Лабораторна робота 1. Вступ у розробку форм

Вправа 1. Налаштування прямокутної форми Windows (3 бали)
Вправа 2. Створення непрямокутної форми Windows (3 бали)
Вправа 3. Створення спадковної/похідної форми (3 бали)
Вправа 4. Створення MDI-застосування (6 балів)
Додаткова вправа
Завдання 1. (2 бали)
Завдання 2. (2 бали)

Мета роботи: Вивчення методів побудови форм Windows і отримання навичок з налаштування форм, створенню непрямокутних і спадкованих (похідних) форм.

Вправа 1. Налаштування прямокутної форми Windows (3 бали)

Форми Windows - це основний компонент інтерфейсу користувача. Форми надають *контейнер*, який містить елементи управління, меню і дозволяє відображати застосування у вже звичному та однаковому стилі. Форми можуть реагувати на події миші і клавіатури, які надходять від користувача, і виводити на екран дані для користувача за допомогою елементів управління, які містяться у формі.

Форми Windows містять багато властивостей, які дозволяють налаштовувати їх зовнішній вигляд і поведінку. Переглядати і змінювати ці властивості можна у вікні **Properties** конструктора при розробці, а також програмно під час виконання застосування.

У наступній таблиці перераховані деякі властивості форм Windows, які відповідають за зовнішній вигляд і поведінку застосування:

Властивість	**Опис**
Name	Задає ім'я класу Form, показаному в конструкторі. Ця властивість задається виключно під час розробки
BackColor	Вказує колір фону форми
Enabled	Вказує, чи може форма приймати введення від користувача. Якщо властивості Enabled задано значення False, всі елементи управління форми також блокуються
ForeColor	Вказує колір переднього плану форми, тобто колір тексту, що виводиться. Якщо окремо не вказати значення властивості ForeColor елементів управління форми, вони набудуть того ж значення
FormBorderStyle	Вказує вигляд і поведінку межі і рядка заголовка форми. Значення властивості:
	None - Форма не має межі, не може бути мінімізована або розгорнута до максимальних розмірів і у неї немає екранної кнопки управління вікном і кнопки довідки
	FixedSingle - Форма має тонку межу, і розміри форми не можна змінити під час виконання. Форма може бути мінімізована, розгорнута до максимальних розмірів, і мати кнопку довідки або кнопку управління вікном, що визначається іншими властивостями
	Fixed3D - Форма має об'ємну межу, і розміри форми не можна змінити під час виконання. Форма може бути мінімізована, розгорнута до максимальних розмірів, і мати кнопку довідки або кнопку управління вікном, що визначається іншими властивостями
	FixedDialog - Форма має тонку межу, і розміри форми не можна змінити під час виконання. У форми немає екранної кнопки управління вікном, але може бути кнопка довідки, що визначається іншими властивостями. Форму можна мінімізувати і розгорнути до максимальних розмірів

	Sizable - Форма має налаштування за замовчуванням, але вони можуть змінюватися користувачем. Форма може бути мінімізована, розгорнута до максимальних розмірів, і мати кнопку довідки, що визначається іншими властивостями
	FixedToolWindow - Форма має тонку межу, і розміри форми не можна змінити під час виконання. Форма містить тільки кнопку закриття
	SizableToolWindow - Форма має тонку межу, і розміри форми можуть бути змінені користувачем. Форма містить тільки кнопку закриття
Location	Коли властивості StartPosition задано значення Manual, ця властивість вказує початкове положення форми відносно верхнього лівого кута екрану
MaximizeBox	Вказує, чи є у форми кнопка MaximizeBox
MaximumSize	Встановлює максимальний розмір форми. Якщо задати цьому властивості розмір 0; 0, у форми не буде верхнього обмеження розміру
MinimizeBox	Вказує, чи є у форми кнопка MinimizeBox
MinimumSize	Встановлює мінімальний розмір форми, який користувач може задати
Opacity	Встановлює рівень непрозорості або прозорості форми від 0 до 100%. Форма, непрозорість якої складає 100%, повністю непрозора, а форма, що має 0 % непрозорості, навпаки, повністю прозора
Size	Приймає і встановлює початковий розмір форми
StartPosition	Вказує положення форми у момент її першого виведення на екран
Text	Вказує заголовок форми
TopMost	Вказує, чи завжди форма відображається поверх усіх інших форм, властивості TopMost яких не задано значення True
Visible	Вказує, чи видима форма під час роботи
WindowState	Вказує, чи є форма мінімізованою, розгорнутою до максимальних розмірів, або ж при першій появі їй заданий розмір, вказаний у властивості Size

Створення нового проекту

1. Відкрийте Visual Studio і створіть новий проект **Windows Forms**. Проект відкриється з формою за замовчуванням з ім'ям **Form1** в конструкторі.
2. Виберіть форму в конструкторі. Властивості форми відображаються у вікні **Properties**.
3. У вікні **Properties** задайте властивостям значення, як зазначено нижче:

Властивість	**Значення**
Text	Trey Research
FormBorderStyle	Fixed3D
StartPosition	Manual
Location	100; 200
Opacity	75%

4. Перетягніть три кнопки з **Toolbox** у форму і розмістіть їх так, як вам буде зручно.
5. По черзі виберіть кожну кнопку і у вікні **Properties** задайте властивості кнопок **Text** значення **Border Style, Resize** та **Opacity**.
6. Для кнопки **Border Style** задайте властивість **Anchor - Top, Left**.

Реалізація обробників подій

7. У конструкторі двічі клацніть кнопку **Border Style**, щоб відкрити вікно з кодом обробника події **Buttonl Click**. Додайте в цей метод наступний рядок коду:

```
this.FormBorderStyle = FormBorderStyle.Sizable;
```

8. Поверніться у вікно конструктора, двічі клацніть кнопку **Resize** і додайте наступний рядок:

```
this.Size = new Size(300, 500);
```

9. Поверніться у вікно конструктора, двічі клацніть кнопку **Opacity** і додайте наступний рядок:

```
this.Opacity = 1;
```

Запуск готового рішення

10. Для побудови рішення виберіть меню **Build** (**Построение**), далі команду **Build Solution** (**Построить решение**). За наявності помилок виправте їх і знову побудуйте рішення. Надалі при потребі вибору послідовності дій черговість команд описуватиметься, наприклад, так: **Build → Build Solution**.
11. Натисніть **Ctrl+F5** або виберіть **Debug (Отладка) → Start Without Debugging (Запуск без отладки)**, щоб запустити застосування. Клацайте кожну кнопку і спостерігайте, як змінюється вигляд форми.
12. Змініть по черзі розташування лівої і верхньої меж форми і порівняйте поведінку кнопок усередині форми. Зверніть увагу, що відстань до цих меж від кнопки **Border Style** залишається сталою. Чому?

Вправа 2. Створення непрямокутної форми Windows (3 бали)

У цій вправі ви створите трикутну форму Windows.

1. Відкрийте Visual Studio і створіть новий проект **Windows Forms**. Проект відкриється з формою за замовчуванням з ім'ям **Form1** в конструкторі.
2. У вікні **Properties** задайте властивості **FormBorderStyle** значення **None**, а властивості **BackColor** значення **Red**. У цьому випадку форму легше буде побачити при тестуванні застосування.
3. Перетягніть кнопку з **Toolbox** в лівий верхній кут форми. Задайте властивості **Text** кнопки значення **Close Form**.
4. Двічі клацніть кнопку **Close Form** і додайте в обробник події **Button1_Click** наступний код:

```
this.Close();
```

5. У конструкторі двічі клацніть форму, щоб відкрити обробник події **Form1_Load**. Додайте в цей метод наступний код (він задає області форми трикутну форму зазначенням багатокутника з трьома кутами):

```
System.Drawing.Drawing2D.GraphicsPath myPath = new System.Drawing.Drawing2D.GraphicsPath();
myPath.AddPolygon(new Point[]
{
        new Point(0, 0),
        new Point(0, this.Height),
        new Point(this.Width, 0)
});
Region myRegion = new Region(myPath);
this.Region = myRegion;
```

6. Побудуйте і запустіть застосування. З'явиться трикутна форма.

Вправа 3. Створення спадковної (похідної) форми (3 бали)

Якщо у вас є вже готова форма, яку ви збираєтесь використати в декількох застосуваннях, зручно створити *спадковну* (похідну) форму. У цій вправі ви створите нову форму і успадкуєте її від існуючої базової форми, а потім зміните похідну форму, налаштувавши її для конкретної роботи.

1. Відкрийте проект з попередньої вправи. Базовою формою для створення похідної буде трикутна форма.
2. Для кнопки **Close Form** задайте властивість **Modifiers** як **protected**.
3. Додайте похідну форму: меню **Project (Проект) → Add Windows Form… (Добавить форму Windows),** у вікні **Categories** (**Категории**) вкажіть **Windows Form**, у вікні **Templates** (**Шаблоны**) виберіть **Inherited Form** (**Наследуемая (производная) форма**).
4. У вікні **Add New Item** в полі **Name** вкажіть назву форми: **nForm.cs** і натисніть **Add** для додавання форми.
5. У вікні **Inheritance Picker**, яке з'явилося і в якому відображаються усі форми поточного проекту, виберіть базову форму **Form1** і натисніть **OK**.
6. Побудуйте проект.
7. Відкрийте форму **nForm** в режимі конструктора. Перевірте, що вона має трикутну форму і властивості базової форми та елемента управління успадковані.
8. Налаштуйте властивості похідної форми:
   a. для кнопки:
      i. властивість **Text - Hello!!!**
      ii. властивість **BackColor - Brown**
   b. для форми: властивість **BackColor - Blue**
9. Побудуйте проект.
10. Задайте похідну форму як стартову, вказавши у функції Main наступний код:

```
Application.Run(new nForm());
```

Порада: У мові C# метод **Main()** — це головний метод програми, його точка входження, є частиною класу. Клас, в якому знаходиться цей метод, ***за замовчуванням*** називається **Program**. Це статичний метод, має ключове слово **static**, тому він реалізується ще до створення об’єкта класу, в якому він знаходиться. Є декілька варіантів створення методу **Main()**. Пояснення на прикладах для кращого розуміння:
Метод **Main()** має ***порожній тип*** **void**.

Main.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Main
{
        class Program
        {
                public static void Main()
                {
                        Console.WriteLine("Миру Мир!");
                        Console.ReadKey();
                }
        }
}
```

Метод **Main()** ***має тип*** **int**, відповідно має оператор **return**, який повертає операційній системі значення **0**, що означає нормальне завершення програми.

Main.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Main
{
        class Program
        {
                public static int Main()
                {
                        Console.WriteLine("Миру Мир!");
                        Console.ReadKey();
                        return 0;
                }
        }
}
```

Метод **Main()** ***містить параметри командного рядка***, тип значення, яке повертається може бути будь-яким, **int** або **void**.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Main
{
        class Program
        {
                public static int Main(string[] args)
                {
                        decimal val0 = decimal.Parse(args[0]);
                        decimal val1 = decimal.Parse(args[1]);
                        Console.WriteLine("Сума: {0}", (val0 + val1));
                        Console.ReadKey();
                        return 0;
                }
        }
}
```

Програма ***містить більше одного методу*** **Main()**. Це використовується для різних тестів. Щоб скомпілювати таке застосування, потрібно створити точку входження в програму.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace Main
{
        class Program1
        {
                public static int Main()
                {
                        Console.WriteLine("Програма запускається із класу Program1");
                        Console.ReadKey();
                        return 0;
                }
        }
        class Program2
        {
                public static int Main()
                {
                        Console.WriteLine("Програма запускається із класу Program2");
                        Console.ReadKey();
                        return 0;
                }
        }
}
```

Створіть точку входження в програму. **Project → Properties (див. рис. 1.1.).**

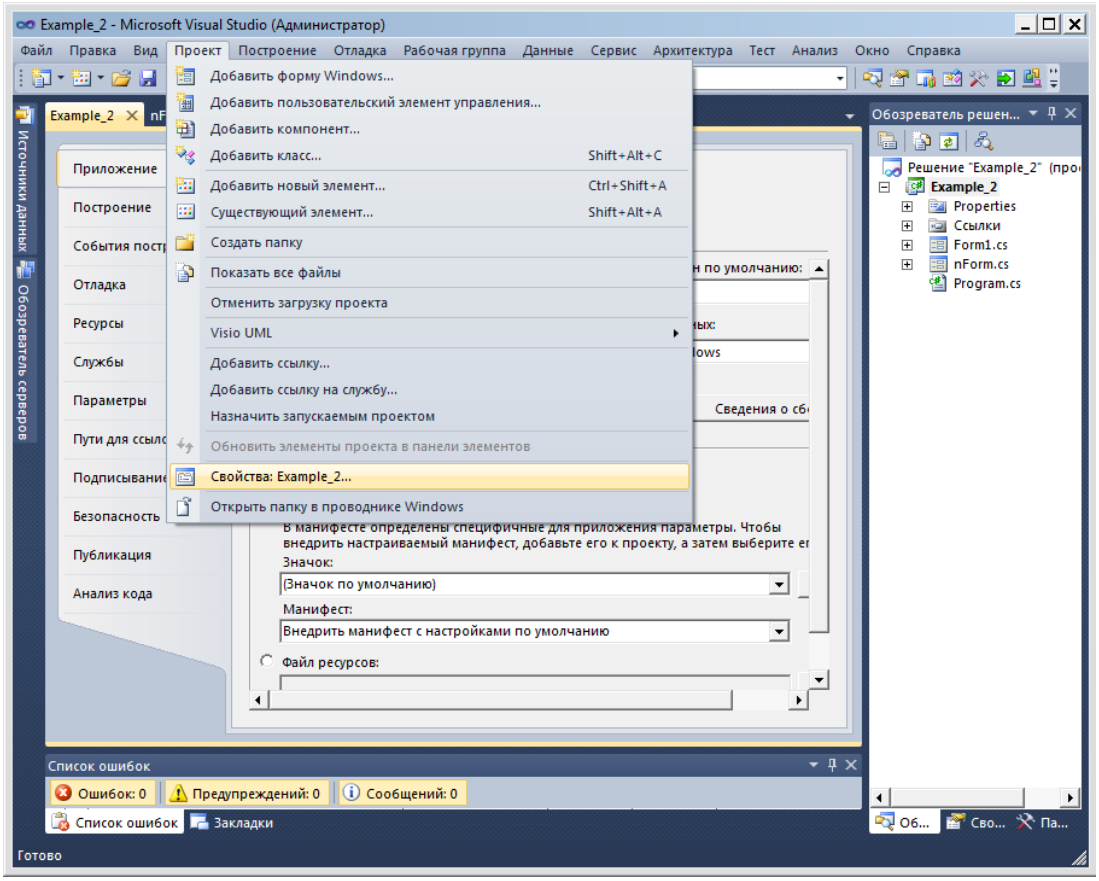

Рис. 1.1. Створення точки входження

Виберіть вкладку **Application → Startup object**. Виберіть клас **Program1 (**див. рис. 1.2**).** Збережіть файл.

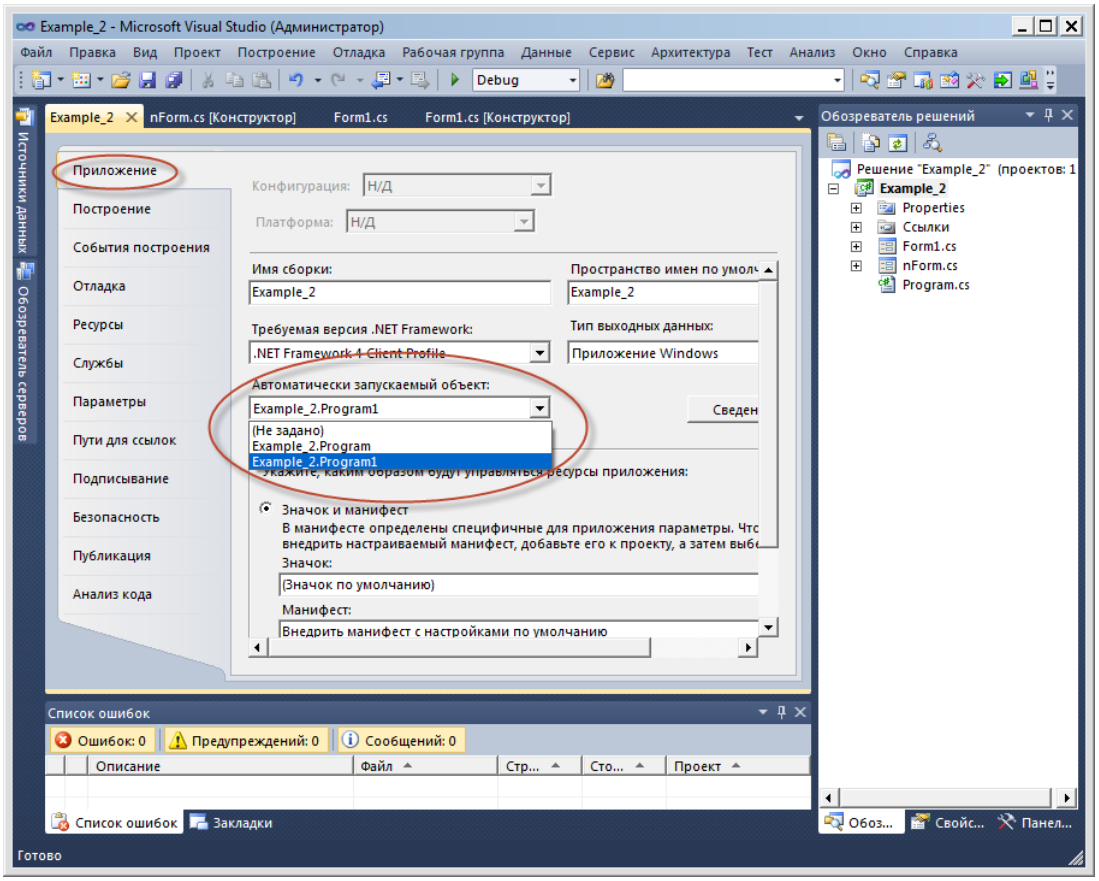

Рис. 1.2.

11. Побудуйте і запустіть застосування. Повинна відкритися похідна форма зі своїми властивостями. Перевірте, чи спадкується закриття форми кнопкою.

Вправа 4. Створення MDI-застосування (6 балів)

У цій вправі ви створите MDI-застосування з батьківською формою, яка завантажує і створює дочірні форми. Також ви познайомитеся з елементом управління **MenuStrip**, який дозволяє створити меню форми.

Створення нового проекту з базовою формою

1. Створіть новий проект **Windows Forms**, вкажіть ім'я **MdiApplication**.
2. Перейменуйте файл **Form1.cs** на **ParentForm.cs**.
3. Для форми задайте наступні властивості:

Name	ParentForm
Size	420; 320
Text	Parent Form

4. Перевірте, що сталися зміни у функції Main так, щоб форма **ParentForm** стала стартовою.
5. Відкрийте файл **ParentForm.cs** в режимі конструктора.
6. Для властивості форми **IsMdiContainer** задайте значення **True**.
   У такий спосіб ця форма буде визначена як батьківська форма **MDI**.

Створення меню для роботи з формами

7. Створіть пункт меню **File**:

a. Відкрийте панель інструментів **Toolbox**, додайте на форму елемент управління **MenuStrip** і задайте для його властивості **Name** значення **MdiMenu**.
b. Виділіть меню у верхній частині форми і задайте ім'я першого пункту меню **&File**.
c. Для властивості **Name** пункту меню **File** задайте значення **FileMenuItem**.
d. Розкрийте меню **File**.
e. Виділіть елемент, який з'явився під елементом **File**, і задайте його як **&New**.
f. Для властивості **Name** пункту меню **New** задайте значення **NewMenuItem**.
g. Виділіть елемент, що з'явився під елементом **New**, і задайте його як **&Exit**.
h. Для властивості **Name** пункту меню **Exit** задайте значення **ExitMenuItem**.
i. Двічі клацніть лівою кнопкою миші по пункту меню **Exit** для створення обробника події **Click**.
j. В обробник події **Click** для пункту меню **Exit** додайте наступний код:

```
this.Close();
```

8. Створіть пункт меню **Window**:
   a. Перейдіть в режим конструктора.
   b. Виділіть другий пункт меню праворуч від **File** і задайте його значення **&Window**.
   c. Для властивості **Name** пункту меню **Window** задайте значення **WindowMenuItem**.
   d. Розкрийте меню **Window**.
   e. Виділіть елемент, який з'явився під елементом **Window**, і задайте для його властивості **Text** значення **&Cascade**.
   f. Для властивості **Name** пункту меню **Cascade** задайте значення **WindowCascadeMenuItem**.
   g. Виділіть елемент, що з'явився під елементом **Cascade**, і задайте для його властивості **Text** значення **&Tile**.
   h. Для властивості **Name** пункту меню **Tile** задайте значення **WindowTileMenuItem**.
   i. Двічі клацніть лівою кнопкою миші по пункту меню **Cascade** для створення обробника події **Click**:

```
this.LayoutMdi(System.Windows.Forms.MdiLayout.Cascade);
```

   j. Поверніться в режим конструктора і двічі кликніть лівою кнопкою миші по пункту меню **Tile**.
   k. В обробник події **Click** для пункту меню **Tile** додайте наступний код:

```
this.LayoutMdi(System.Windows.Forms.MdiLayout.TileHorizontal);
```

9. Реалізуйте список відкритих вікон у меню **Window**:
   a. У конструкторі виберіть компонент **MdiMenu**. Вкажіть у властивості **MdiWindowListItem** ім'я пункту, створеного для цього - **WindowMenuItem**.

Створення дочірньої форми

10. Створіть дочірню форму:
    a. Виберіть пункт меню **Project → Add Windows Form…**.
    b. Задайте ім'я форми **ChildForm.cs**.
    c. Для властивості **Text** форми задайте значення **Child Form**.
    d. На панелі інструментів **Toolbox** двічі клацніть лівою кнопкою миші по елементу управління **RichTextBox** і задайте для його властивості **Name** значення **ChildTextBox**.
    e. Для властивості **Dock** елемента управління **RichTextBox** задайте значення **Fill**.
    f. Видаліть існуючий текст (якщо він є) для властивості **Text** елемента управління **RichTextBox** і залишіть його порожнім.

g. На панелі інструментів **Toolbox** двічі клацніть лівою кнопкою миші по елементу управління **MenuStrip**.
h. Для властивості **Name** елемента управління **MenuStrip** задайте значення **ChildWindowMenu**.
i. Виділіть меню у верхній частині форми і наберіть текст **F&ormat**.
j. Для властивості **Name** пункту меню **Format** задайте значення **FormatMenuItem**, для властивості **MergeAction** встановіть значення **Insert**, а властивості **MergeIndex** – **(+1)**. У цьому випадку меню **Format** розташовуватиметься після **File** при об'єднанні базового і дочірніх меню.
k. Виділіть елемент, який з'явився під елементом **Format**, і наберіть текст **&Toggle Foreground**.
l. Для властивості **Name** пункта меню **Toggle Foreground** задайте значення **ToggleMenuItem**.
m. Двічі клацніть лівою кнопкою миші по пункту меню **Toggle Foreground** і додайте наступний код в обробник події **Click**:

```
if (ToggleMenuItem.Checked)
{
    ToggleMenuItem.Checked = false;
    ChildTextBox.ForeColor = System.Drawing.Color.Black;
}
else
{
    ToggleMenuItem.Checked = true;
    ChildTextBox.ForeColor = System.Drawing.Color.Blue;
}
```

Відображення дочірньої форми

11. Відобразіть дочірню форму у батьківській формі:
    a. Відкрийте **ParentForm.cs** в режимі конструктора.
    b. Двічі кликніть лівою кнопкою миші по кнопці **New** в меню **File** для створення обробника події **Click**.
    c. Додайте наступний код для обробника події **Click** для пункту меню **New**:

```
ChildForm newChild = new ChildForm();
newChild.MdiParent = this;
newChild.Show();
```

Робота із застосуванням

12. Перевірте роботу застосування :
    a. Побудуйте і запустіть застосування.
    b. Коли з'явиться батьківська форма, виберіть пункт меню **File → New**.
    c. У батьківському вікні з'явиться нова дочірня форма. Зверніть увагу на те, що дочірнє меню зливається з батьківським і пункти меню упорядковуються у відповідності з властивістю **MergeIndex**, встановленою раніше.
    d. Наберіть який-небудь текст в дочірньому вікні і скористайтеся пунктом меню **Format** для зміни кольору шрифту тексту.
    e. Відкрийте ще декілька дочірніх вікон.
    f. Виберіть пункт меню **Window → Tile**. Зверніть увагу на те, що дочірні вікна вибудовуються впорядковано.
    g. Закрийте усі дочірні вікна.
    h. Зверніть увагу на те, що, коли закриється останнє дочірнє вікно, меню батьківської форми зміниться, і звідти зникне пункт **Format**.

i. Для закриття застосування виберіть пункт меню **File → Exit**.

13. Зверніть увагу, що заголовок у дочірніх вікон однаковий. При створенні декількох документів, наприклад у Microsoft Word, вони називаються **ДокументN**, де **N** - номер документу. Реалізуйте цю можливість:
    a. Відкрийте код батьківської форми і в класі **ParentForm** оголосіть змінну **openDocuments**:

```
private int openDocuments = 0;
```

    b. До властивості **Text** дочірньої форми додайте лічильник числа документів (який відкривається, в коді обробника події **Click** для пункту меню **New**):

```
newChild.Text = newChild.Text+" "+ ++openDocuments;
```

14. Запустіть застосування. Тепер заголовки нових документів містять порядковий номер.

Додаткова вправа

Для поглиблення знань про додавання і налаштування форм Windows виконайте наступні завдання.

Завдання 1. Створіть призначену для користувача форму, яка під час виконання матиме овальний контур. Ця форма повинна містити функціональність, яка дає можливість користувачеві закривати її під час виконання.

Рекомендація: при розробці форми у вигляді еліпса використайте наступний код:

```
// Додавання еліпса, вписаного в прямокутну форму заданої ширини і висоти
myPath.AddEllipse(0, 0, this.Width, this.Height);
```

Завдання 2. Створіть застосування з двома формами і встановіть другу форму як стартову. Зробіть так, щоб при запуску стартова форма розгорталася до максимальних розмірів і містила функціональність, що дає можливість користувачеві відкрити першу форму, яка відображається у вигляді ромба зеленого кольору з кнопкою (в центрі ромба) закриття форми з написом **GREENPEACE**.