TEP
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            radioButton1.Checked = true;
        }
    }
}
```

Властивість BackColor дозволяє встановити колір тла. Існують різновиди кольору, які можна використовувати для встановлення кольору тла. У коді, що наведений нижче, використовуються різні кольори для кожної кнопки (рис. 1.3).

```
using System;
using System.Drawing;
```

```

using System.Windows.Forms;

namespace strnull
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            radioButton1.BackColor = Color.AliceBlue;
            radioButton2.BackColor = Color.AntiqueWhite;
            radioButton3.BackColor = Color.Aqua;
            radioButton4.BackColor = Color.Aquamarine;
            radioButton5.BackColor = Color.Azure;
            radioButton6.BackColor = Color.Beige;
            radioButton7.BackColor = Color.Bisque;
            radioButton8.BackColor = Color.BlueViolet;
        }
    }
}

```

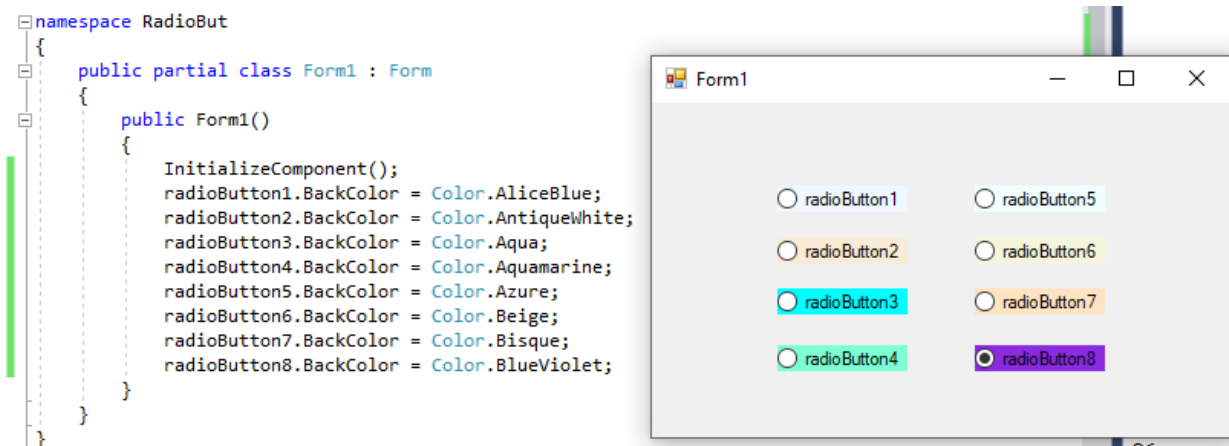


Рисунок 1.3 – Демонстраційний приклад роботи з властивістю BackColor елемента RadioButton



Рисунок 1.4 – Демонстраційний приклад роботи з властивістю ForeColor елемента RadioButton

Властивість `ForeColor` дозволяє змінити колір тексту (рис. 1.4). Властивість `Font` дозволяє зміни розмір шрифту `RadioButton` (рис. 1.5). Щоб встановити розмір необхідно передати прототип сімейства шрифтів та розмір шрифту як параметр у конструкторі шрифтів як показано нижче:.

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

namespace TEP
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            radioButton1.Font = new Font(Font.FontFamily, 10);
            radioButton2.Font = new Font(Font.FontFamily, 11);
            radioButton3.Font = new Font(Font.FontFamily, 12);
            radioButton4.Font = new Font(Font.FontFamily, 13);
            radioButton5.Font = new Font(Font.FontFamily, 12);
            radioButton6.Font = new Font(Font.FontFamily, 11);
            radioButton7.Font = new Font(Font.FontFamily, 10);
            radioButton8.Font = new Font(Font.FontFamily, 9);
        }
    }
}
```

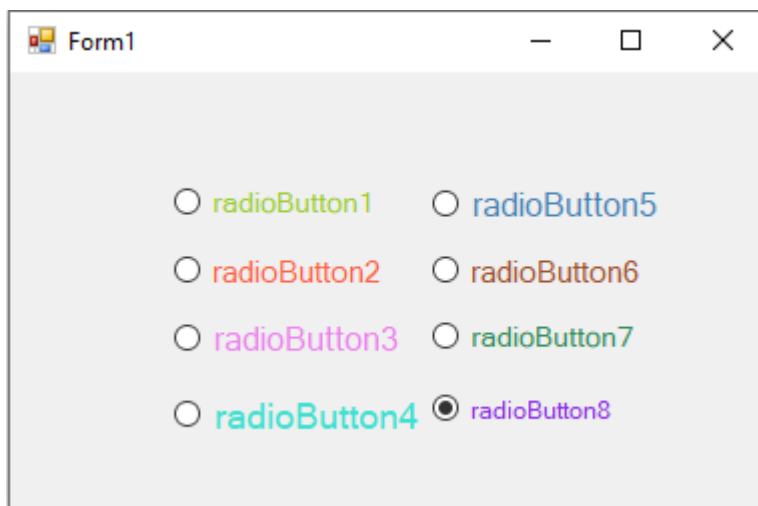


Рисунок 1.5 – Демонстрація зміни розміру шрифту `RadioButton`(властивість `Font`)

Для зміни типу шрифту використовують властивість `Font` у якій потрібно задати ім'я шрифту та розмір тексту (рис. 1.6).

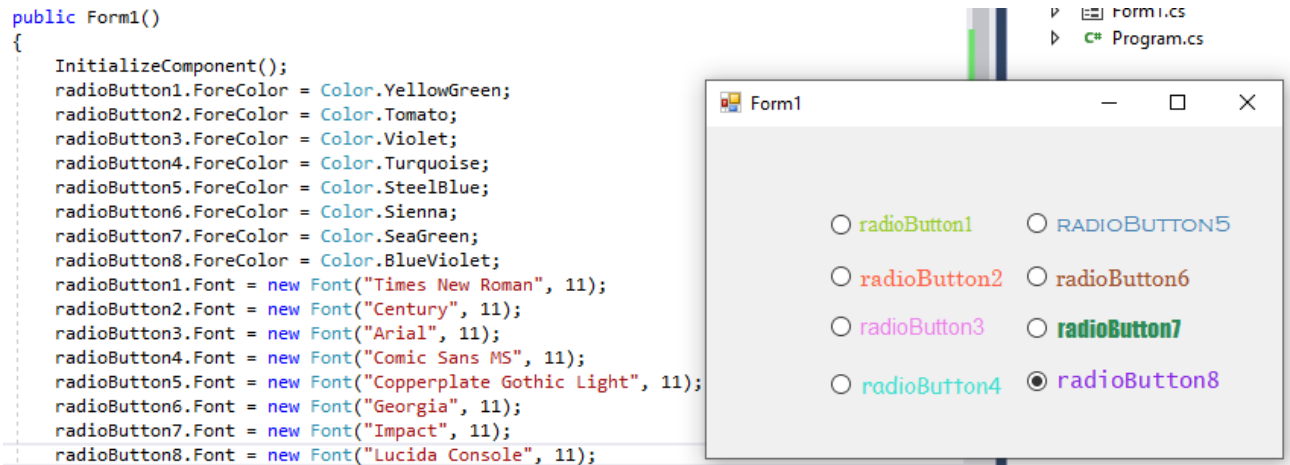


Рисунок 1.6 – Демонстрація зміни імені шрифту та розміру тексту RadioButton (властивість Font)

Властивість Checked використовується для визначення стану radioButton. Умовний оператор *if* використовується, щоб створити логіку роботи перемикача. У демонстраційному коді кожна RadioButton містить логічне твердження, яке повертає значення у вікні повідомлення, якщо воно встановлено (рис. 1.7).

```

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

namespace TEP
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            if (radioButton1.Checked == true)
            {
                MessageBox.Show(radioButton1.Text);
            }
            else if (radioButton2.Checked == true)
            {
                MessageBox.Show(radioButton2.Text);
            }
            else if (radioButton3.Checked == true)
            {
                MessageBox.Show(radioButton3.Text);
            }
            else if (radioButton4.Checked == true)
            {
                MessageBox.Show(radioButton4.Text);
            }
        }
    }
}

```

```

    }
    else if (radioButton5.Checked == true)
    {
        MessageBox.Show(radioButton5.Text);
    }
    else if (radioButton6.Checked == true)
    {
        MessageBox.Show(radioButton6.Text);
    }
    else if (radioButton7.Checked == true)
    {
        MessageBox.Show(radioButton7.Text);
    }
    else if (radioButton8.Checked == true)
    {
        MessageBox.Show(radioButton8.Text);
    }
}
}
}

```

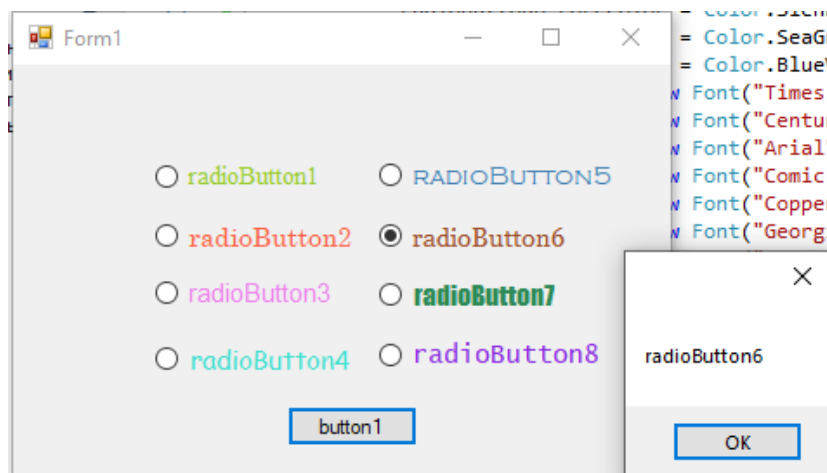


Рисунок 1.7 – Демонстрація застосування властивості Checked

Властивість Image використовується для встановлення зображення як фон для перемикача. У наступному коді показаний приклад, як встановити зображення як фон.

```
radioButton1.Image = Image.FromFile("C:\\Pictures\\brownImage.jpg");
```

Для роботи з елементом RadioButton використовують такі події користувача:

- RadioButton BackColorChanged
- RadioButton CheckChanged
- RadioButton Click
- RadioButton ForeColorChanged

RadioButton MouseHover
RadioButton MouseLeave
RadioButton TextChanged

Подія `RadioButton BackColorChanged` відбувається лише при зміні кольору фону перемикача. Припустимо, що потрібно показати повідомлення або виконати будь-яку функціональність, коли колір фону перемикача змінився, тоді використовується `BackColorChanged` Event. Є два способи за допомогою яких можна встановити цю подію для `RadioButton`. Перший спосіб - активувати цю подію з розділу властивостей перемикача через події. Другий метод полягає у створенні визначеної користувачем функції та встановленні як подія `BackColorChanged` у властивостях події (рис. 1.8)

Перший метод

Другий метод

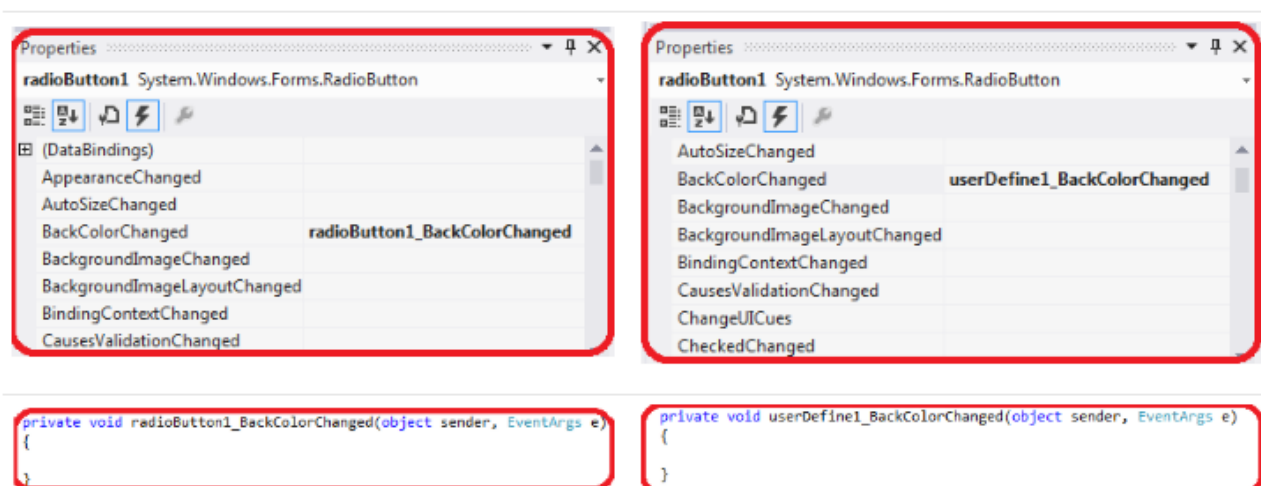


Рисунок 1.8 – Встановлення події `BackColorChanged` для `RadioButton`

Подія `CheckedChanged` відбувається коли користувач змінить стан `RadioButton` (checked/unchecked). Наприклад, користувач вибирає `RadioButton A`, а потім вибирає `RadioButton B`. Тоді буде викликана подія `CheckedChanged`.

Подія `Click` відбудеться, коли користувач натисне на `RadioButton`. Подія `ForeColorChange` виконується, коли користувач змінить колір `RadioButton`.

Подія `MouseHover` виконується щоразу, коли користувач наводить курсор миші на перемикач. Якщо навести курсор миші на текст перемикача, то подія також виконується, тому що текст також входить до меж `RadioButton`. Продемонструємо вищевказаний сценарій.

```
private void radioButton1_MouseHover(object sender, EventArgs e)
{
    MessageBox.Show("Mouse Hover to " + radioButton1.Text.ToString());
}
```

Подія `MouseLeave` відбувається, коли курсор миші залишить межі перемикача, що протилежне події `mouseHover`. Продемонструємо цей сценарій у наступному коді разом із подією `MouseHover`. Так що можна спостерігати обидві події одночасно.

```
private void radioButton1_MouseHover(object sender, EventArgs e)
{
    MessageBox.Show("Mouse Hover to " + radioButton1.Text.ToString());
}

private void radioButton1_MouseLeave(object sender, EventArgs e)
{
    MessageBox.Show("Mouse Left " + radioButton1.Text.ToString());
}
```

Подія `TextChanged` буде виконана, коли користувач змінить текст. Щоб спрацювала така подія, створюємо подію натискання кнопки, в якій змінюється текст `RadioButton`. У наступному коді демонструється запропонований сценарій.

```
private void button1_Click(object sender, EventArgs e)
{
    radioButton1.Text = "New Text";
}

private void radioButton1_TextChanged(object sender, EventArgs e)
{
    MessageBox.Show("Radio Button Text changed to " + radioButton1.Text.ToString());
}
```

1.2 Елемент *CheckBox*

`CheckBox` - це прапорець разом із текстом, який є назвою прапорця. Він дозволяє користувачу зробити множинний вибір. Якщо користувач перший раз натисне на прапорець, то на позначці прапорець буде позначено галочку. Коли користувач знову натисне прапорець, він знімає позначку.

Для додавання `CheckBox` у свій додаток необхідно перетягнути його з панелі інструментів на робочий стіл програми. На рис. 1.9 показана форма з 15 `CheckBox` елементами.

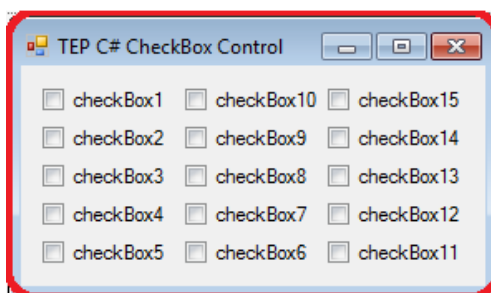


Рисунок 1.9 – Вікно проекту з елементами `CheckBox`

Є два варіанти змінити назву CheckBox. Перший спосіб полягає в тому, щоб змінити властивість «Text» на вкладці «Properties» Visual Studio (рис. 1.10).

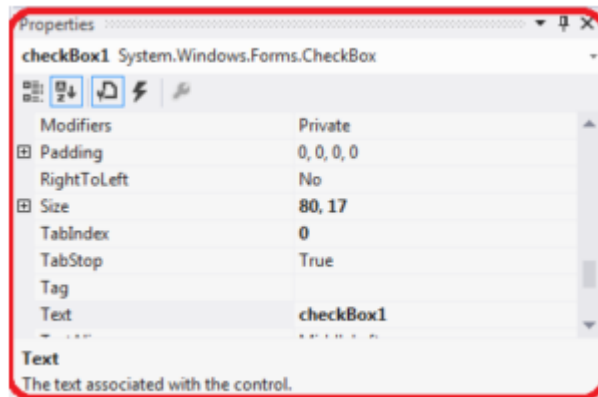


Рисунок 1.10 –Властивість «Text» вкладки «Properties»

Другий метод - це зміна назви програмно, як показано нижче:

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
        checkBox1.Text = "ТЕР CheckBox1";
        checkBox2.Text = "ТЕР CheckBox2";
    }
}
```

Колір тексту, розмір тексту, сімейство текстових шрифтів можемо використовувати для налаштування CheckBox. Так можемо використовувати властивість Checked, щоб задати початковий стан CheckBox

```
public Form1()
{
    InitializeComponent();
    checkBox1.Checked = true;
    checkBox2.Checked = true;
    checkBox3.Checked = true;
    checkBox4.BackColor = Color.BurlyWood;
    checkBox5.BackColor = Color.DarkBlue;
    checkBox6.BackColor = Color.DarkOrange;
    checkBox7.ForeColor = Color.DeepSkyBlue;
    checkBox8.ForeColor = Color.Gainsboro;
    checkBox9.ForeColor = Color.LawnGreen;
    checkBox6.Font = new Font("Georgia", 11);
    checkBox7.Font = new Font("Impact", 11);
    checkBox8.Font = new Font("Lucida Console", 11);
}
```

Для перевірки стану CheckBox застосовують умовні оператори, як у наступного коді (рис. 1.11):

```
private void button1_Click(object sender, EventArgs e)
{
    if (checkBox1.Checked == true)
    {
        MessageBox.Show(checkBox1.Text);
    }
    if (checkBox2.Checked == true)
    {
        MessageBox.Show(checkBox2.Text);
    }
    if (checkBox3.Checked == true)
    {
        MessageBox.Show(checkBox3.Text);
    }
}
```

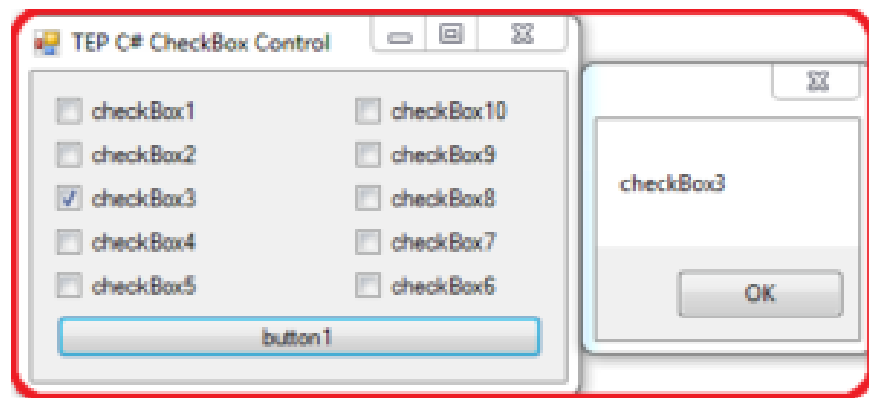


Рисунок 1.11 – Демонстрація фрагменту коду з перевіркою стану CheckBox

Для роботи з елементом CheckBox використовують такі події користувача:

- RadioButton BackColorChanged
- RadioButton CheckedException
- RadioButton Click
- RadioButton ForeColorChanged
- RadioButton MouseHover
- RadioButton MouseLeave
- RadioButton TextChanged

Особливості роботи з подіями CheckBox аналогічні як і для подій елемента RadioButton, які були вже розглянуті

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.

Скласти програму, в якій реалізовано головоломку Лойда: із заданого набору чисел вибрати ті, сума яких дорівнює 50.

1. Додаємо на форму елемент *CheckedListBox*. Властивість *Item* змінюємо, щоб додати назви *CheckBox* (рис. 2.1). Властивість *CheckOnClick* встановлюємо в *True* (переключалось одним кліком миши, а не двома).

2. Додаємо на форму елементи *Label1*, *Label2*, *Button1*, *Button2* (рис. 2.1). Елементу *Label2* задайте значення *label2.Text = ""* та оголосіть глобальну змінну *int sum*.

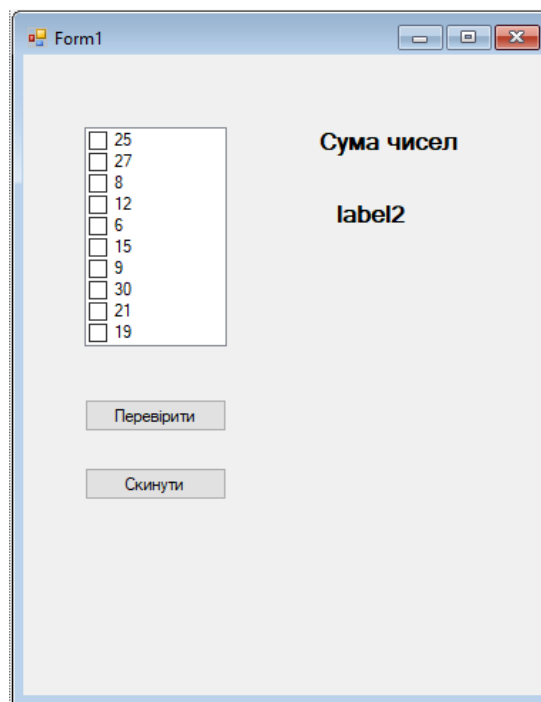


Рисунок 2.1 – Зовнішній вигляд форми для реалізації головоломки Лойда

3. Створити подію *Click* для *Button1*, у якій виконується додавання значень *CheckBox* з *CheckedListBox*, що встановлені. Результат виводиться до *Label2*.

```
private void button1_Click(object sender, EventArgs e)
{
    sum = 0;
    for (int i = 0; i < checkedListBox1.Items.Count; i++)
        if (checkedListBox1.GetItemChecked(i))
        {
```

```

        sum += int.Parse(checkedListBox1.Items[i].ToString());
    }
    label2.Text = Convert.ToString(sum);
}

```

4. Перевірити роботу програму та переконатися, що виконується обробка значень ChexBox

5. Додати обробник події Click для Button 2 у якому скидаються встановлені ChexBox елементи та результат.

```

private void button2_Click(object sender, EventArgs e)
{
    for (int i = 0; i < checkedListBox1.Items.Count; i++)
        if (checkedListBox1.GetItemChecked(i))
        {
            checkedListBox1.SetItemChecked(i, false);
        }
    label2.Text = "";
}

```

6. Запустіть проект. Спробуйте розв'язати головоломку.

```

1  using System;
2  using System.Windows.Forms;
3
4  namespace ChexBox
5  {
6      public partial class Form1 : Form
7      {
8          int sum;
9
10         public Form1()
11         {
12             InitializeComponent();
13             label2.Text = "";
14         }
15
16         private void button1_Click(object sender, EventArgs e)
17         {
18             sum = 0;
19             for (int i = 0; i < checkedListBox1.Items.Count; i++)
20                 if (checkedListBox1.GetItemChecked(i))
21                 {
22                     sum += int.Parse(checkedListBox1.Items[i].ToString());
23                 }
24             label2.Text = Convert.ToString(sum);
25         }
26
27         private void button2_Click(object sender, EventArgs e)
28         {
29             for (int i = 0; i < checkedListBox1.Items.Count; i++)
30                 if (checkedListBox1.GetItemChecked(i))
31                 {
32                     checkedListBox1.SetItemChecked(i, false);
33                 }
34             label2.Text = "";
35         }
36     }
37 }
38
39
40

```

Рисунок 2.2 – Код програми, що реалізовує головоломку Лойда

Завдання 2. Створити проект для розв'язання задачі.

Скласти програму, в якій виконується додавання значень елементів CheckBox.

1. Додаємо на форму три елемента CheckBox. Властивість Text елементів checkBox встановити у відповідності з рис. 2.3. Додати на форму елемент Label.

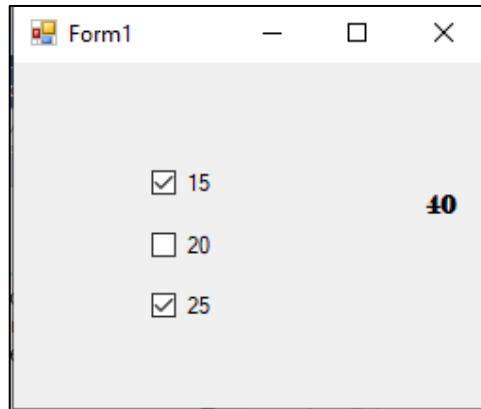


Рисунок 2.3 –Зовнішній вигляд форми для реалізації завдання 2.

2. Додати для кожного елемента обробник події CheckedChanged за яким додаються значення вибраних значень CheckBox. Лістинг програми, що реалізовує запропонований сценарій, наведений нижче:

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

namespace Test_Check
{
    public partial class Form1 : Form
    {
        int sum=0;
        int ch1, ch2, ch3;

        public Form1()
        {
            InitializeComponent();
            label1.Text = "";
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked) {
                ch1 = int.Parse(checkBox1.Text);
            }
            else
            {
                ch1=0;
            }

            sum = ch1+ch2+ch3;
            label1.Text = sum.ToString();
        }

        private void checkBox2_CheckedChanged(object sender, EventArgs e)
```

```

    {
        if (checkBox2.Checked)
        {
            ch2 = int.Parse(checkBox2.Text);
        }
        else
        {
            ch2 = 0;
        }

        sum = ch1+ch2+ch3;
        label1.Text = sum.ToString();
    }

    private void checkBox3_CheckedChanged(object sender, EventArgs e)
    {
        if (checkBox3.Checked)
        {
            ch3 = int.Parse(checkBox3.Text);
        }
        else
        {
            ch3 = 0;
        }

        sum = ch1 + ch2+ch3;
        label1.Text = sum.ToString();
    }
}

```

3. Внести зміни до форми та програми щоб реалізувати головоломку Лойда.

3. Контрольні запитання

1. Для чого використовується елементи RadioButton, ChexBox, CheckedListBox?
2. Як перевірити стан RadioButton, ChexBox, CheckedListBox?
3. Наведіть властивості елементів RadioButton, ChexBox, CheckedListBox. Як їх можна попередньо встановити та змінити під час роботи програми?
4. Які існують події при роботі з RadioButton, ChexBox, CheckedListBox? Як їх налаштувати?
5. Як визначити кількість встановлених елементів в CheckedListBox?

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 C# 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. C#. Теорія та практика. URL: https://www.bestprog.net/uk/sitemap_ua/c-3
7. Сажин А. Справочник по языку программирования C#. 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/>