Лабораторна робота №13

Розробка MDI- і SDI-додатків за допомогою компонентів Designer Forms

Мета роботи – Отримати практичні навички роботи щодо створення MDI- і SDI-додатків за допомогою компонентів Designer Forms.

Дана лабораторна робота сприяє напрацюванню наступних **компетентностей** у відповідності до Національної рамки кваліфікації:

Знання:

склад і призначення типових компонентів Designer Forms;

технологія створення MDI- і SDI-додатків;

особливості застосування діалогових вікон.

Уміння:

використовувати стандартні компоненти Designer Forms;

розробляти MDI- і SDI- Windows-додатки,

Комунікації:

аргументована взаємодія з клієнтами та замовниками при виборі технології розробки MDI- і SDI- Windows додатків;

робота в команді над окремими фрагментами MDI- і SDI- Windows додатків.

Автономність і відповідальність:

самостійне формулювання рекомендацій щодо вибору компонентів Designer Forms для створення інтерфейсу користувача;

здатність обґрунтувати доцільності застосування певної технології для створення MDI- і SDI- Windows-додатків.

Основні положення

Елементи управління - це компоненти, що забезпечують взаємодію між користувачем і програмою. Середа Visual Studio.NET надає велику кількість елементів, які можна згрупувати за кількома функціональних групах.

Група командних об'єктів

Елементи управління Button, LinkLabel, ToolBar реагують на натискання кнопки миші і негайно запускають яку-небудь дію. Найбільш поширена група елементів.

Група текстових об'єктів

Більшість додатків надають можливість користувачеві вводити текст і, в свою чергу, виводять різну інформацію у вигляді текстових записів. Елементи TextBox, RichTextBox приймають текст, а елементи Label, StatusBar виводять її. Для обробки введеного користувачем тексту, як правило, слід натиснути на один або декілька елементів з групи командних об'єктів.

Група перемикачів

Додаток може містити кілька визначених варіантів виконання дії або завдання; елементи управління цієї групи надають можливість вибору користувачеві. Це одна із самих великих груп елементів, в яку входять ComboBox, ListBox, ListView, TreeView, NumericUpDown і багато інших.

Група контейнерів

З елементами цієї групи дії додатки практично ніколи не зв'язуються, але вони мають велике значення для організації інших елементів управління, їх угруповання і загального дизайну форми. Як правило, елементи цієї групи, розташовані на формі, служать підкладкою кнопок, текстових полів, списків - тому вони й називаються контейнерами. Елементи Panel, GroupBox, TabControl, крім усього іншого, поділяють можливості програми на логічні групи, забезпечуючи зручність роботи.

Група графічних елементів

Навіть найпростіше додаток Windows містить графіку - іконки, заставку, вбудовані зображення. Для розміщення і відображення їх на формі використовуються елементи для роботи з графікою - Image List, Picture Box.

Діалогові вікна

Виконуючи різні операції з документом - відкриття, збереження, друк, попередній перегляд, - ми стикаємося з відповідними діалоговими вікнами. Розробникам .NET не доводиться займатися створенням вікон стандартних процедур: елементи OpenFileDialog, SaveFile Dialog, ColorDialog, PrintDialog містять вже готові операції.

Група меню

Багато користувачів налаштовують інтерфейс додатків на свій смак: одним подобається наявність певних панелей інструментів, іншим - індивідуальне розташування вікон. Але в будь-якому додатку буде

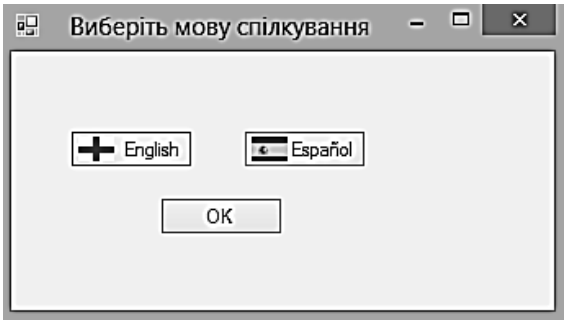
присутній меню, що містить в собі доступ до всіх можливостей і налаштувань програми. Елементи MainMenu, ContextMenu являють собою готові форми для внесення заголовків і пунктів меню.

Нижче наведено декілька прикладів, які дозоляють залучить практичні навики розробки Windows додатків за допомогою Designer Forms.

Приклад 1. Дослідження керуючого елемента Button

У прикладі буде створено діалог, в якому використовуються три кнопки.

Перші дві кнопки (рис. 8) будуть змінювати мову спілкування з англійської на іспанську і навпаки (можна використовувати будь-яку іншу мову на свій розсуд). Остання кнопка (OK) буде використовуватися для завершення діалогу.



Початковий стан інтерфейсу

Натиснута кнопка English

Натиснута кнопка Español

Рис. 8. Результат виконання програми

Технологія створення користувальницького інтерфейсу.

Крок 1. Відкрийте Visual Studio.NET і створіть новий додаток C# Windows Application. Назвіть це додаток ButtonTest.

Крок 2. Розгорніть Toolbox і виконайте клацання мишею на елементі Button. Потім пересуньте кнопки і встановіть відповідний розмір форми, як показано на рисунку 9.

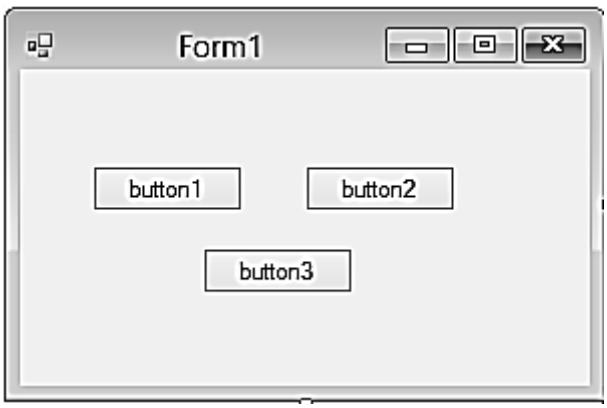


Рис. 9. Початкова форма додатку

Крок 3. Клацніть правою кнопкою миші на вкладці Properties.

Змініть властивість Name для всіх трьох кнопок наступним чином:

button1 → btnEnglish;

button2 → btnEspan;

button3 → btnOK.

Крок 4. Змініть властивість Text кожної з трьох кнопок на відповідний текст:

button1 → English;

button2 → Español

button3 → OK.

Змініть ім’я форми «Form1» на «Виберіть мову спілкування» (рис. 10)

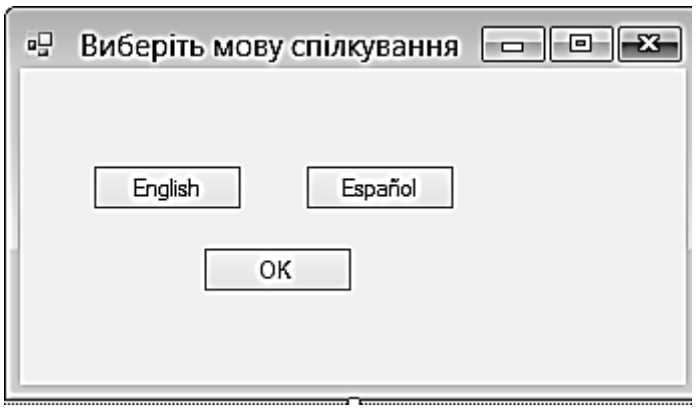


Рис. 10. Результат виконання кроку 4

Крок 5. Перед текстом необхідно вивести відповідний прапорець, щоб було зрозуміло, про яку мову йдеться.

Виберіть кнопку English і знайдіть властивість image (рис. 11).

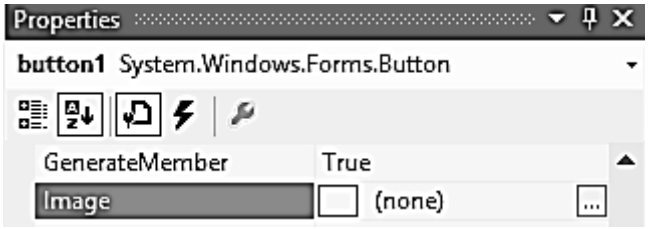


Рис. 11. Знаходження властивості image на панелі Properties

Клацніть мишею праворуч від image, для того щоб перейти в діалогове вікно, в якому можна вибирати малюнки.

Іконки з різними прапорами поставляються разом з Visual Studio.NET. Якщо ви встановили Visual Studio.NET на стандартне місце (мається на увазі англомовна версія), то вони повинні розташовуватися в директорії C \ Program Files \ Microsoft Visual Studio.NET \ Common7 \ Graphics \ icons \ Flags.

У іншому разі необхідно імпортувати (рис. 12) іконку з папки, яка додається до лабораторної роботи.

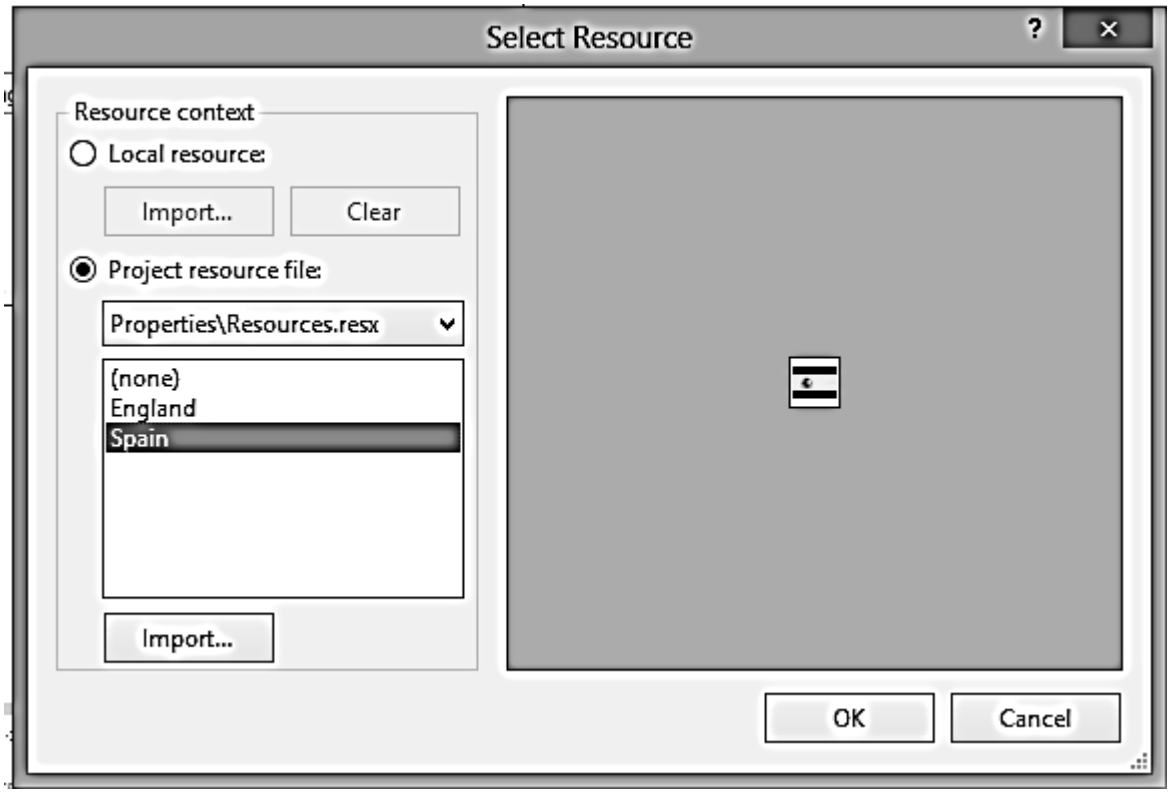


Рис. 12. Вікно імпортування іконки

Виберіть відповідну іконку. Повторіть ту ж саму процедуру для кнопки Español (за бажанням можна вибрати будь-який інший прапор з наявних у даній директорії. Результат надано на рис. 13.

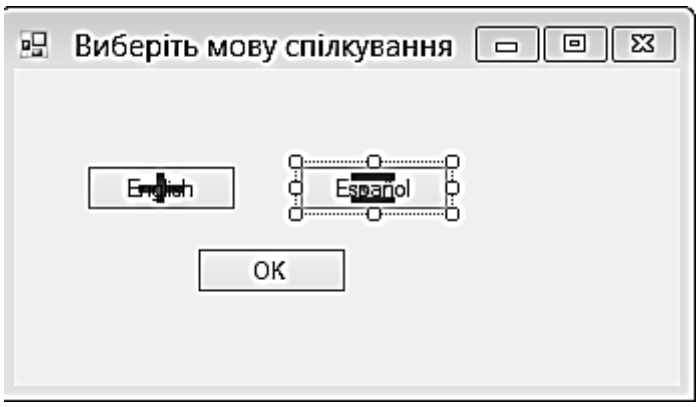


Рис. 13. Результат виконання кроку 5

Крок 6. На даний момент текст і іконка, які розташовані на кнопці, накладаються один на одного, тому необхідно змінити місця їх розташування. Для обох кнопок змініть (рис. 14):

значення властивості ImageAlign на MiddleLeft;

значення властивості TextAlign на MiddleRight.

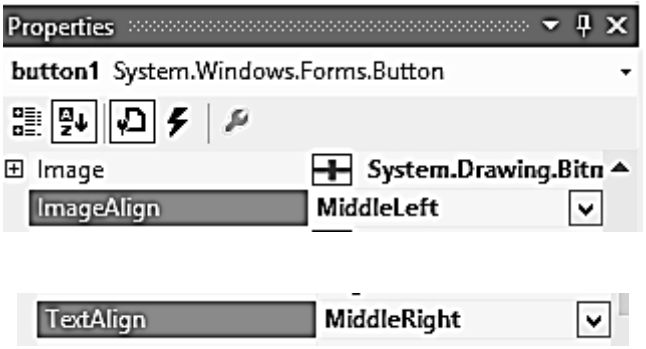


Рис. 14. Налаштування властивостей ImageAlign та TextAlign

Крок 7. У цей момент можна скорегувати ширину кнопок, з тим, щоб текст не починався безпосередньо в тому місці, де закінчується зображення. Для цього виберіть кожну кнопку і розтягніть її правий край.

Результат повинен виглядати наступним чином (рис. 15):

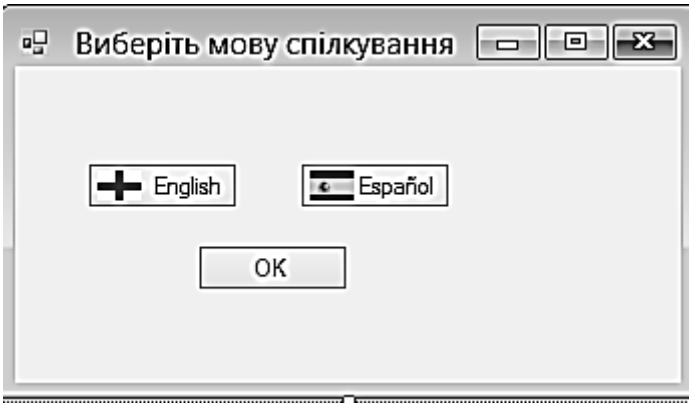


Рис. 15. Інтерфейс користувача

Додавання обробників подій.

Крок 8. Клацніть мишею два рази на кнопці English - ви потрапите безпосередньо на обробник події.

Click є подією, яка використовується для кнопок за замовчуванням; це саме та подія, яка створюється при подвійному натисканні мишею на кнопці.

При подвійному натисканні мишею на керуючому елементі в коді, який лежить в основі даної форми, відбуваються дві речі.

По-перше, в методі InitializeComponent () створюється підписка на подію (додається обробник btnEnglish_Click в список подій):

this.btnEnglish.Click + = new System.EventHandler (this.btnEnglish_Click);

Якщо виникає необхідність підписатися на подію, відмінне від події за замовчуванням, то доведеться заносити відповідний код самостійно.

Необхідно пам'ятати, що код методу InitializeComponent () перезаписується кожен раз, коли ви перемикаєтеся з режиму розробки і переходите до коду. З цієї причини ніколи не виконуйте підписку в цьому методі. Замість цього слід застосовувати конструктор даного класу.

Друге, що відбувається, - це додається власне обробник події:

```
private void btnEnglish_Click (object sender, System.EventArgs e)
{

}
```

Ім'я методу утворюється за допомогою об'єднання імені керуючого елемента, символу підкреслення та імені події, яка повинна обробляться.

Перший параметр - object sender - містить обраний керуючий елемент. У даному прикладі, це завжди буде елемент, який входить в ім'я методу, хоча в деяких інших випадках кілька керуючих елементів можуть використовувати один і той же метод для обробки події, і в такому випадку за значенням цього параметра можна точно визначити, який саме елемент викликає даний метод.

У другому параметрі - System.EventArgs e - міститься інформація про те, що саме сталося. У даному випадку немає необхідності використовувати інформацію, що зберігається в обох параметрах.

Ключове слово this, позначає поточний екземпляр даного класу. Оскільки клас, з яким ми працюємо зараз, є саме таким екземпляром, тому можна отримувати доступ до його властивостей і керуючим елементам за допомогою цього слова.

Задаємо значення властивості Text поточного екземпляра форми:

```
private void btnEnglish_Click (object sender, System.EventArgs e)
{
```

```
        this.Text = "Do you speak English?";
    }
```

Крок 9. Поверніться в Form Designer і клацніть мишею два рази на кнопці Español для переходу до обробника подій для цієї кнопки. Ось його код:

```
    private void btnEspan_Click (object sender, System.EventArgs e)
    {
        this.Text = "Usted puede hablar español ?";
    }
```

Цей метод ідентичний методу btnEnglish_Click за винятком тексту іспанською мовою.

Крок 11. Оброблювач подій для кнопки ОК додається таким же засобом, як і два попередніх. Однак код в цьому випадку буде трохи відрізнятися:

```
    private void btnOK_Click (object sender, System.EventArgs e)
    {
        Application.Exit ();
    }
```

За допомогою цього методу завершується виконання програми. Повний текст файлу ButtonTest.Form1 наведено нижче:

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

    namespace ButtonTest
    {
       public partial class Form1: Form
       {
          public Form1()
          {
             InitializeComponent();
          }
          private void btnEnglish_Click(object sender, EventArgs e)
          {
```

```
            this.Text = "Do you speak English?";
        }
        private void btnEspan_Click(object sender, EventArgs e)
        {
            this.Text = "Usted puede hablar español  ?";
        }
        private void btnOK_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Результат виконання програми відповідає рис. 8..

Приклад 2. Розробка Windows-додатка Блокнот

Необхідно розробити додаток Блокнот, який повторює в найпростішому варіанті функції стандартної однойменної Windows-програми.

Створення інтерфейсу головного меню

Більшість Windows-додатків має головне меню, яке являє собою ієрархічну структуру функцій і команд. Практично всі функції, які можна здійснити за допомогою елементів управління, мають свій аналог у вигляді відповідних пунктів меню.

Нижче наведена покрокова технологія розробки основного меню додатка Блокнот за допомогою стандартних компонентів панелі Designer Forms.

Крок 1. Налаштування початкової форми та панелі Designer Forms.

Створіть новий додаток і назвіть його NotepadCSharp.

Встановіть наступні властивості форми:

Name → frmmain

Icon → …\Icon\README.ICO

Text → Notepad C#

WindowState → Maximized

Результат наведено на рис. 16.

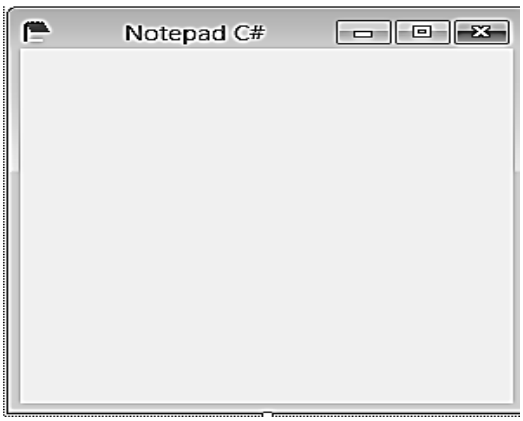


Рис. 16. Початкова форма додатка Блокнот

Перетягуємо елемент управління MainMenu, який знаходиться на панелі інструментів Toolbox, на форму (рис. 17).

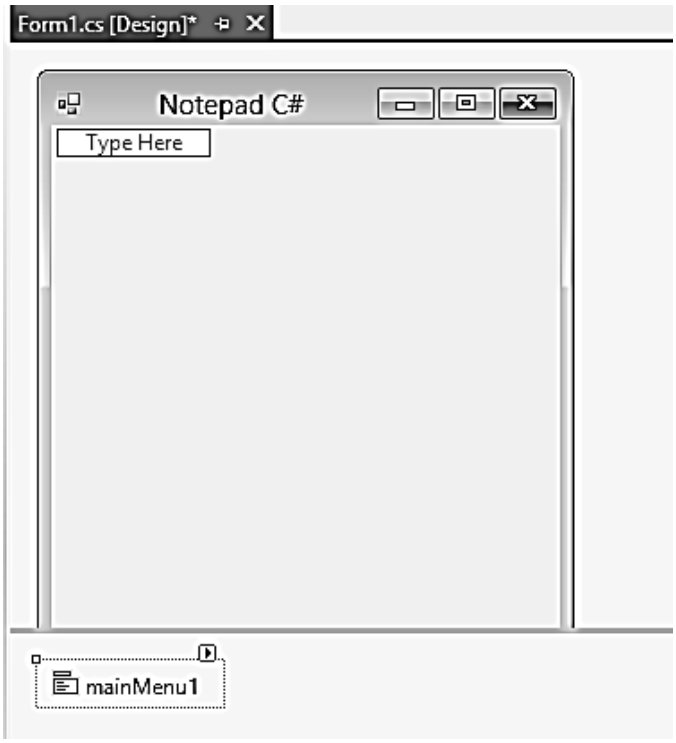


Рис. 17. Результат виконання кроку 1.

Якщо елемент MainMenu відсутній (в версії Visual Studio 2012) можна застосовувати елемент MenuStrip, але йому бракує декілька необхідних надалі властивостей.

Для додавання в панель Toolbox компонента MainMenu необхідно з контекстного меню вкладки Menu & Toolbars викликати команду Choose Items… Далі у вікні (рис. 18) необхідно відмітити елемент Menu.

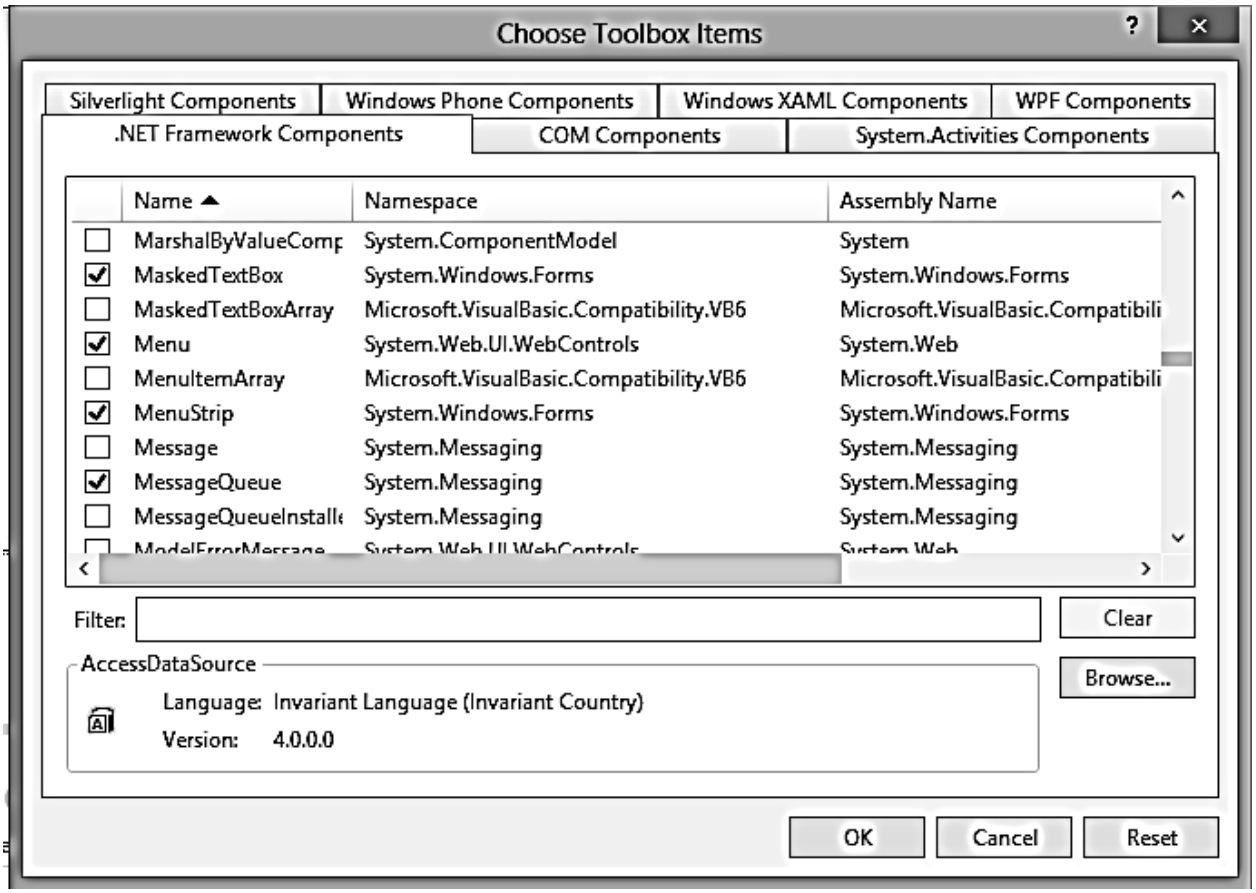


Рис. 18. Вікно додавання елементу MainMenu в панель Toolbox

Як результат в панелі Toolbox з'явиться необхідний нам компонент.

Крок 2. Створення інтерфейсу головного меню.

Необхідно заповнити рядки меню наступними пунктами (рис. 19):

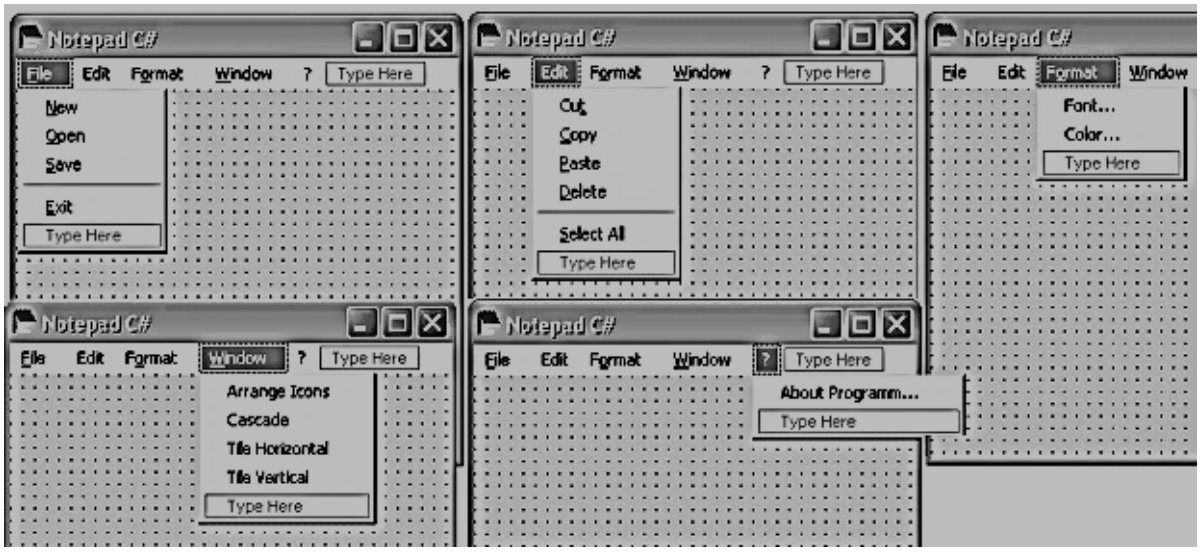


Рис. 19. Пункти головного меню програми Notepad C#

Кожен пункт головного меню має своє вікно властивостей, в якому, подібно іншим елементам управління, задаються значення властивостей Name, Text і Shortcut (рис. 20).

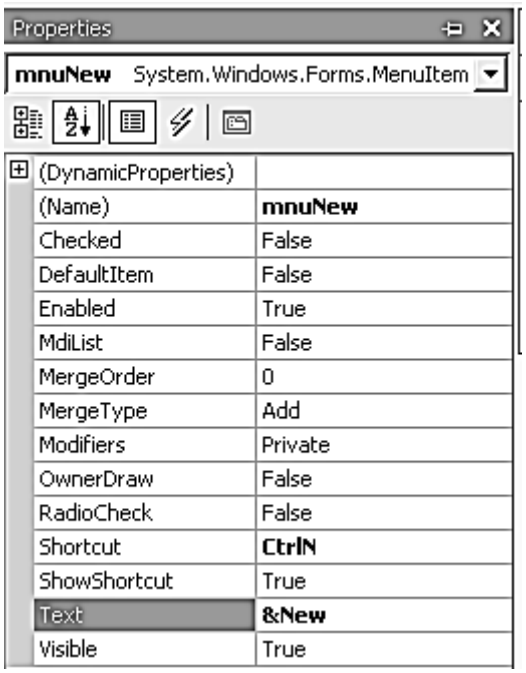


Рис. 20. Властивості пункту меню New

У полі Text перед словом New стоїть знак & - так званий амперсанд, який вказує, що N повинно бути підкреслена і буде частиною вбудованого клавіатурного інтерфейсу Windows. Коли користувач на клавіатурі натискає клавішу Ctrl і потім N, виводиться підменю New.

У Windows є ще інтерфейс для роботи з так званими швидкими клавішами, або акселераторами. Поєднання клавіш вказують з перерахування Shortcut.

Горизонтальна розділова лінія використовується в тих випадках, коли треба візуально відокремити подібні групи завдань.

Для використання пунктів меню в коді, їм також призначають імена (властивість Name).

Слід призначати стандартним пунктам загальноприйняті сполучення клавіш. Властивості пунктів меню в додатку Notepad C # наводяться в таблиці 2.

Таблиця 2

Пункти головного меню додатку Notepad C#

Name	Text	Shortcut
mnuFile	File	
mnuNew	&New	CtrlN
mnuOpen	&Open	CtrlO
mnuSave	&Save	CtrlS
mnuExit	&Exit	CtrlE
mnuEdit	Edit	
mnuCut	Cut	
mnuCopy	Copy	
mnuPaste	Paste	
mnuDelete	Delete	
mnuSelectAll	SelectAll	
mnuFormat	Format	
mnuFont	Font…	
mnuColor	Color…	
mnuWindow	Window	
mnuArrangeIcons	Arrange Icons	
mnuCascade	Cascade	
mnuTileHorizontal	Tile Horizontal	
mnuTileVertical	Tile Vertical	
mnuHelp	?	
mnuAbout	About Programm...	

Можна самостійно вибрати поєднання клавіш, які не зазначені в табл. 1, для відповідних пунктів меню.

Результат виконання програми у вигляді відповідного інтерфейсу наведено на рис. 21.

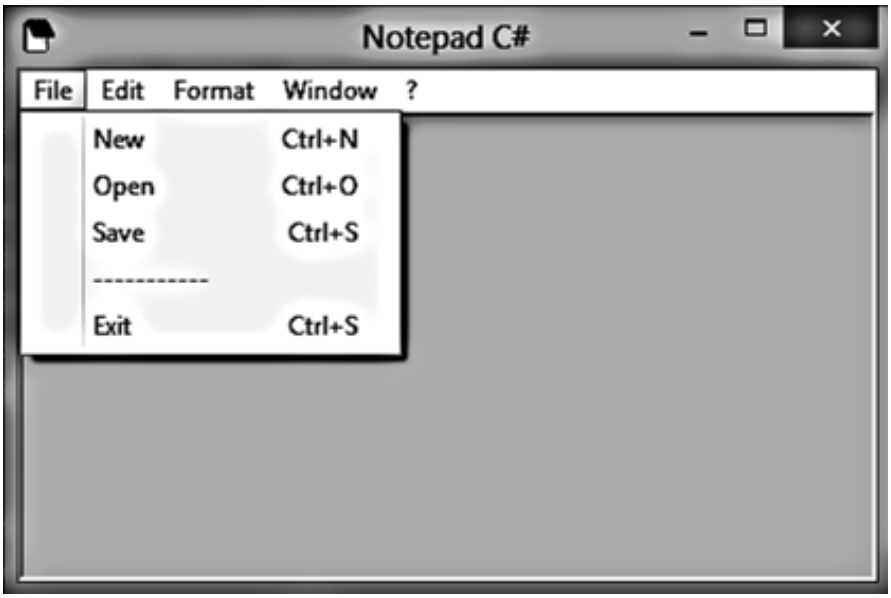


Рис. 21. Інтерфейс програми Блокнот

Створення MDI-додатків

Такі програми, як блокнот і Microsoft Paint, відносяться до SDI (Single - Document Interface) додаткам, здатним працювати тільки з одним документом. Інші, такі як Microsoft Word або Adobe Photoshop, підтримують роботу відразу з декількома документами і називаються MDI (Multiple - Document Interface) додатками.

У MDI-додатках головна форма містить в собі кілька документів, кожен з яких є полотном в графічних програмах або полем для тексту в редакторах.

Продовжимо роботу над додатком Notepad C#. Наша задача – на базі вже розробленого інтерфейсу програми Блокнот створити багато віконний (MD)I додаток.

Спочатку (перший етап) необхідно додати до основної форми додаткову (дочірню) форму і виконати відповідне налаштування її властивостей. Після чого (другий етап) потрібно підключити до кожного управляючого елементу меню відповідні обробники подій.

Етап 1. Створення і налаштування дочірній форми.

Крок 1. Створення пустої форми blank.

У вікні Solution Explorer клацаємо правою кнопкою на імені проекту і в контекстному меню вибираємо команду Add / Add Windows Form ....

У вікні, яке з’явилось (рис. 22) присвоюємо формі ім’я blank.cs.

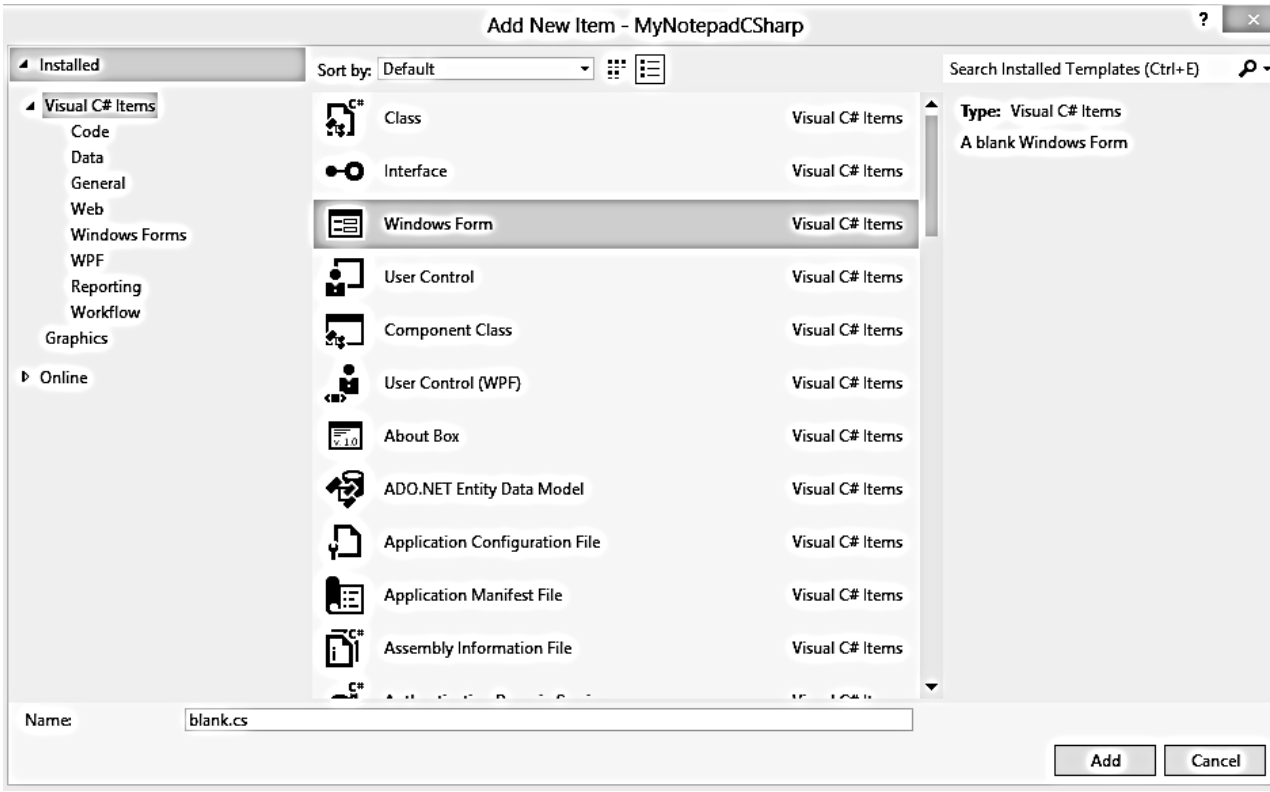


Рис. 22. Вікно створення пустої форми blank

У нашому проекті з'явилася нова форма - будемо називати її дочірньою формою.

У вікні Solution Explorer ім’я головної форми Form1 замінимо на frmmain (рис. 23).

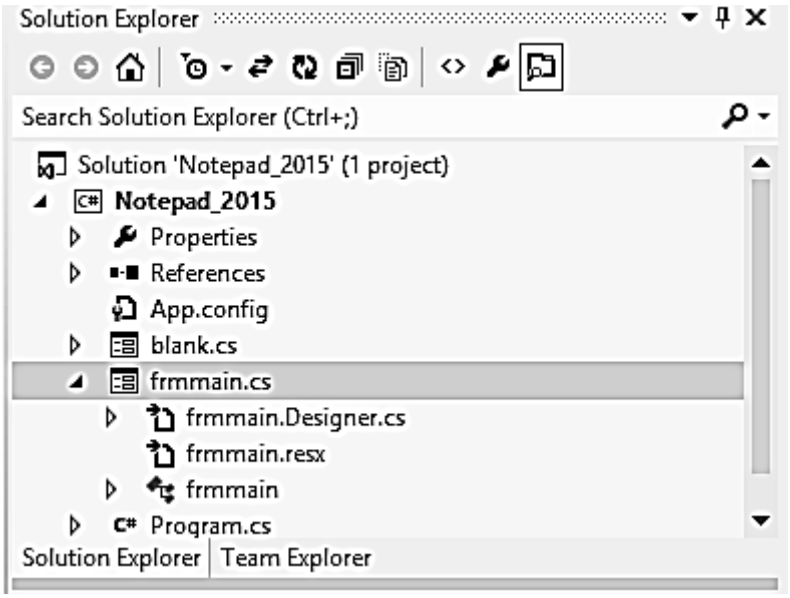
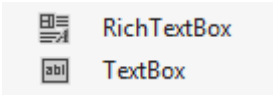


Рис. 23. Вікно інспектора Solution Explorer

Крок 2. Налаштування властивостей форми blank.

У режимі дизайну перетягуємо на форму blank елемент управління RichTextBox: 

На відміну від елемента textBox, розмір вмісту тексту в ньому не обмежується 64 Кб; крім того, RichTextBox дозволяє редагувати колір тексту та додавати зображення.

Властивість Dock цього елемента фіксуємо значенням Fill (рис. 24):

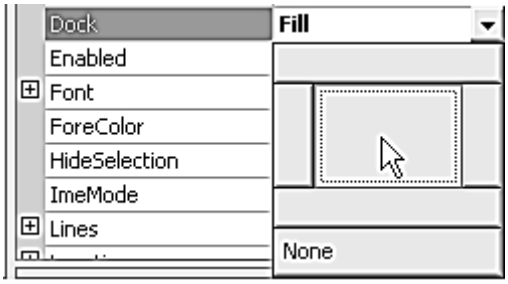


Рис. 24. Властивість Dock елемента RichTextBox

Крок 3. Налаштування кольору основної форми.

Переходимо в режим дизайну форми frmmain і встановлюємо властивості IsMdiContainer значення true (рис.25). Колір форми при цьому стає темно-сірим.

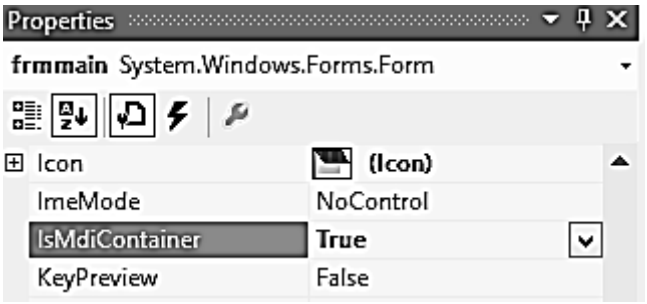


Рис. 25. Налаштування кольору основної форми

Етап 2. Підключення обробників подій.

Крок 1. Обробник події кнопки New.

Нові екземпляри документів повинні з'являтися при натисканні пункту меню New (або поєднання клавіш Ctrl + N).

Двічі клацаємо на пункті New і переходимо в шаблон обробника події. Після чого записуємо в шаблон наступний код:

```
private void mnuNew_Click(object sender, EventArgs e)
        {
            // Створюємо новий екземпляр форми frm
            blank frm = new blank();
            // Вказуємо, що батьківським контейнером нового екземпляру
            //                                буде ця, головна форма
            frm.MdiParent = this;
            // Викликаємо форму
            frm.Show();
        }
```

Запускаємо програму. Тепер кожен раз при натисканні клавіш Ctrl + N або виборі пункту меню New з'являється кілька вікон, розташованих каскадом. Однак заголовок у всіх однаковий – blank (рис. 26).

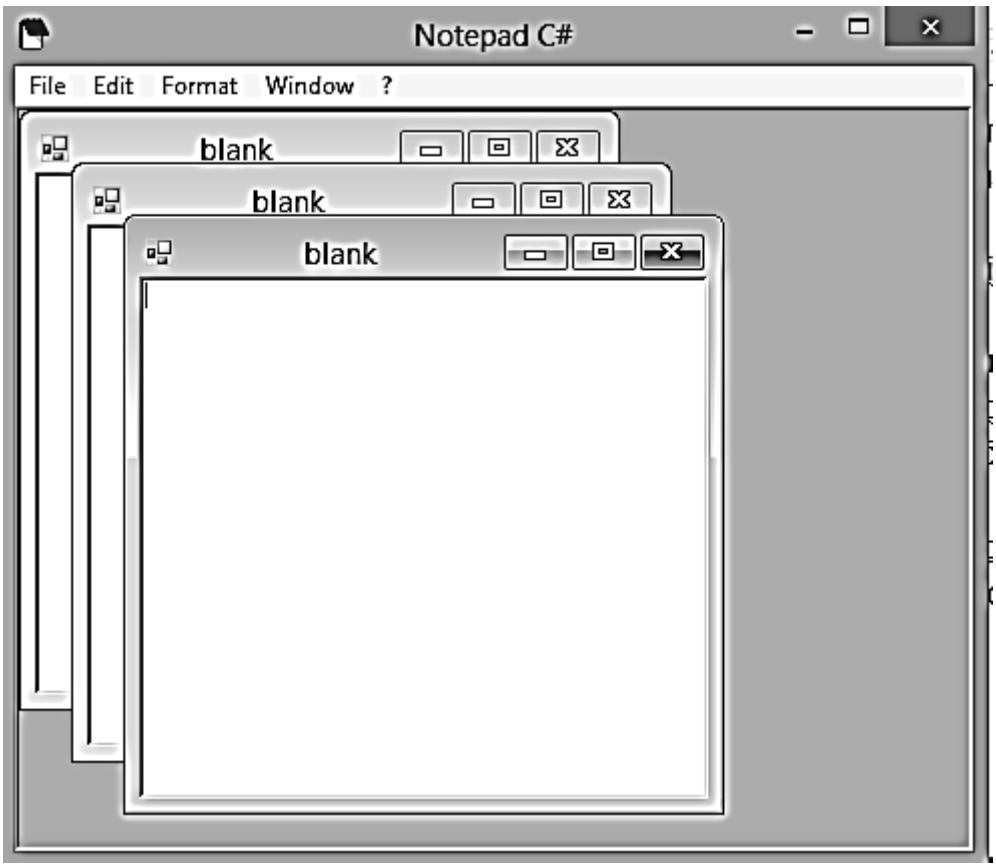


Рис. 26. Результат виклику дочірніх вікон

Крок 2. Привласнення кожному екземпляру документа відповідного номера.

Для того, щоб кожному екземпляру нового документа присвоювався свій номер необхідно виконати наступні дії.

Перемикаємося в код форми blank, і в класі blank оголошуємо змінну DocName:

```
public partial class blank : Form
   {
      public string DocName = "";
      public blank()
      {
         InitializeComponent();
      }
   }
```

Перемикаємося в код форми frmmain і в класі frmmain оголошуємо змінну openDocuments:

```
public partial class frmmain : Form
   {
      private int openDocuments = 0;
      public frmmain()
      {
         InitializeComponent();
      }
...
```

Модифікуємо обробник події mnuNew_Click наступним чином:

```
private void mnuNew_Click(object sender, EventArgs e)
      {
         // Створюємо новий екземпляр форми frm
         blank frm = new blank();
         frm.DocName = "Untitled " + ++openDocuments;
         // Вказуємо, що батьківським контейнером
         // нового примірника буде ця, головна форма.
         frm.MdiParent = this;
         frm.Text = frm.DocName;
         // Викликаємо форму
         frm.Show();
      }
```

Запускаємо програму. Тепер нові документи містять різні заголовки (рис. 27).

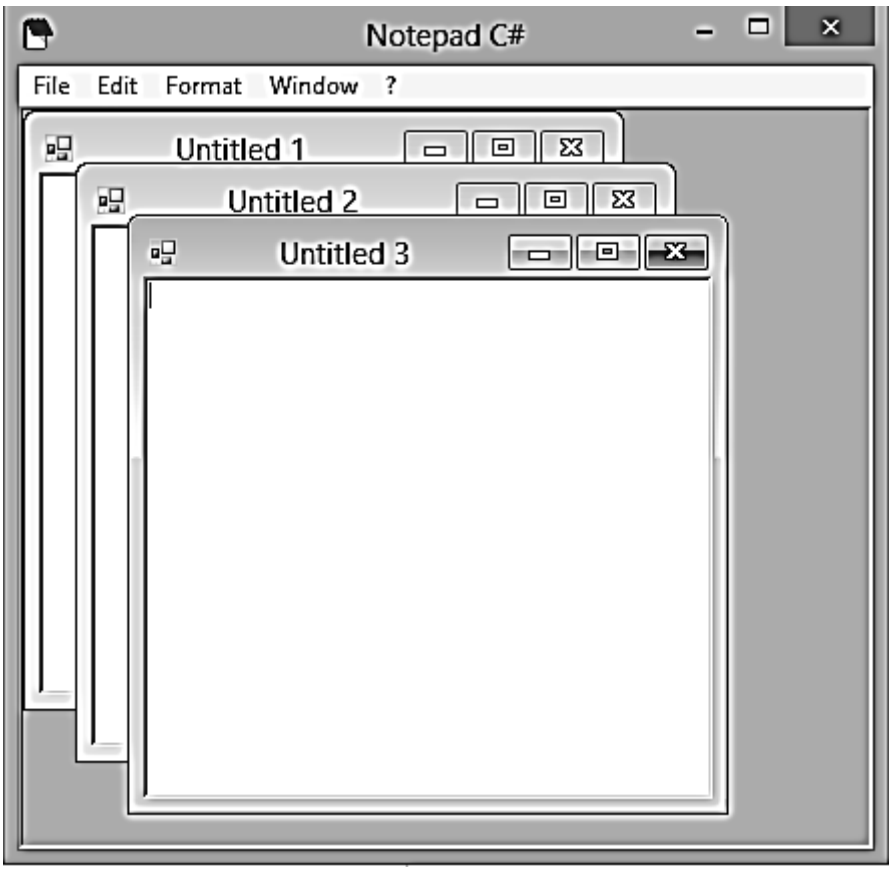


Рис. 27. Нові документи мають впорядковані назви

Крок 3. Впорядкування документів на екрані. Обробники подій пункту Window.

При роботі з декількома документами в MDI-додатках зручно впорядковувати їх на екрані. Можна, звичайно, розподіляти форми вручну, але при роботі з великою кількістю документів це представляється скрутним. У пункті меню Window реалізуємо процедуру вирівнювання вікон, для чого створюємо відповідні обробники:

```
private void mnuArrangeIcons_Click(object sender, EventArgs e)
    {
       this.LayoutMdi(MdiLayout.ArrangeIcons);
    }

    private void mnuCascade_Click(object sender, EventArgs e)
    {
       this.LayoutMdi(MdiLayout.Cascade);
    }
```

```
private void mnuTileHorizontal_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileHorizontal);
}

private void mnuTileVertical_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileVertical);
}
```

Як слід з обробників, метод LayoutMdi містить перерахування MdiLayout, що містить чотири відповідні члени.

При вибору пункту ArrangeIcons фокус повинен перемикатись на обрану форму. Для чого у властивості MdiList пункту меню ArrangeIcons необхідно встановити значення true.

Тепер при відкритті кількох нових документів вікна розташовуються каскадом (рис. 28), їх можна розташувати горизонтально - значення TileHorizontal (рис. 29) або вертикально - значення TileVertical (рис.30), а потім знову повернути каскадне розташування - Cascade

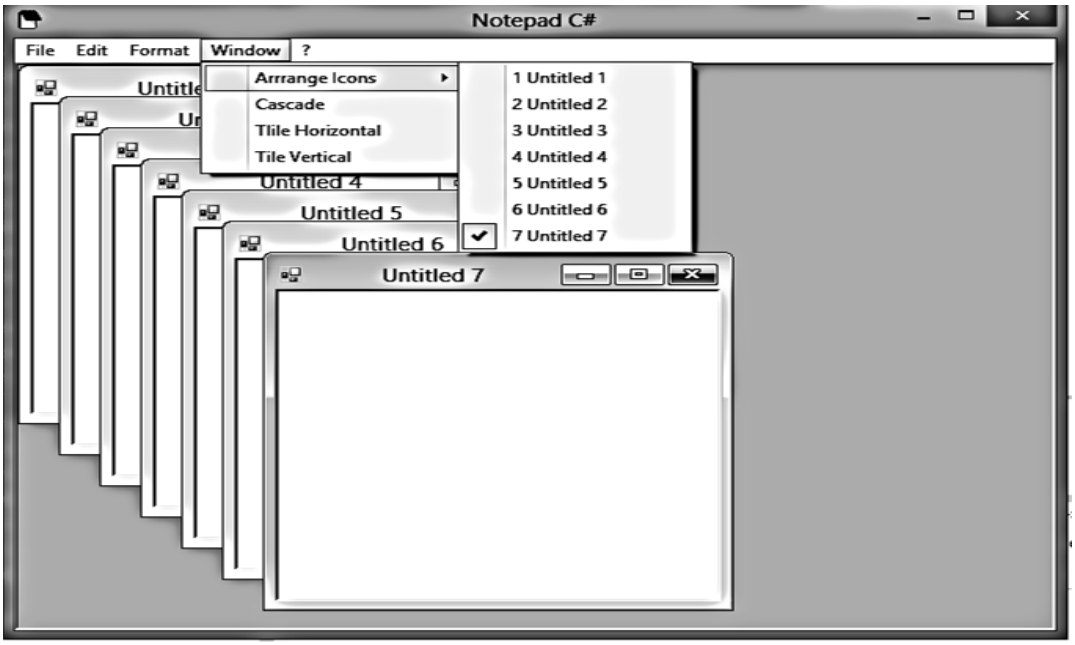


Рис. 28. При відкритті кількох нових документів вікна розташовуються каскадом

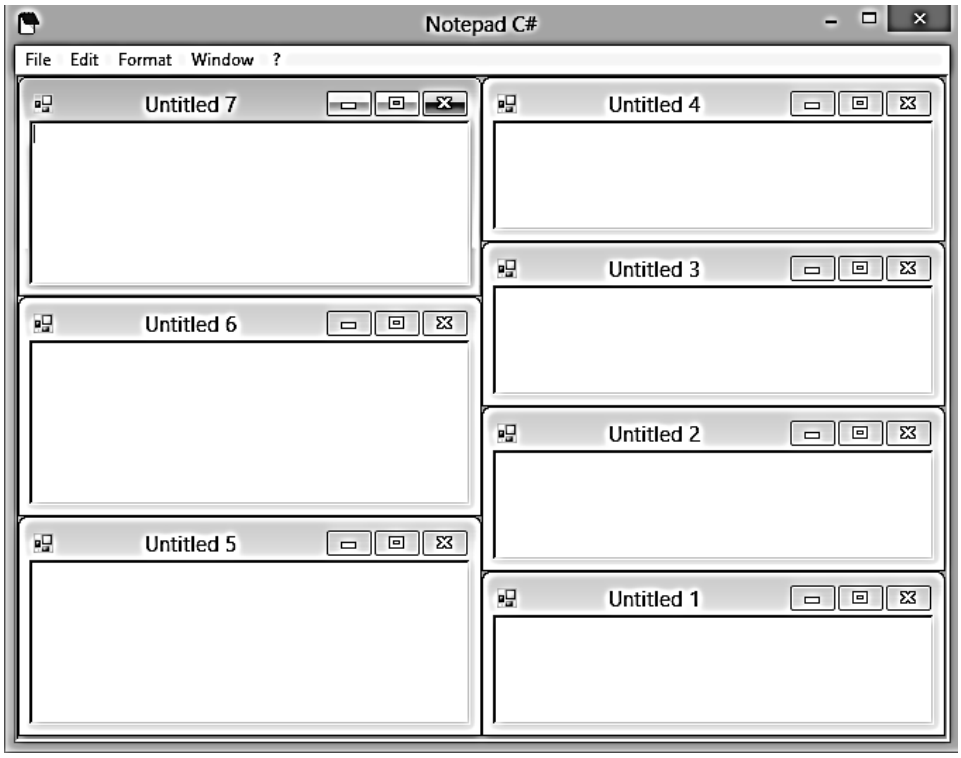


Рис. 29. Вікна розташовуються горизонтально

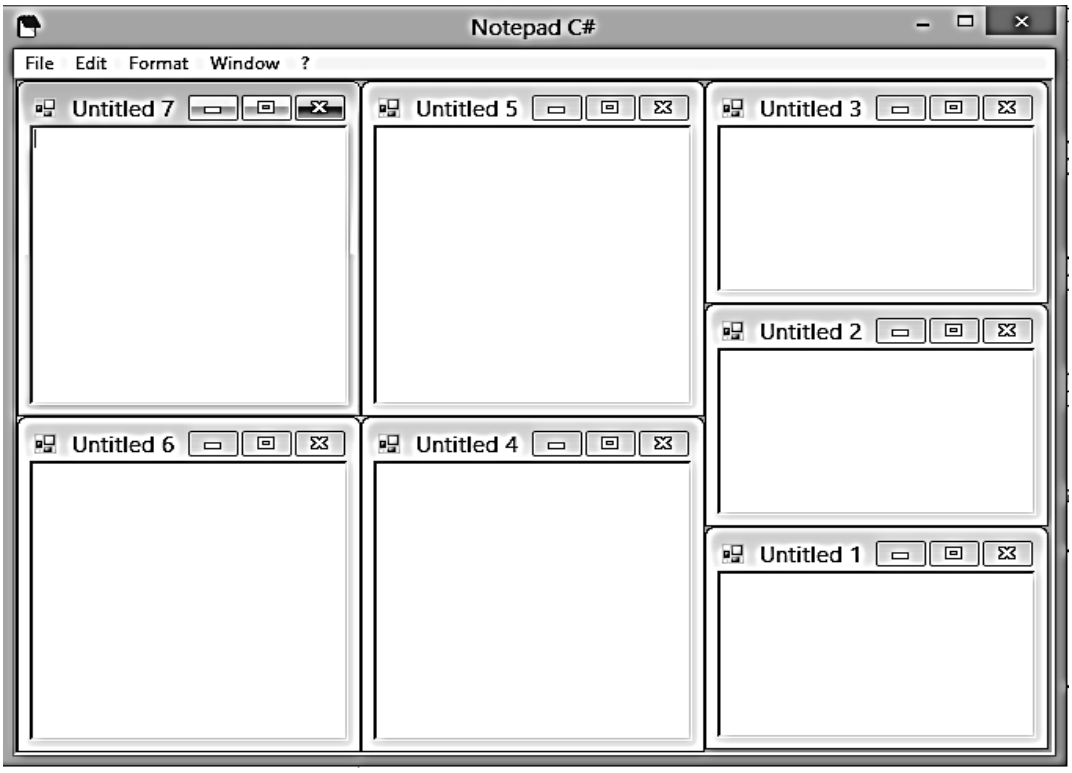


Рис. 30. Вікна розташовуються вертикально

Вирізання, копіювання і вставка текстових фрагментів

З додатком працювати буде зручніше, якщо при створенні нового документа він відразу буде займати всю область головної форми. Для цього встановимо властивість WindowState форми blank Maximized.

Приступимо до створення обробників для стандартних операцій вирізання, копіювання і вставки. Елемент управління RichTextBox має властивість SelectedText, яке містить виділений фрагмент тексту. На підставі цієї властивості і будуть реалізовані дії по роботі з текстом.

Крок 1. Оголошення буферної зміни.

У коді форми **blank** оголошуємо змінну (буфер) BufferText, в якій буде зберігатися фрагмент тексту, який потрібно запам'ятати для подальшої обробки:

```
public partial class blank : Form
   {
      public string DocName = "";
      private string BufferText = "";
      public blank()
        {
          InitializeComponent();
      }

   }
```

Крок 2. У коді форми **blank** створюємо відповідні методи пункту Edit головного меню.

```
      // Вирізання тексту
      public void Cut()
      {
         this.BufferText = richTextBox1.SelectedText;
         richTextBox1.SelectedText = "";
      }
      // Копіювання тексту
      public void Copy()
      {
         this.BufferText = richTextBox1.SelectedText;
      }
```

```
        // Вставка
        public void Paste()
        {
            richTextBox1.SelectedText = this.BufferText;
        }
        // Виділення всього тексту - використовуємо властивість SelectAll елементу управління RichTextBox
        public void SelectAll()
        {
            richTextBox1.SelectAll();
        }
        // Видалення
        public void Delete()
        {
            richTextBox1.SelectedText = "";
            this.BufferText = "";
        }
```

Крок 3. Перемикаємося в режим дизайну форми **frmmain** і створюємо обробників для пунктів меню:

```
private void mnuCut_Click (object sender, System.EventArgs e)
    {
      blank frm = (blank) this.ActiveMdiChild;
      frm.Cut ();
    }

private void mnuCopy_Click (object sender, System.EventArgs e)
    {
    blank frm = (blank) this.ActiveMdiChild;
      frm.Copy ();
    }

private void mnuPaste_Click (object sender, System.EventArgs e)
    {
      blank frm = (blank) this.ActiveMdiChild;
      frm.Paste ();
    }
```

```
private void mnuDelete_Click (object sender, System.EventArgs e)
    {
      blank frm = (blank) this.ActiveMdiChild;
      frm.Delete ();
    }

private void mnuSelectAll_Click (object sender, System.EventArgs e)
    {
      blank frm = (blank) this.ActiveMdiChild;
      frm.SelectAll ();
    }
```

Властивість ActiveMdiChild перемикає фокус на поточну форму, якщо їх відкрито декілька, і викликає один з методів, визначених у дочірньої формі. Запускаємо програму. Тепер ми можемо виконувати всі стандартні операції з текстом.

Контекстне меню

Контекстне меню, яке дублює деякі дії основного меню, - самий звичний інструмент для користувача. Елемент управління TextBox містить в собі найпростіше контекстне меню, яке дублює дії підміню Edit. Для того щоб переконатися в цьому, досить нанести цей елемент управління на форму і запустити додаток (рис. 2.8):

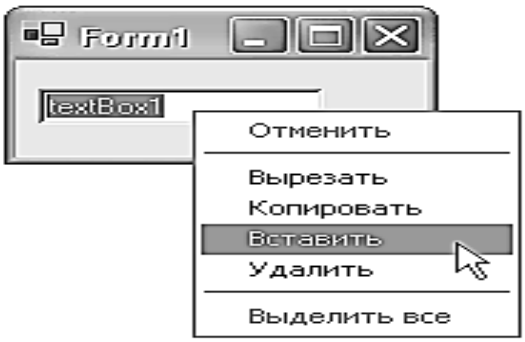


Рис. 31. Контекстне меню елемента TextBox

Крок 1.

У нашому додатку Notepad C# в якості текстового елемента ми використовуємо RichTextBox.

Перетаскуємо елемент управління contextMenuStrip (у разі Visual Studio 2012) з вікна ToolBox на форму **blank**.

Крок 2. Додаємо пункти контекстного меню точно так само, як ми це робили для головного меню (рис. 32).

Не забуваємо перейменувати (властивість Name) відповідним чином кожен з пунктів меню:

Cut → cmnuCut;

Copy → cmnuCopy;

Paste → cmnuPaste;

Delete → cmnuDelete;

Select All → cmnuSelectAll;

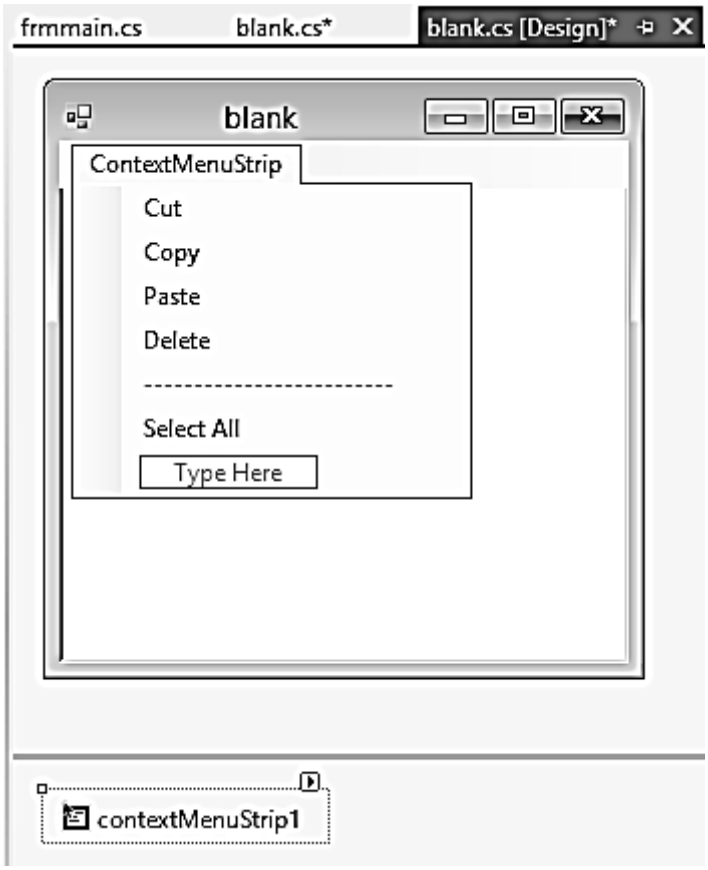


Рис. 32. Пункти контекстного меню

Крок 3. Перемикаємося в **режим дизайну форми blank** і створюємо обробників для відповідних пунктів контекстного меню:

```
private void cmnuCut_Click(object sender, EventArgs e)
    {
        Cut();
```

```
}

private void cmnuCopy_Click(object sender, EventArgs e)
{
    Copy();
}

private void cmnuPaste_Click(object sender, EventArgs e)
{
    Paste();
}

private void cmnuDelete_Click(object sender, EventArgs e)
{
    Delete();
}

private void cmnuSelectAll_Click(object sender, EventArgs e)
{
    SelectAll();
}
```

Крок 4. Останнє, що нам залишилося зробити, - це визначити, де буде з'являтися контекстне меню.

Елемент RichTextBox, так само як і форми frmmain і blank, має властивість ContextMenuStrip (рис. 33). Необхідно праворуч вибрати contextMenuStrip1, оскільки нам потрібно відображати меню саме в текстовому полі.

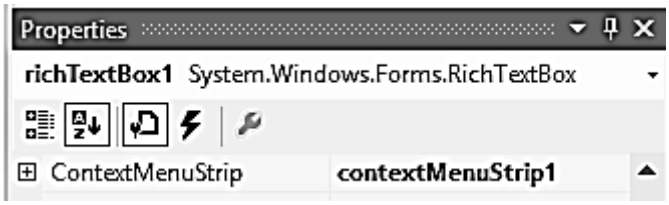


Рис. 33. Налаштування місця де буде з'являтися контекстне меню

Запускаємо програму - тепер в будь-якій точці тексту доступно меню (рис. 34).

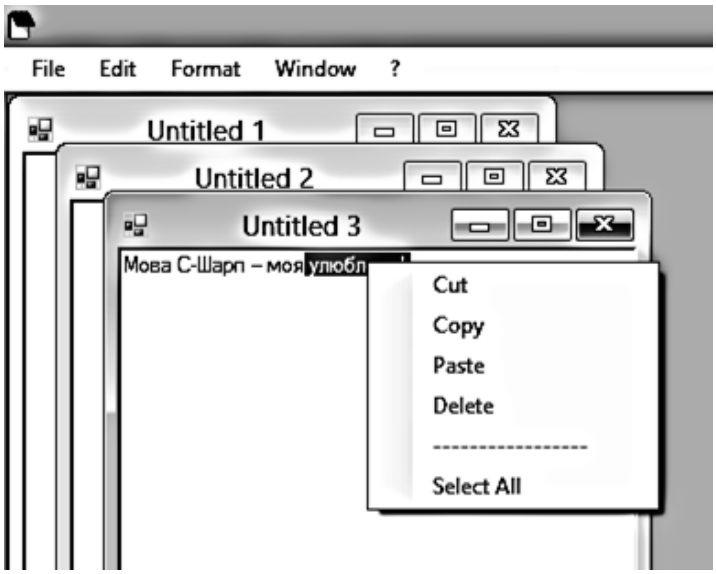


Рис. 34. Виклик контекстного меню

Діалогові вікна

Середа Visual Studio .NET містить готові діалогові вікна, що пропонують вибрати файл для відкривання або шлях к диску для збережень поточного файлу. Розглянемо покрокові технології створення типових діалогових вікон.

OpenFileDialog

Крок 1. Налаштування властивостей вікна.

Додайте на форму frmmain елемент управління OpenFileDialog з вікна панелі інструментів ToolBox. Подібно елементу MainMenu, він буде розташовуватися на панелі невидимих компонентів.

В інспекторі властивостей елементу управління OpenFileDialog (рис. 35) властивість FileName задає назву файлу, яке буде знаходитися в полі «Ім'я файлу» при появі діалогу. На рисунку назва в цьому полі – «Текстові файли (.txt)», оскільки ми будемо вводити саме текст.

Властивість Filter задає обмеження файлів, які можуть бути обрані для відкриття - у вікні будуть показуватися тільки файли з заданим розширенням.

Тут введено Text Files (* .txt) | * .txt | All Files (*. *) | *. *, що означає огляд або текстових файлів, або всіх разом.

Властивість InitialDirectory дозволяє задати директорію, звідки буде починатися огляд. Якщо це властивість не встановлено, вихідної директорією буде робочий стіл.

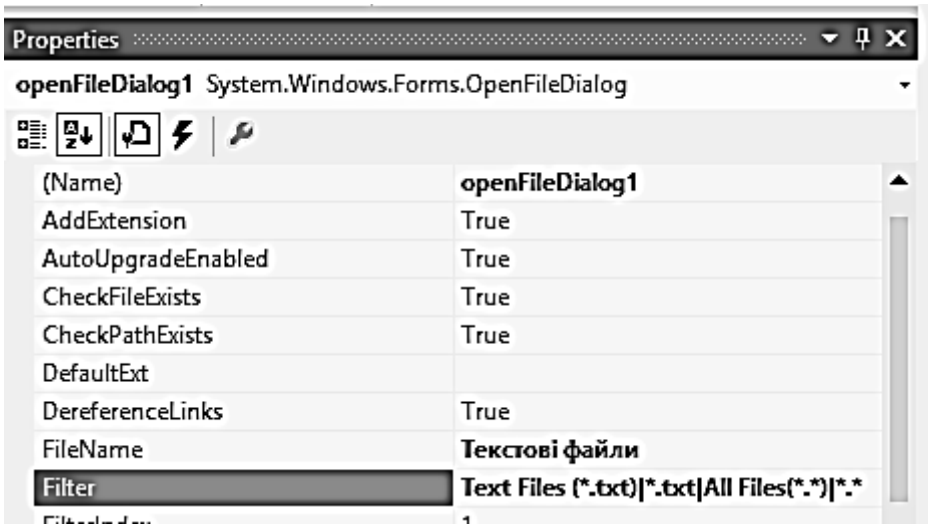


Рис. 35. Налаштування властивостей вікна OpenFileDialog

Крок 2. Робота з файловими потоками в коді форми **blank**.

2.1. Перемикаємося в режим дизайну форми blank. Для роботи з файловими потоками в коді форми blank підключаємо простір імен System.IO: using System.IO;

2.2. У методі Open записуємо вміст файлу, який потрібно відкрити, в полі RichTextBox. Це робиться за допомогою наступного методу:

```
// Створюємо метод Open, де як параметр оголошуємо рядок адресу файлу.
    public void Open (string OpenFileName)
    {
      // Якщо файл не обраний, повертаємося назад (з'явиться вбудоване попередження)
     if (OpenFileName == "")
     {
       return;
     }
     else
     {
```

```
// Створюємо новий об'єкт StreamReader і передаємо йому змінну
                                                    // OpenFileName
      StreamReader sr = new StreamReader (OpenFileName);
      // Читаємо весь файл і записуємо його в richTextBox1
      richTextBox1.Text = sr.ReadToEnd ();
      // Закриваємо потік
      sr.Close ();
      // Перемінної DocName присвоюємо адресний рядок
      DocName = OpenFileName;
    }
  }
```

Крок 3. Додаємо обробник пункту меню Open форми frmmain:

Перемикаємося в режим дизайну форми **frmmain**. Двічі клацаємо по пункту меню Open. В шаблон методу, який з’явився, записуємо наступний код:

```
private void mnuOpen_Click(object sender, EventArgs e)
      {
        // Можна програмно задавати доступні для огляду розширення файлів
//openFileDialog1.Filter = "Text Files (* .txt) | * .txt | All Files (*. *) | *. *";
        // Якщо обрано діалог відкриття файлу, виконуємо умова
        if (openFileDialog1.ShowDialog() == DialogResult.OK)
        {
          // Створюємо новий документ
          blank frm = new blank();
          // Викликаємо метод Open форми blank
          frm.Open(openFileDialog1.FileName);
          // Вказуємо, що батьківською формою є форма frmmain
          frm.MdiParent = this;
          // Надаємо змінної DocName ім'я файлу
          frm.DocName = openFileDialog1.FileName;
          // Властивості Text форми присвоюємо змінну DocName
          frm.Text = frm.DocName;
          // Викликаємо форму frm
          frm.Show();
```

```
}
```

Крок 4. Запускаємо програму і відкриваємо текстовий файл, збережений у блокноті в форматі Unicod (або ANSI) (рис. 36).

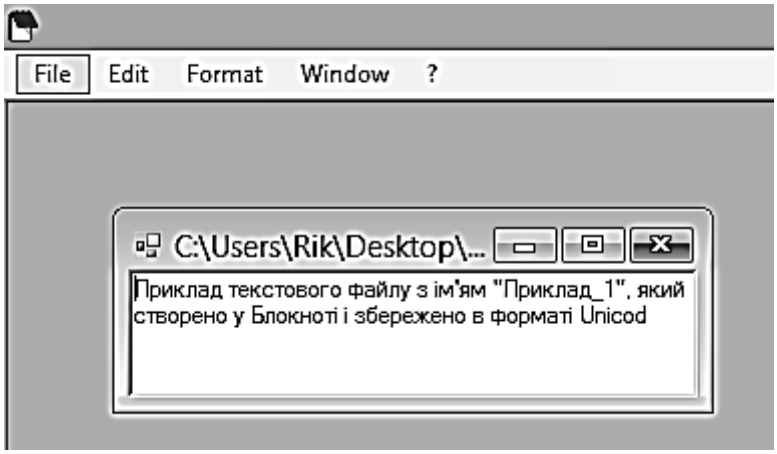


Рис. 36. Назва вікна являє собою адресу та ім'я відкритого файлу

Для коректного читання кирилиці текст в блокноті повинен бути збережений в кодуванні Unicode або ANSI. На жаль, вбудовані діалогові вікна OpenFileDialog Visual Studio .NET не містять додаткового поля, що дозволяє вибирати кодування файлу при його відкритті або збереженні, як це реалізовано, наприклад, в стандартному Блокноті.

SaveFileDialog

Крок 1. Налаштування властивостей вікна.

Для збереження файлів додаємо на форму **frmmain** елемент управління saveFileDialog1. Властивості цього діалогу в точності такі ж, як і у компонента OpenFileDialog (див. рис. 35).

Крок 2. Робота з файловими потоками в коді форми **blank**.

2.1. Перемикаємося в режим дизайну форми blank.

Переходимо в код форми blank:

// Створюємо метод Save, де як параметр оголошуємо рядок адреси файлу.

```
public void Save (string SaveFileName)
{
  // Якщо файл не обраний, повертаємося назад (з'явиться вбудоване попередження)
```

```
            if (SaveFileName == "")
            {
              return;
            }
            else
            {
        // Створюємо новий об'єкт StreamWriter і передаємо йому змінну // OpenFileName
              StreamWriter sw = new StreamWriter (SaveFileName);
              // Вміст richTextBox1 записуємо в файл
              sw.WriteLine (richTextBox1.Text);
              // Закриваємо потік
              sw.Close ();
        // Встановлюємо в якості імені документа назву збереженого файлу
              DocName = SaveFileName;
            }
          }
```

Крок 3. Додаємо обробник пункту меню Save форми frmmain:

Перемикаємося в режим дизайну форми **frmmain**. Двічі клацаємо по пункту меню Save. В шаблон методу, який з'явився, записуємо наступний код:

```
private void saveToolStripMenuItem_Click(object sender, EventArgs e)
      {
        // Можна програмно задавати доступні для огляду розширення файлів
        //openFileDialog1.Filter = "Text Files (* .txt) | * .txt | All Files (*. *) | *. *";
        // Якщо обрано діалог відкриття файлу, виконуємо умова
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
          // Перемикаємо фокус на дану форму.
          blank frm = (blank)this.ActiveMdiChild;
          // Викликаємо метод Save форми blank
          frm.Save(saveFileDialog1.FileName);
          // Вказуємо, що батьківською формою є форма frmmain
```

```
            frm.MdiParent = this;
            // Надаємо змінної FileName ім'я файлу
            frm.DocName = saveFileDialog1.FileName;
            // Властивості Text форми присвоюємо змінну DocName
            frm.Text = frm.DocName;
        }
    }
```

Крок 3. Запускаємо програму. Тепер файли можна відкривати, редагувати і зберігати.

FontDialog

Додаємо тепер можливість вибирати шрифт, його розмір і накреслення.

Крок 1. У режимі дизайну перетягнемо на форму **frmmain** з вікна ToolBox елемент управління FontDialog.

Крок 2.Переходимо в обробник пункту Font головного меню і заповнюємо його шаблон кодом, який наведено нижче:

```
    private void mnuFont_Click (object sender, System.EventArgs e)
      {
        // Перемикаємо фокус на дану форму.
        blank frm = (blank) this.ActiveMdiChild;
        // Вказуємо, що батьківською формою є форма frmmain
        frm.MdiParent = this;
        // Викликаємо діалог
        fontDialog1.ShowColor = true;
        // Зв'язуємо властивості SelectionFont і SelectionColor елемента RichTextBox
        // з відповідними властивостями діалогу
        fontDialog1.Font = frm.richTextBox1.SelectionFont;
        fontDialog1.Color = frm.richTextBox1.SelectionColor;
        // Якщо обрано діалог відкриття файлу, виконуємо умова
        if (fontDialog1.ShowDialog () == DialogResult.OK)
        {
          frm.richTextBox1.SelectionFont = fontDialog1.Font;
          frm.richTextBox1.SelectionColor = fontDialog1.Color;
```

```
    }
    // Показуємо форму
    frm.Show ();
  }
```

Крок 3. Переходимо на форму **blank**. Встановлюємо властивість Modifiers згідно рис. 37.

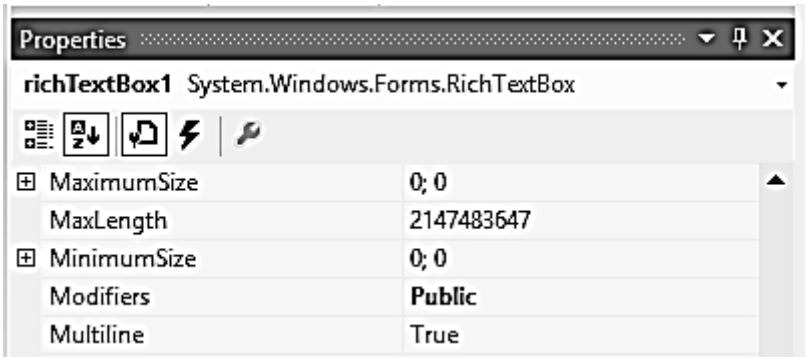


Рис. 37. Налаштування властивості Modifiers

Крок 4. Запускаємо програму. Тепер при виборі пункту Font меню Format можна міняти параметри шрифту поточного тексту (рис. 38).

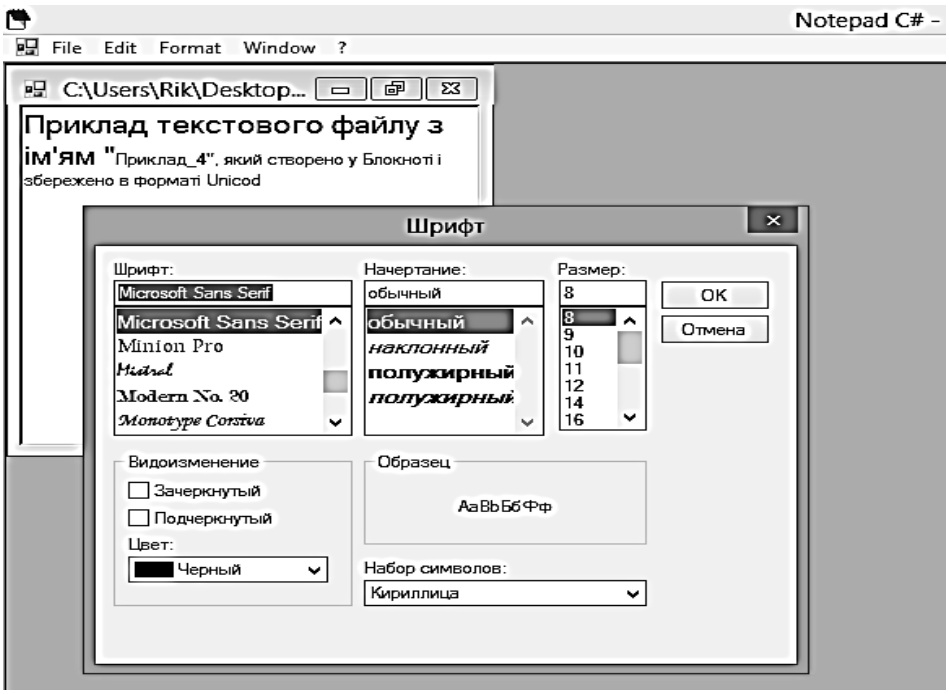


Рис. 38. Вікно вибору шрифту (пункту Font меню Format)

ColorDialog

Діалогове вікно FontDialog містить список кольорів, які можуть бути застосовані до тексту, але пропонований список обмежений. Більш цікавою видається можливість призначати користувальницький колір, який може бути визначений у великому діапазоні

Крок 1. У режимі дизайну перетягнемо на форму **frmmain** з вікна ToolBox елемент управління ColorDialog

Крок 2.Переходимо в обробник пункту Color головного меню і заповнюємо його шаблон кодом, який наведено нижче:

```
private void mnuColor_Click(object sender, EventArgs e)
    {
       blank frm = (blank)this.ActiveMdiChild;
       frm.MdiParent = this;
       colorDialog1.Color = frm.richTextBox1.SelectionColor;

       if (colorDialog1.ShowDialog() == DialogResult.OK)
       {
          frm.richTextBox1.SelectionColor = colorDialog1.Color;
       }

       frm.Show();
    }
```

Крок 3. Запускаємо програму. Тепер при виборі пункту Color меню Format можна міняти колір шрифту поточного тексту (рис. 38) в значно більшому діапазоні в порівнянні з можливістю, яку дає пункт.

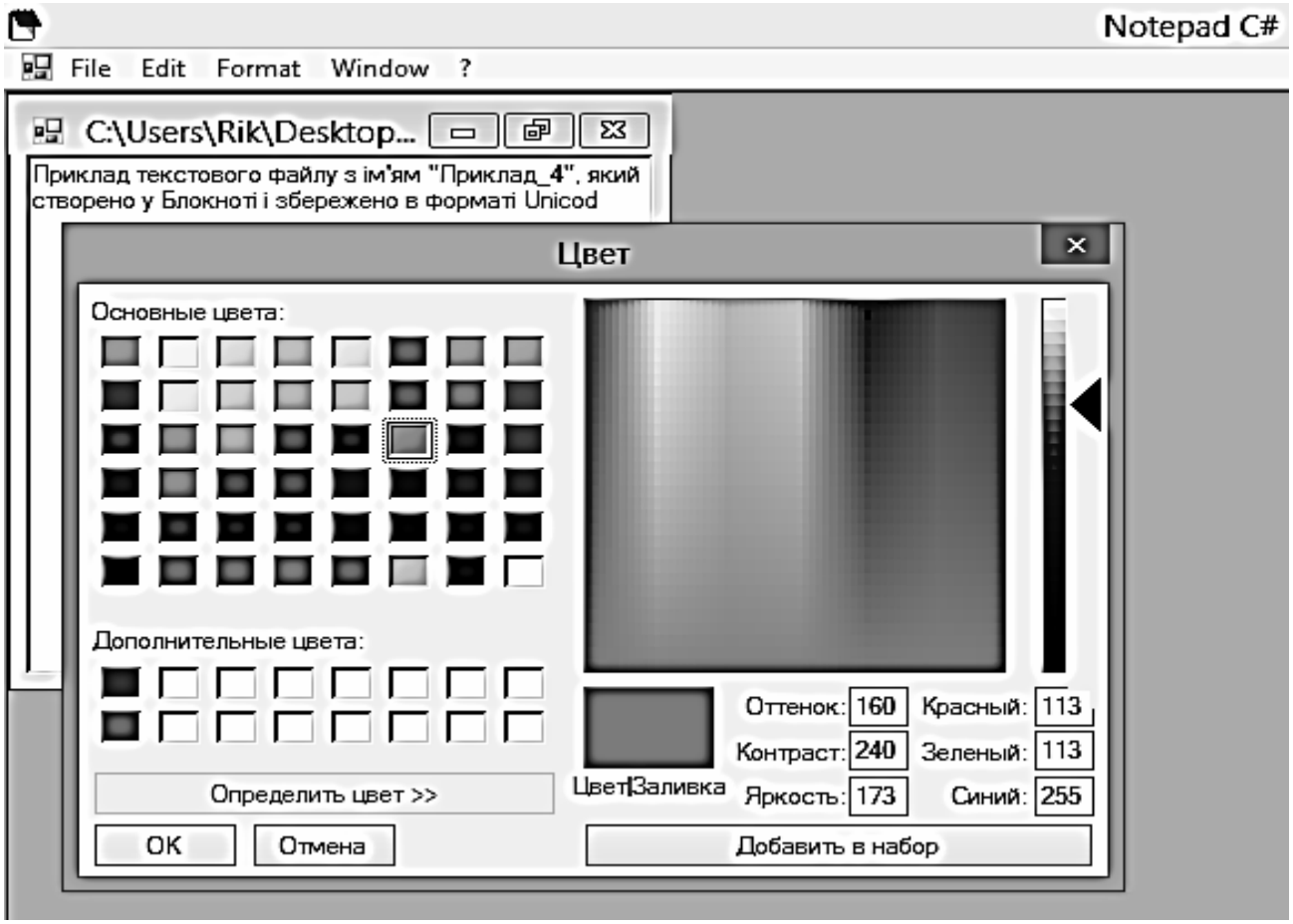


Рис. 39. Вікно вибору коліру користувача

Зверніть увагу на те, що код для ColorDialog в точності такий же, як і частина коду для властивості Color діалогу FontDialog. Це й не дивно: адже ми пов'язуємо ці діалоги з властивостями одного і того ж об'єкта - RichTextBox.

Закривання форми

У обробнику пункту Clos головного меню форми **frmmain** додаємо код:

```
private void mnuExit_Click(object sender, EventArgs e)
      {
         this.Close();
      }
```

Метод Close закриває форму і може бути призначений іншим елементам управління - наприклад, кнопці.

Порядок виконання лабораторної роботи

Загальна частина.

1. Набрати, відкомпілювати і запустити на виконання першої програми (приклад 1), яка була наведена в розділі «Основні положення» даної лабораторної роботи.

Індивідуальна частина.

Повторити всі кроки розробки програми Блокнот (приклад 2). При цьому слід до імен (властивість Name) відповідних пунктів головного меню додати префікс з перших трьох букв свого прізвища. Наприклад, було – mnuFile, стало – mnuFile_Fan, якщо прізвище студента є Фандорін.

Зміст звіту

1. Титульний лист.
2. Цілі лабораторного заняття і вказівка, які навички та вміння передбачається отримати в результаті його виконання.
3. Тексти відповідних обробників подій індивідуального завдання, які пов’язані з пунктами головного і контекстного меню.
4. Висновки.

Контрольні питання

1. Які групи елементів надає середа Visual Studio.NET, щоб забезпечить взаємодію між користувачем і програмою, яка розробляється? Опишіть групу командних об'єктів.
2. Дайте порівняння елементів TextBox, RichTextBox.
3. Опишіть технологію створення головного меню MDI-додатка.
4. Для чого і яким чином застосовується компонент OpenFileDialog?
5. Опишіть технологію створення контекстного меню MDI-додатка.