Тема 11. Програмування графіки

(розробка Windows-додатків з елементами графіки)

Мета лекції – отримати теоретичні знання та практичні навички роботи щодо створення MDI- і SDI-додатків з елементами графіки.

Вона сприяє напрацюванню наступних **компетентностей** у відповідності до Національної рамки кваліфікації:

Знання:

взаємодія форм;

модальні та немодальні форми;

передача інформації між формами;

запуск програм з програми,

Уміння:

використовувати стандартні компоненти Designer Forms;

розробляти MDI- додатки з стандартним Windows інтерфейсом.

Комунікації:

аргументована взаємодія з клієнтами та замовниками при виборі технології розробки елементів інтерфейсу Windows додатків;

робота в команді, яка проектує інтерфейс користувача певного Windows додатка.

Автономність і відповідальність:

самостійне формулювання рекомендацій щодо вибору технології створення інтерфейсу користувача;

здатність обґрунтувати доцільності застосування певного набору стандартних компонентів для створення відповідного інтерфейсу Windows додатка.

Основні положення

Взаємодія форм

Звичайний Windows-додаток завжди містить кілька форм. Одні відкриваються в процесі роботи, інші закриваються. У кожний поточний момент на екрані може бути відкрита одна або декілька форм, користувач може працювати з однією формою або перемикатися по ходу роботи з однієї на іншу.

Слід чітко розрізняти процес створення форми - відповідного об'єкта, що належить класу Form або спадкоємцю цього класу, - і процес показу форми на екрані.

Для показу форми служить метод Show цього класу, що викликається відповідним об'єктом; для приховування форми використовується метод Hide. Реально методи Show і Hide змінюють властивість Visible об'єкта, так що замість виклику цих методів можна міняти значення цієї властивості, встановлюючи його або в true, або в false.

Зауважте різницю між приховуванням і закриттям форми - між методами Hide і Close. Перший з них робить форму невидимою, але сам об'єкт залишається живим і неушкодженим. Метод Close відбирає у форми її ресурси, роблячи об'єкт відтепер недоступним; викликати метод Show після виклику методу Close неможливо, якщо тільки не створити об'єкт заново. Відкриття та показ форми завжди означає одне і те ж - виклик методу Show. У форми є метод Close, але немає методу Open. Форми, як і всі об'єкти, створюються при виклику конструктора форми при виконанні операції new.

Форма, що відкривається в процедурі Main при виклику методу Run, називається головною формою проекту. Її закриття призводить до закриття всіх інших форм і завершення Windows-програми. Завершити додаток можна і програмно, викликавши в потрібний момент статичний метод Exit класу Application. Закриття інших форм не приводить до завершення проекту. Найчастіше головна форма проекту завжди відкрита, в той час як інші форми відкриваються і закриваються (ховаються). Якщо ми хочемо, щоб в кожний поточний момент була відкрита тільки одна форма, то потрібно вжити певних заходів, щоб при закритті (приховуванні) форми відкривалася інша. Інакше можлива клінчевая ситуація - всі форми закриті, зробити нічого не можна, а додаток не завершено. Звичайно, вихід завжди є - завжди можна натиснути магічну трійку клавіш CTRL + ALT + DEL і завершити будь-який додаток.

Можна створювати форми як об'єкти класу Form. Однак такі об'єкти досить рідкісні. Найчастіше створюється спеціальний клас FormX - спадкоємець класу Form. Так, зокрема, відбувається в Windows-додатку, створюваному за замовчуванням, коли створюється клас Form1 - спадкоємець класу Form. Так відбувається в режимі проектування, коли в проект додається нова форма з використанням пункту меню Add Windows Form. Як правило, кожна форма в проекті - це об'єкт власного класу. Можлива ситуація, коли знову створювана форма

багато в чому повинна бути схожою на вже існуючу, і тоді клас нової форми може бути зроблений спадкоємцем класу форми існуючої. Спадкування форм ми розглянемо докладніше трохи пізніше.

Модальні та немодальні форми

Первинним є поняття модального і немодального вікна. Вікно називається модальним, якщо не можна закінчити роботу у відкритому вікні доти, поки воно не буде закрито. Модальне вікно не дозволяє, якщо воно відкрито, тимчасово переключитися на роботу з іншим вікном. Вийти з модального вікна можна, тільки закривши його. Немодальні вікна допускають паралельну роботу у вікнах. Форма називається модальної або немодального залежно від того, яке її вікно.

Метод Show відкриває форму як немодального, а метод ShowDialog - як модальну. Назва методу відображає основне призначення модальних форм - вони призначені для організації діалогу з користувачем, і поки діалог не завершиться, покидати форму забороняється.

Передача інформації між формами

Часто багато форми повинні працювати з одним і тим же об'єктом, виробляючи над ним різні операції. Як це реалізується? Звичайна схема така: об'єкт створюється в одній з форм, найчастіше, у головній. При створенні наступної форми глобальний об'єкт передається конструктору нової форми в якості аргументу. Природно, одне з полів нової форми має представляти посилання на об'єкт відповідного класу, так що конструктору залишиться тільки зв'язати посилання з переданим йому об'єктом. Зауважте, все це ефективно реалізується, оскільки об'єкт створюється лише один раз, а різні форми містять посилання на цей єдиний об'єкт.

Якщо такий глобальний об'єкт створюється в головній формі, то можна передавати не об'єкт, необхідний іншим формам, а містить його контейнер - головну форму. Це зручніше, оскільки при цьому можна передати кілька об'єктів, можна не замислюватися над тим, який об'єкт передавати тій чи іншій формі. Мати посилання на головну форму часто

необхідно, хоча б для того, щоб при закритті будь-якої форми можна було б відкривати головну, якщо вона була попередньо прихована.

Уявімо собі, що кілька форм повинні працювати з об'єктом класу Books. Нехай в головній формі такий об'єкт оголошений:

public Books myBooks;

У конструкторі головної форми такий об'єкт створюється:

myBooks = new Books (max_books);

де max_books - задана константа. Нехай ще в головній формі оголошена форма - об'єкт класу NewBook:

public NewBook form2;

При створенні об'єкта form2 його конструктору передається посилання на головну форму:

form2 = new NewBook (this);

Клас NewBook містить поля:

private Form1 mainform;

private Books books;

а його конструктор наступний код:

mainform = form;

books = mainform.myBooks;

Тепер об'єкту form2 доступні раніше створені об'єкти, що задають книги і головну форму, так що в обробнику події Closed, що виникає при закритті форми, можна задати код:

```
private void NewBook_Closed (object sender, System.EventArgs e)
{
mainform.Show ();
}
```

що відкриває головну форму.

Продовжимо роботу над додатком Блокнот (дивись Приклад 2 попереднього лабораторного заняття).

Додавання пункту SaveAs… в меню File програми Блокнот.

Запускаємо програму Блокнот. Тепер файли можна відкривати, редагувати і зберігати. Однак, при збереженні внесених змін у вже раніш збереженому файлі і знову відкритим, замість його перезапису знову з'являється вікно SaveFileDialog.

Змінимо програму так, щоб можна було зберігати і перезаписувати файл.

Крок 1. У конструкторі форми **frmmain** після InitializeComponent відключимо доступність пункту меню Save:

mnuSave.Enabled = false;

Крок 2. Перемикаємося в режим дизайну форми **frmmain** і додаємо пункт меню Save As після пункту Save.

Встановлюємо наступні властивості цього пункту: Name - mnuSaveAs, Shortcut - CtrlShiftS, Text - Save &As.

Крок 3. У обробнику Save As вставляємо вирізаний обробник пункту Save і додаємо включення доступності Save:

mnuSave.Enabled = true;

Крок 4.Зберігати зміни потрібно як у щойно збережених документах, так і в документах, створених раніше і відкритих для редагування. Тому додаємо в метод Open включення доступності пункту меню Save:

```
private void mnuOpen_Click (object sender, System.EventArgs e)
{
  mnuSave.Enabled = true;
}
```

Крок 5. У обробнику пункту Save додаємо виклик методу Save форми blank:

```
private void mnuSave_Click (object sender, System.EventArgs e)
  {
      . . .
      // Викликаємо метод Save форми blank
      frm.Save (frm.DocName);
  }
```

Запускаємо програму. Тепер, якщо ми працюємо з незбережених документом, пункт Save неактивний, після збереження він стає активним і, крім того, працює сполучення клавіш Ctrl + S. Можна зберегти копію поточного документа, знову скориставшись пунктом меню Save As.

Збереження файлу при закритті форми

Всякий раз, коли ми закриваємо документ Microsoft Word, в який внесли зміни, повинно з'являється вікно попередження, що пропонує зберегти документ. Додаймо аналогічну функцію в наш додаток.

Крок 1. У класі **blank**: System.Windows.Forms.Form форми **blank** створюємо змінну, яка буде фіксувати факт внесення зміни в поточний документа:

public bool IsChanged = false;

Крок 2 У вікні властивостей перемикаємо на події форми, клацнувши на значок з блискавкою.

В поле події TextChanged двічі клацаємо і переходимо в шаблон обробника події, в який записуємо наступний код, який фіксує зміну в поточному документі:

```
private void richTextBox1_TextChanged(object sender, EventArgs e)
      {
         IsChanged = true;
      }
```

Крок 3. В обробник методів Save і Save As форми **frmmain** додаємо значення цієї змінної, яка дорівнює - false.

Таким чином, якщо в поточному файлі були якісь зміни, а потім він був збережений, то при повторному його відкритті признак зміни IsChanged повинен бути в протилежному стані.

```
private void mnuSave_Click (object sender, System.EventArgs e)
    {
...
        frm. IsChanged = false;
    }

  private void mnuSaveAs_Click (object sender, System.EventArgs e)
    {
...
frm. IsChanged = false;
    }
```

Крок 4. Переходимо в режим дизайну форми **blank** і у вікні властивостей перемикаємо на події форми, клацнувши на значок з блискавкою.

В поле події FormClosing двічі клацаємо і переходимо в шаблон обробника події, в який записуємо наступний код:

```
private void blank_FormClosing(object sender, FormClosingEventArgs e)
        {
            // Якщо змінна IsSaved має значення true, тобто новий доку-мент
            // був збережений (Save As) або у відкритому документі були
            // збережені зміни (Save), то виконується умова
            if (IsSaved == true)
                // З'являється діалогове вікно, що пропонує зберегти доку-мент.
                if (MessageBox.Show("Ви бажаєте зберегти зміни у фалі " + this.DocName + "?",
                    "Message", MessageBoxButtons.YesNo,
                    MessageBoxIcon.Question) == DialogResult.Yes)
                // Якщо була натиснута кнопка Yes, викликаємо метод Save
                {
                    this.Save(this.DocName);
                }
        }
```

Крок 5. Запускаємо програму. Створюємо новий документ (рис. .40), зберігаємо його, далі закриваємо форму. При цьому ніякого попередження не виникає, тому що застосовувався пункт меню Save As.

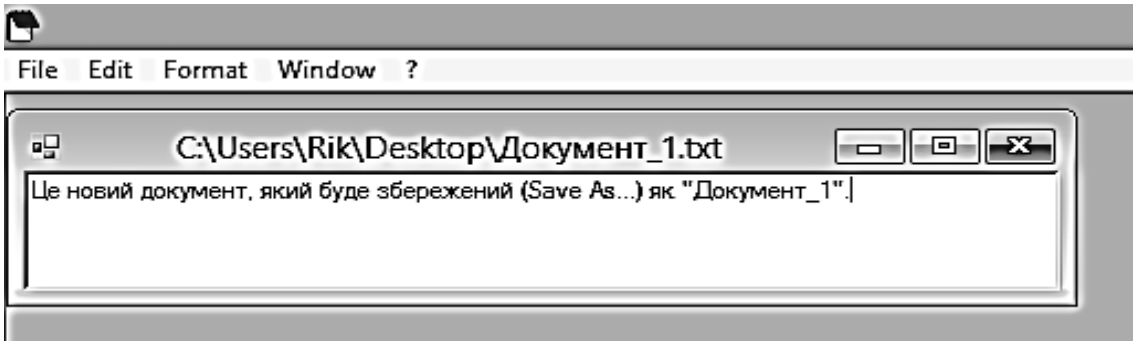

Рис. 40. Новий документ

Відкриваємо документ (пункт Open) і зразу ж закриваємо форму. Ніякого попередження не виникає, тому що в документі відсутні зміни.

Знову відкриваємо документ і дописуємо в нього текст «Виконувач - Студент Фандорін». Після чого натискаємо кнопку закриться форми. Як результат – з'являється вікно попередження (рис. 41).

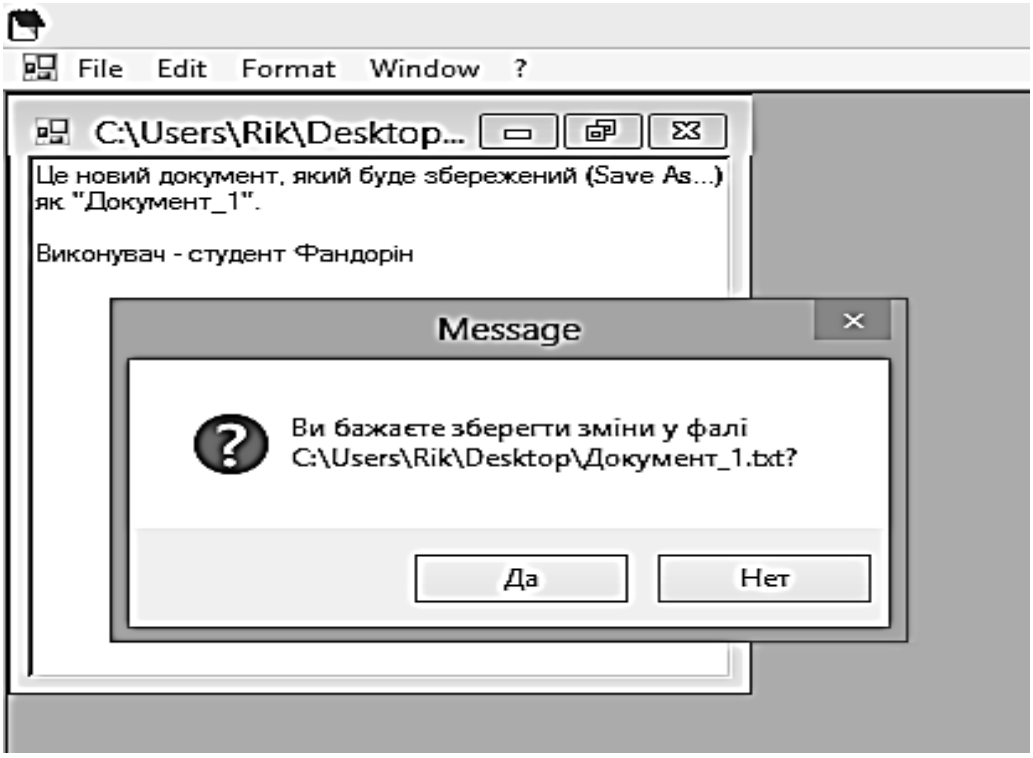

Рис. 41. Вікно попередження

Натискуємо кнопку «Да» і знову відкриваємо документ. Бачимо, що останні зміни в документі були збережені.

StatusBar

Елемент управління StatusBar застосовується в програмах для виведення інформації в рядок стану - невелику смужку, розташовану унизу програми.

Додаймо до додатка Notepad C # рядок стану, на якій здійснюється підрахунок символів, що вводять і виводиться системний час.

Крок 1. Додаємо на форму blank елемент управління StatusBar (рис. 42).

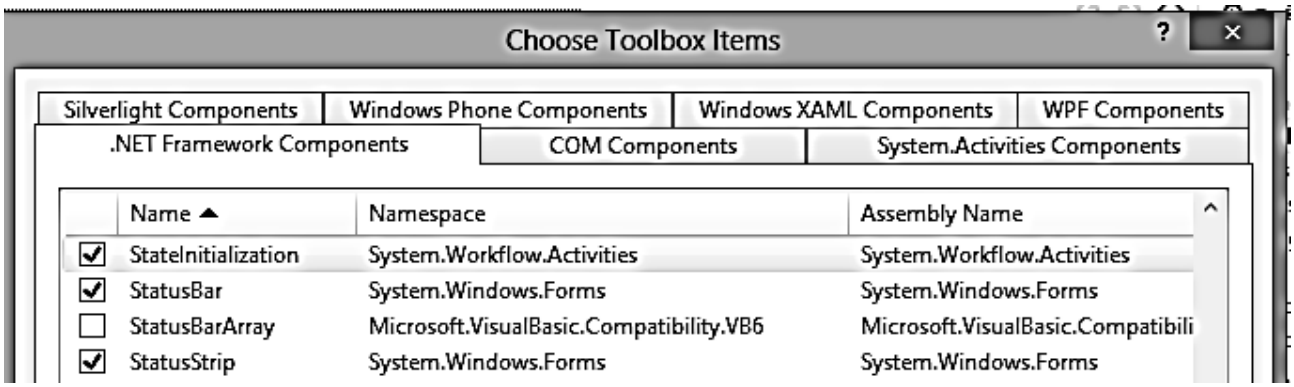

Рис. 42. Додавання на форму елемента управління StatusBar

Видаляємо вміст поля властивості Text.

Крок 2. У полі властивості Panels (рис. 43) клацаємо на кнопку (...).

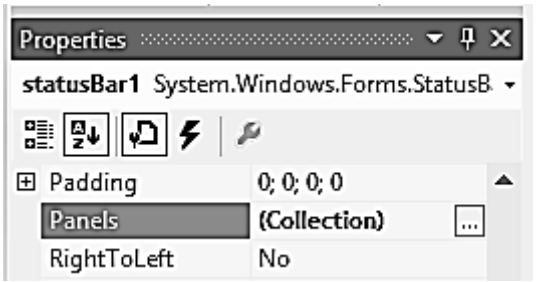

Рис. 43. Поле властивості Panels

Відкривається StatusBarCollectionEditor, в якому ми створюємо панелі для відображення.

Крок 3. Створіть дві панелі, двічі клацаючи на кнопці Add, і встановіть їм такі властивості (змінені значення виділяються жирним шрифтом) (рис. 44 і рис. 45):

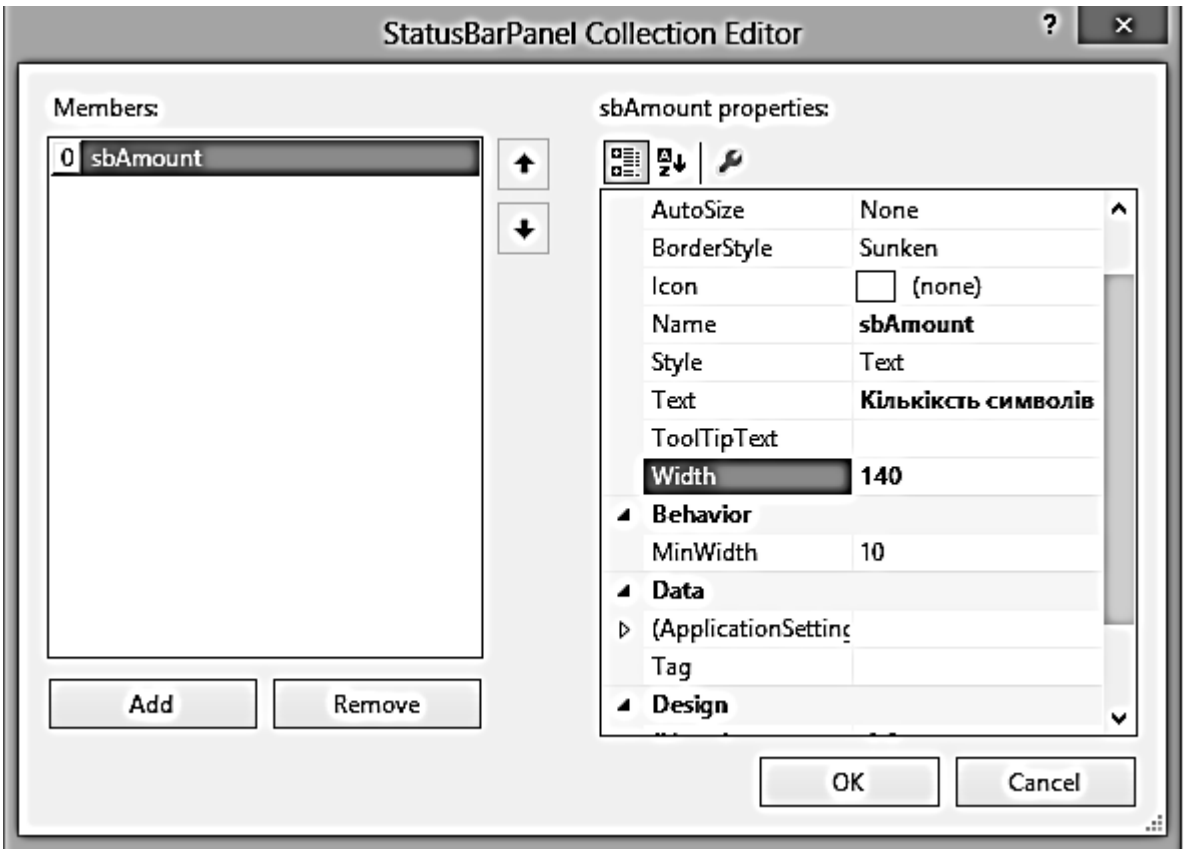

Рис. 44. Властивості Name, Text, Width панелі sbAmount

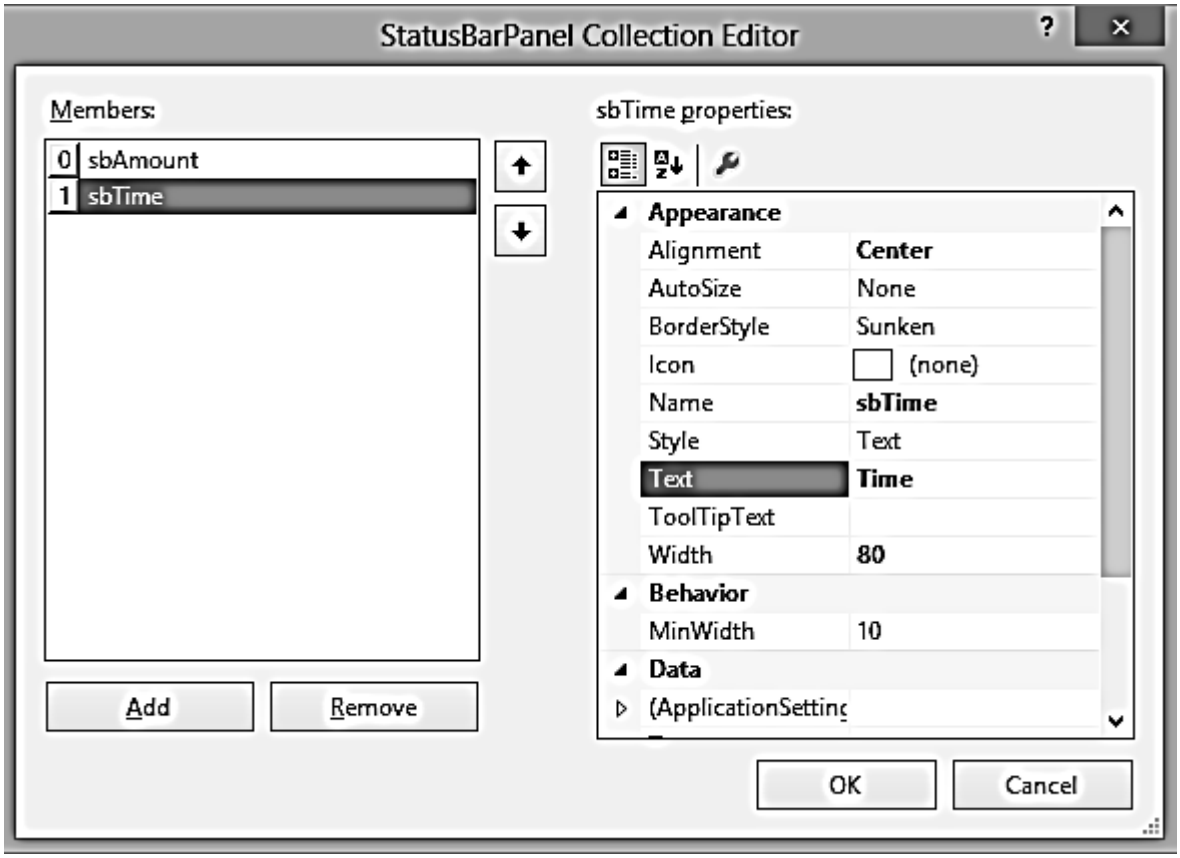

Рис. 45. Властивості Alignment, Name, Text, Width панелі sbTime

Значення деяких властивостей елемент управління StatusBar наведено в таблиці 3.

Таблиця 3

Властивості елементу управління StatusBar

Властивість	Значення
Alignment	Вирівнювання вмісту властивості Text на панелі
AutoSize	Зміна розмірів панелі по вмісту
BorderStyle	Зовнішній вигляд панелі - втоплена, піднесена або без виділення
Icon	Додавання іконки
Style	Стиль панелі
Text	Текст, наявний на панелі
ToolTipText	Спливаюча підказка - з'являється при наведенні курсора на панель
Width	Ширина панелі в пікселях
Name	Назва панелі для звернення до неї в коді

Властивості панелі, призначені у вікні редактора StatusBarCollectionEditor, можна змінювати в коді (саме так буде зроблено нижче у кроці 5).

Після завершення роботи над панелями закриваємо редактор.

Крок 4. Властивості ShowPanels елементу управління StatusBar встановлюємо значення True. На формі негайно відображаються дві панелі (рис. 46).

Рис. 46. Панель StatusBar

Крок 5. Виділяємо елемент управління RichTextBox форми **Blank**, у вікні його властивостей перемикаємо на подію TextChanged і доповнюємо раніш створений обробник новим кодом (виділено жирним шрифтом):

```
private void richTextBox1_TextChanged(object sender, EventArgs e)
  {
  IsChanged = true;

// Властивості Text панелі sbAmount програмно змінюємо
//                                    на «Кількість символів: »
 sbAmount.Text = "Кількість символів: " +
richTextBox1.Text.Length.ToString();
}
```

Властивість Text панелі sbAmount ми змінюємо програмно: навіть якби ми нічого не написали у вікні редактора StatusBarCollectionEditor, при виникненні події TextChanged на панелі з'явиться відповідний напис.

Крок 6. Займімося тепер другий панеллю - тієї, на яку будемо виводити системний час. У конструкторі форми **blank** додаємо код (виділено жирним шрифтом):

```
public blank()
        {
          InitializeComponent();
          // Властивості Text панелі sbTime встановлюємо системний час,
          // Конвертіровав його в тип String
          sbTime.Text                                                    =
Convert.ToString(System.DateTime.Now.ToLongTimeString());
          // В тексті підказки виводимо поточну дату
          sbTime.ToolTipText                                             =
Convert.ToString(System.DateTime.Today.ToLongDateString());
        }
```

Крок 7. Запускаємо програму. Результат надано на рис. 47.

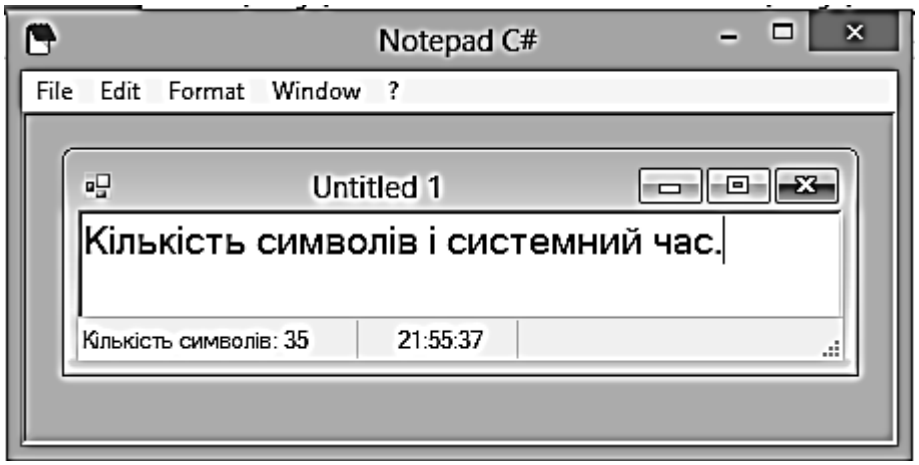

Рис. 47. Панель StatusBar після вводу тексту

Компоненти Label, LinkLabel і PictureBox.

Програми, як правило, містять пункт головного меню "Про програму", де в окремому вікні поміщається логотип компанії, ліцензійна угода, гіперпосилання на сайт розробника та інша інформація. Створимо подібну форму, використовуючи елементи управління - Label, LinkLabel і PictureBox.

Крок 1. Додаємо в наш проект нову форму і назвемо її About.cs. Встановимо наступні властивості форми:

Name → About

FormBorderStyle → FixedSingle

MaximizeBox → False

MinimizeBox → False

Size → 318; 214

Text → About Notepad C#

Крок 2. Додаємо на форму About елемент управління PictureBox - він представляє собою підкладку, що розміщується на формі, яка може містити в собі малюнки для відображення.

У полі властивості Image клацаємо на кнопку (...) і вибираємо малюнок (з файлу роздаткового матеріалу, або з Інтернету)….\ Icon \ logo.gif.

Оскільки logo.gif є анімований малюнком, елемент PictureBox починає відтворювати анімацію відразу ж, навіть в режимі дизайну (рис.48).

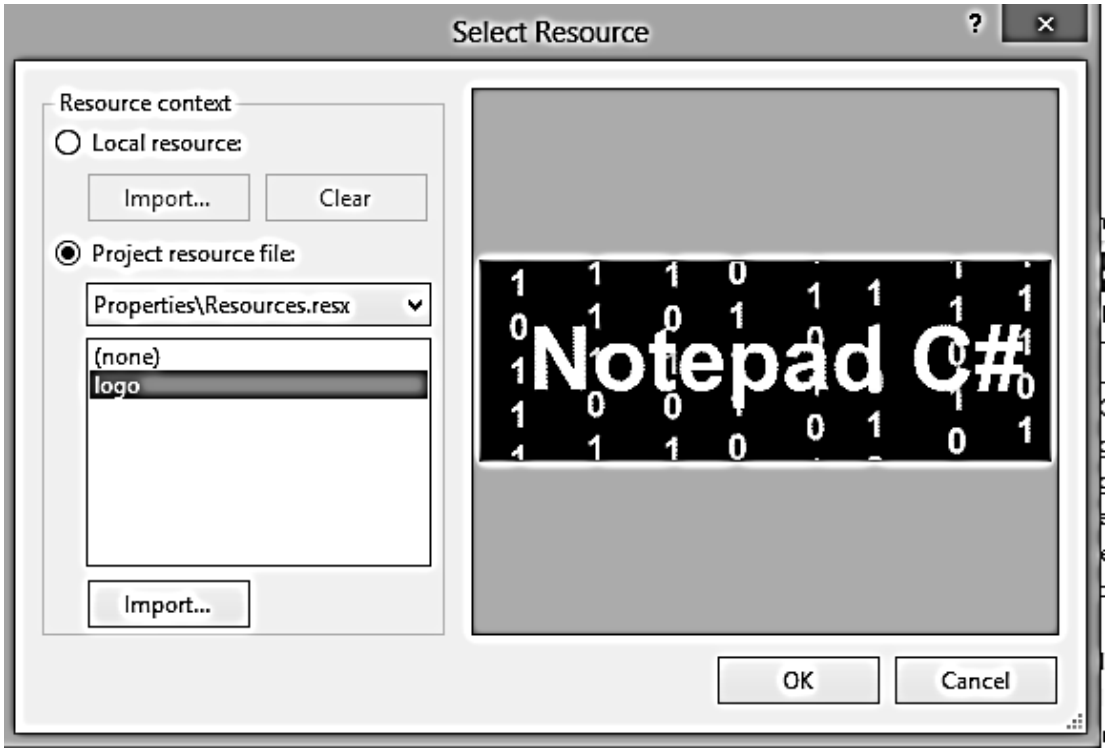

Рис. 48. Вікно імпорту файлу logo.gif з анімований малюнком

Крок 3. З вікна ToolBox перетягнемо на форму елементи: Button, Label і LinkLabel.

У полі властивості Text кнопки введемо &OK.

Елемент Label призначений для розміщення на формі написів, які в готовому додатку будуть доступні тільки для читання. У полі властивості Text введемо «Notepad C# 2015. Розробник – Фандорін В.В.».

Крок 4. Елемент LinkLabel відображає текст на формі в стилі web-посилань і зазвичай використовується для створення навігації між

формами або посилання на сайт. У полі Text цього елемента вводимо адресу гіпотетичного сайту - http://www.mo.hneu.edu.ua. Користувач буде переходити на сайт, натискаючи на це посилання, тому реалізуємо перехід за гіперпосиланням для події Click.

Крок 5. У вікні Properties елементу LinkLabel переходимо на вкладку з обробниками подій і клацаємо двічі на подію Click (рис.49).

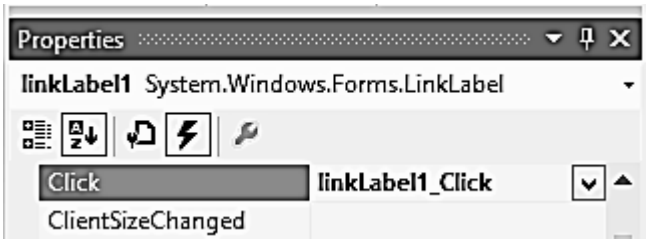

Рис. 49. Виклик шаблону обробника події Click

У шаблон, який з'явився, додаємо обробник:

```
private void linkLabel1_Click(object sender, EventArgs e)
        {
            // Додаємо блок для обробки виключень - з різних причин
            // користувач може не отримати доступу до ресурсу.
            try
            {
                // Викликаємо метод VisitLink, визначений нижче
                VisitLink();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex + "Неможливо відкрити сайт");
            }
        }

        // Створюємо метод VisitLink
        private void VisitLink()
        {
            // Змінюємо колір відвіданого посилання, програмно
            // звертаючись до властивості LinkVisited елемента
LinkLabel
```

```
            linkLabel1.LinkVisited = true;
            // Викликаємо метод Process.Start method для запуску брау-
зера,
            // встановленого за замовчуванням, і відкриття посилання

System.Diagnostics.Process.Start("http://www.mo.hneu.edu.ua");
        }
```

Крок 6. Підключаємо обробник події кнопки OK (форми About), який буде забезпечувати закривання форми:

```
    private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
```

Крок 7. У пункті головного меню About Programm .форми **frmmain** додаємо процедуру виклику форми About:

Для відкриття форми в модальному режимі використовується метод ShowDialog:

```
private void mnuAbout_Click(object sender, EventArgs e)
    {
        // Створюємо новий екземпляр форми About
        About frm = new About();
        frm.ShowDialog();
    }
```

Крок 7. Запускаємо програму (рис. 50).

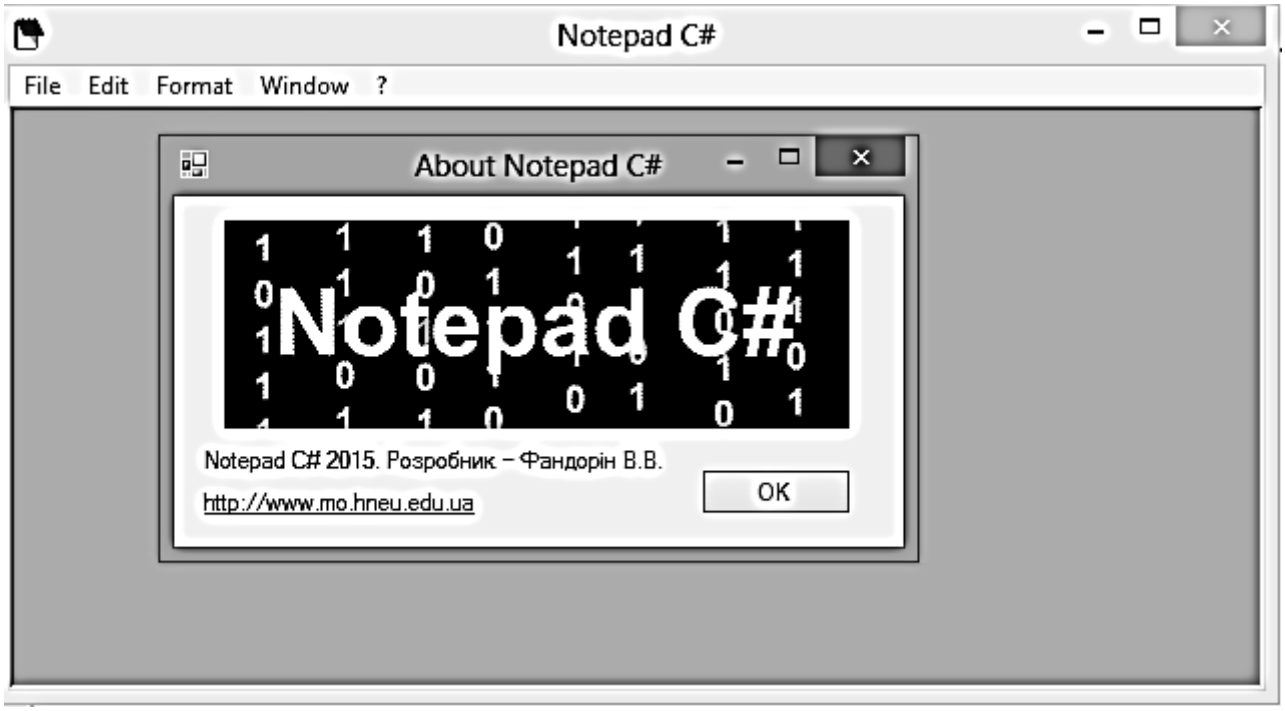

Рис. 50. Вікно About Notepad C#

Панель інструментів (компоненти ToolBar і ImageList)

Панелі інструментів ToolBar містять набори кнопок, як правило, дублюючих пункти головного меню. У графічних програмах панелі інструментів – це основний засіб робот. Нижче наведена технологія створення подібних кнопок на прикладу поточного додатка.

Крок 1. Відкриємо додаток Notepad C# і перетягнемо з вікна ToolBox елемент управління ToolBar. При необхідності потрібно доповнити вікно ToolBox цим елементом.

На кнопках панелі зазвичай розташовуються іконки, тому, перш ніж ми почнемо займатися ними, слід заздалегідь подумати про малюнки на цих кнопках.

Крок 2. Додаймо на форму **frmmain** елемент управління ImageList, який застосовується для зберігання малюнків, що можуть бути використані для оформлення елементів управління додатка.

Клацнемо в поле властивості Images елемента ImageList (рис. 51).

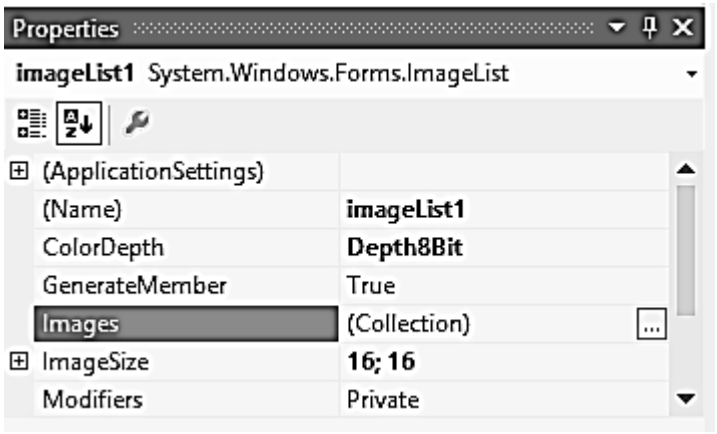

Рис. 51. Властивість Images елемента ImageList

Розкриємо вікно редактора колекцій (кнопка (Collection) ...).

Додаймо файли іконок (рис. 52), послідовно натискаючи кнопку Add і вибираючи директорію - … \ Icon (дивись роздатковий файл).

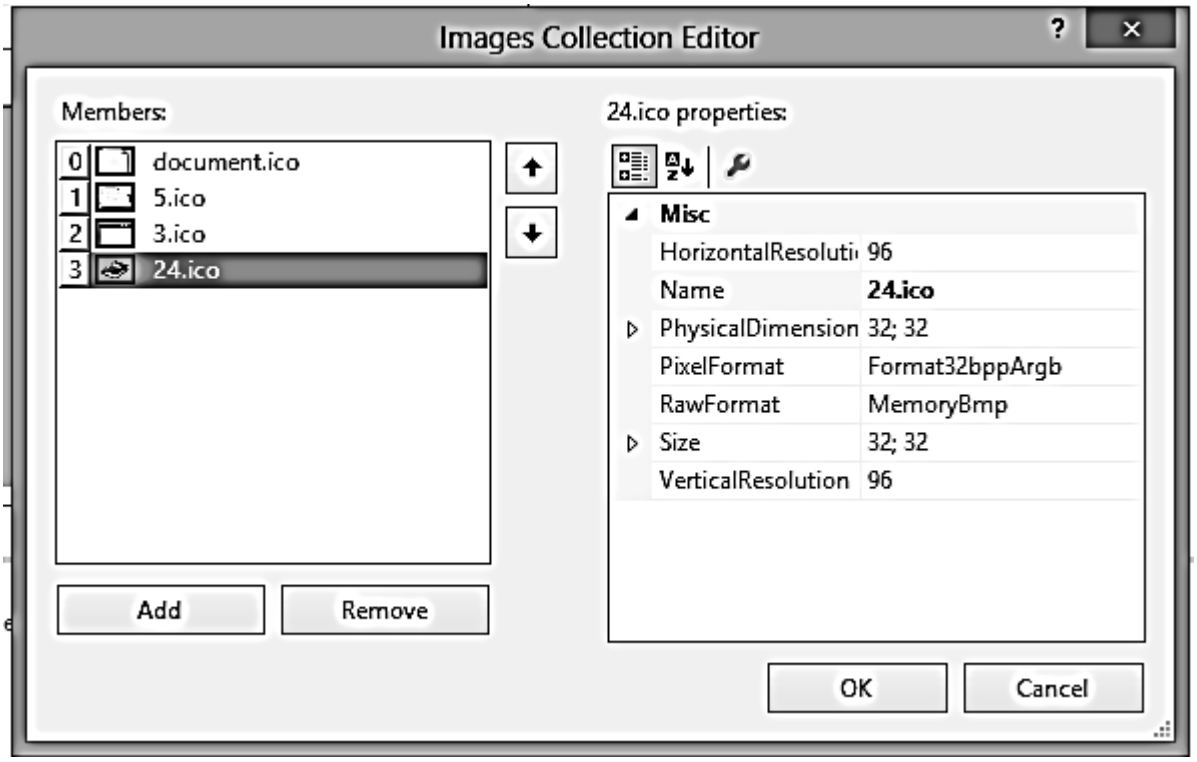

Рис. 52. Вікно редактора колекцій

Можна вибрати відповідні зображення на своєму комп’ютері або скачати з Інтернету. У будь-якому випадку, підібравши іконки, завершуємо роботу з редактором ImageCollectionEditor, натискаючи OK.

Крок 3. Створимо кнопки панелі інструментів, що дублюють дії пунктів меню New, Open, Save, ?.

Виділяємо елемент ToolBar (форма **frmmain**).

Властивості Name встановимо значення toolBarMain, а в поле властивості ImageList виберемо imageList1.

Крок 4. Запустимо редактор ToolBarButton Collection Editor для створення кнопок. Для цього потрібно натиснути кнопку (...) в поле властивості Buttons (рис. 53).

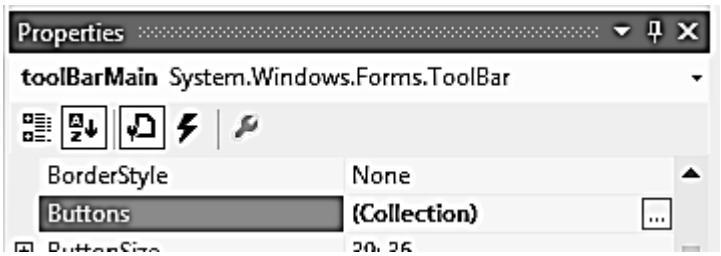

Рис. 53. Запуск редактора ToolBarButton Collection Editor

Як результат – з'являється вікно відповідного редактора. Для додавання кнопок в цьому редакторі теж слід натискати кнопку Add. Створимо чотири кнопки, встановивши для них такі властивості (таблиця 4):

Таблиця 4

Властивості кнопок панелі інструментів ToolBarButton

Name	Image Index	ToolTipText
tbNew	0	Create New
tbOpen	1	Open
tbSave	2	Save
tbAbout	3	About

Властивість Name встановлює назву кнопки для звернення до неї в коді. Властивість Image Index визначає зображення на кнопці, а в поле ToolTipText вводимо текст підказки, яка буде з'являтися при наведенні курсору на кнопку.

Результат наведено на рис. 54.

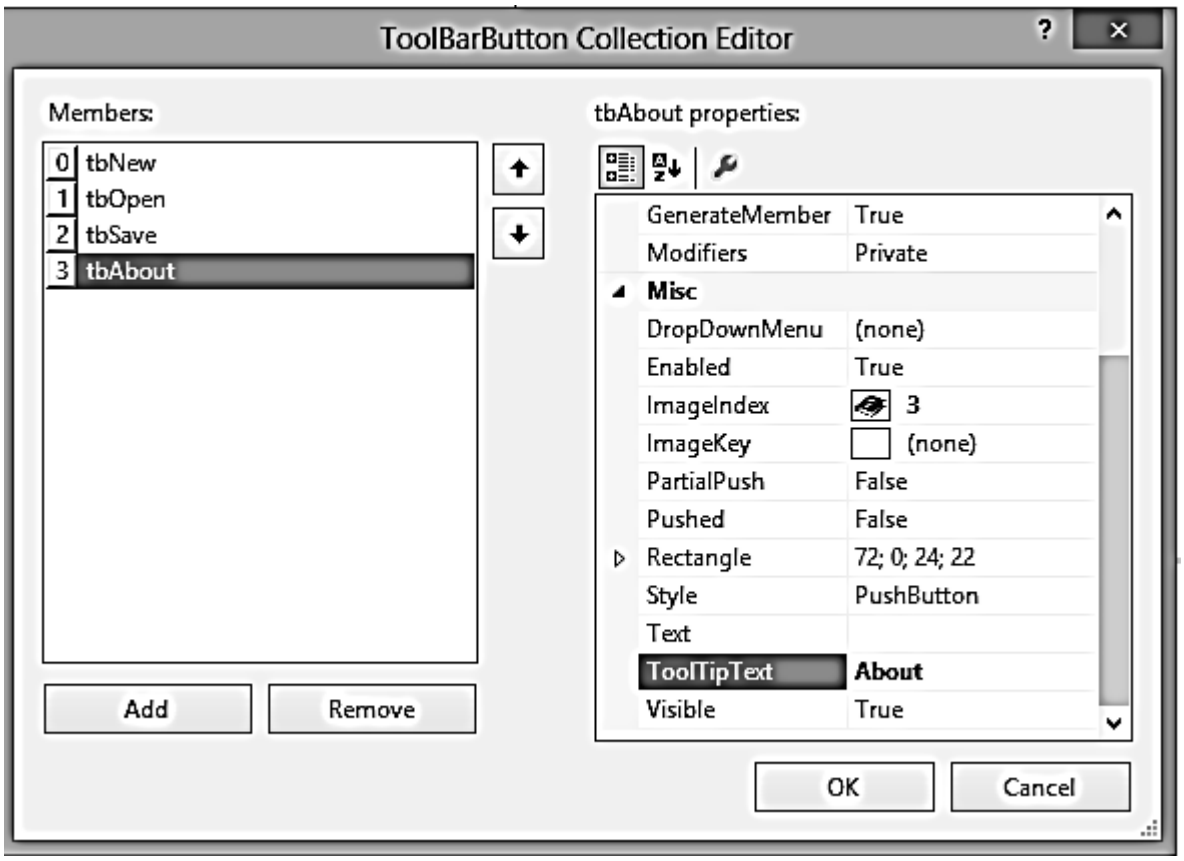

Рис. 54. Редактор ToolBarButton Collection Editor

Крок 5. Створення функціональності кнопок.

Завершивши роботу з редактором кнопок в режимі дизайну форми frmmain, двічі клацаємо на ToolBar і переходимо в шаблон коду.

```
private void toolBarMain_ButtonClick(object sender,
                                     ToolBarButtonClickEventArgs e)
    {

    }
```

Для створення функціональності кнопок пов'язуємо подію Click заданої кнопки с відповідним оброблювачем пунктів меню:

```
private void toolBarMain_ButtonClick(object sender,
                                     ToolBarButtonClickEventArgs e)
    {
        // Create New
        if (e.Button.Equals(tbNew))
        {
            mnuNew_Click(this, new EventArgs());
        }
        // Open
        if (e.Button.Equals(tbOpen))
        {
            mnuOpen_Click(this, new EventArgs());
        }
        // Save
        if (e.Button.Equals(tbSave))
        {
            mnuSave_Click(this, new EventArgs());
        }
        // About
        if (e.Button.Equals(tbAbout))
        {
            mnuAbout_Click(this, new EventArgs());
        }
    }
```

Запускаємо програму. Кнопки панелі інструментів дублюють пункти меню, а при наведенні на них з'являються підказки (рис. 55).

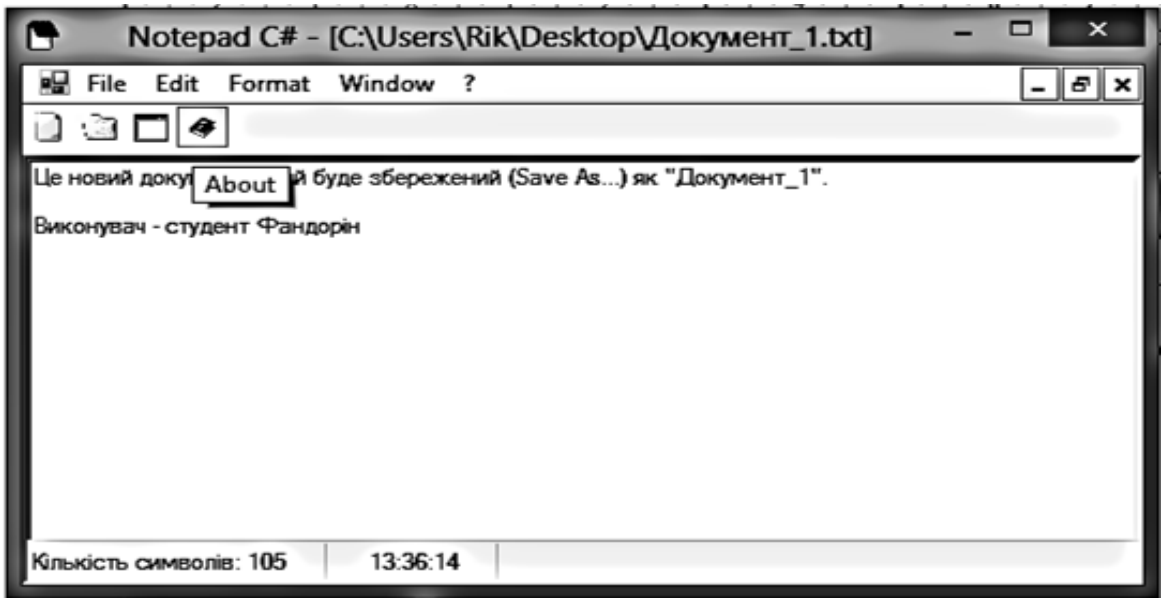

Рис. 55. Заключний результат роботи програми

Завдання.

1. Повторити всі кроки продовження розробки програми Блокнот. які була наведені в розділі «Основні положення» даної лабораторної роботи.

Індивідуальна частина.

Додати в головне меню програми пункт «Про автора» з двома-трьома підпунктами (назва вибрати самостійно), в яких розповісти про себе і свої захоплення. Текст проілюструвати фотографіями і посиланнями на відповідні інтернет-сайти.

Контрольні питання

1. Як здійснюється взаємодія форм?.
2. Дайте порівняння модальних і немодальних форм.
3. Як здійснюється передача інформації між формами?

.

4. Для чого і яким чином застосовується компонент SaveFileDialog?
5. Опишіть технологію створення елементів управління StatusBar.