Лабораторна робота 3.
Створення Windows-застосувань мовою C#

Мета: набути умінь і навичок роботи зі створення Windows-застосувань на мові C# в середовищі розробки Microsoft Visual Studio

Призначення: ознайомлення з технологією створення Windows-застосувань в середовищі Microsoft Visual Studio .

Приклад виконання завдань

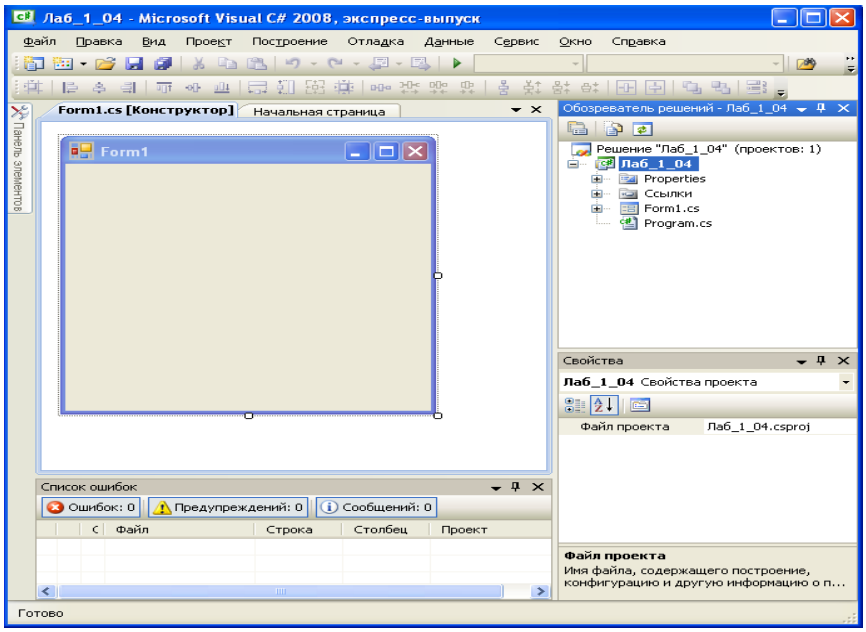
Як приклад, розглянемо такі завдання:

1. Написати Windows-проект на мові C# у Visual Studio і включити до нього два текстових вікна для введення чисел, кнопку для розрахунку суми введених чисел і мітку для виведення результату.

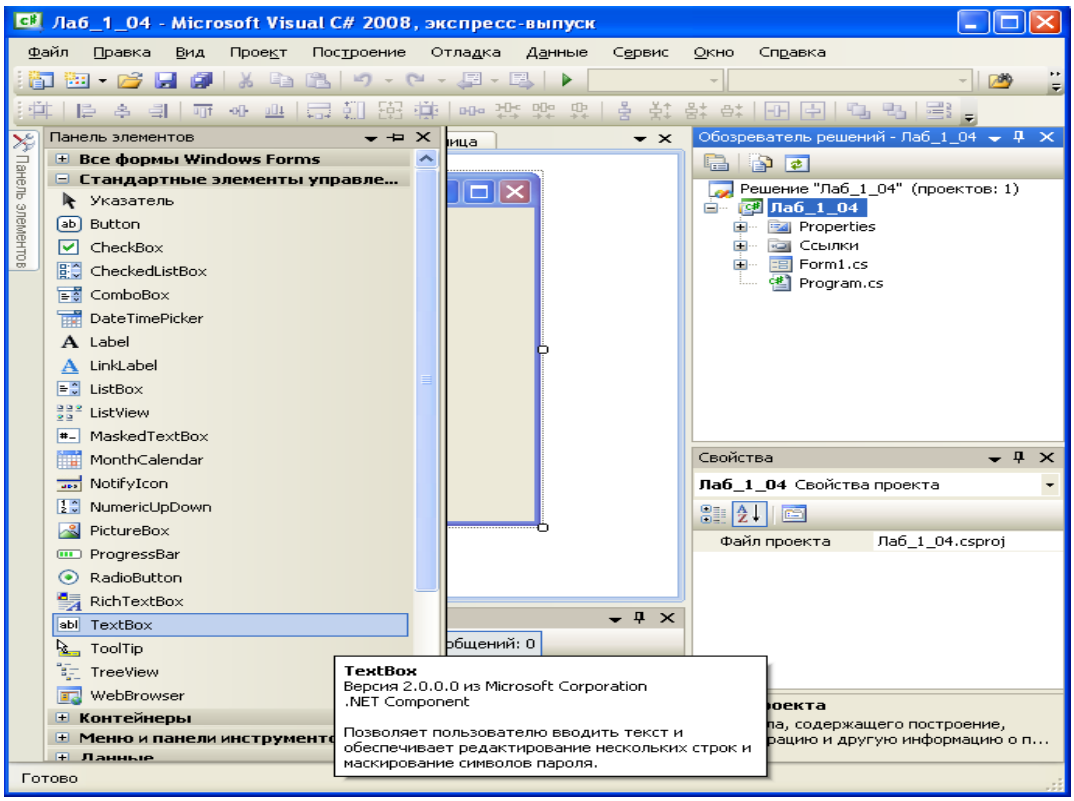
2. Додати до проекту функції збереження даних у файл та зчитування їх з файлу.

Послідовність вирішення завдання:

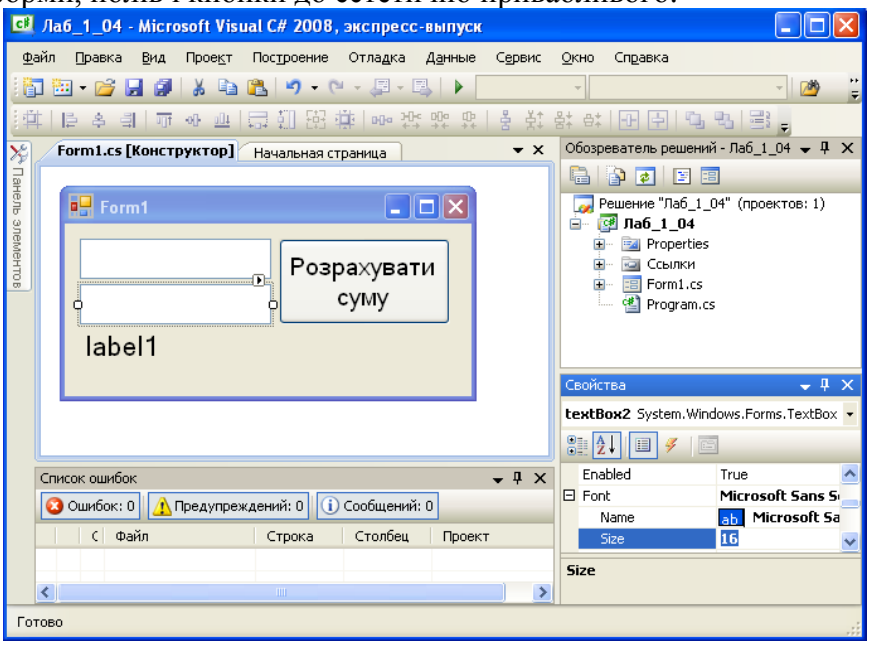
1. Розпочинаємо новий ***Windows Forms -проект.***



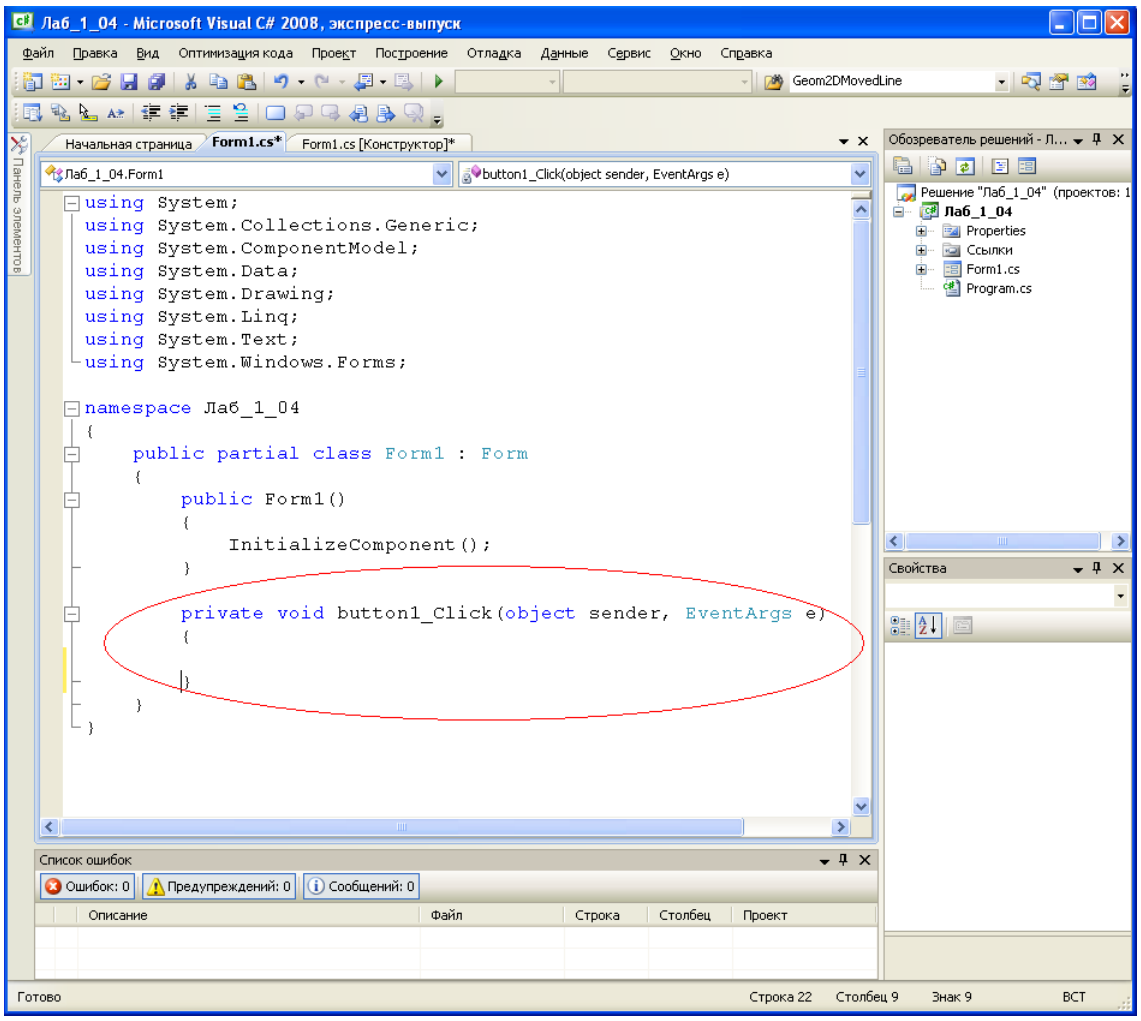
2. Розгортаємо панель елементів:



3. Обираємо і додаємо до форми два текстових поля (textBox1, textBox2), одну мітку (label1) та одну кнопку (button1). Задаємо для їх властивостей „Font.Size” значення 16, змінюємо розмір форми, полів і кнопки до естетично привабливого:



4. Додаємо до проекту функцію-обробник натиснення на кнопку. Для цього у вікні конструктора форми встановлюємо курсор мишки на зображення кнопки button1 і двічі натискаємо ліву кнопку мишки. Розкривається вікно редактору і до часткового опису форми автоматично додається шаблон обробника події натискання кнопки button1:

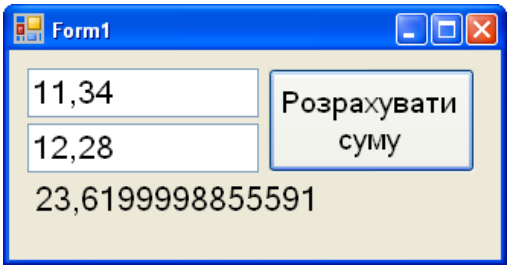


4. Додаємо до функції-обробника оператори розрахунку суми і виведення результату у мітку label1:

```
private void button1_Click(object sender, EventArgs e)
{
    double x1 = float.Parse(textBox1.Text);
    double x2 = float.Parse(textBox2.Text);
    label1.Text = (x1 + x2).ToString();
}
```

Зверніть увагу на додані оператори: введені значення отримуються з властивостей **.Text** відповідних текстових полів і перетворюються у числа методом **.Parse.** Над обчисленою сумою виконується зворотне перетворювання методом **.ToString().**

5. Запускаємо проект на виконання, вводимо контрольні дані і отримуємо результат:

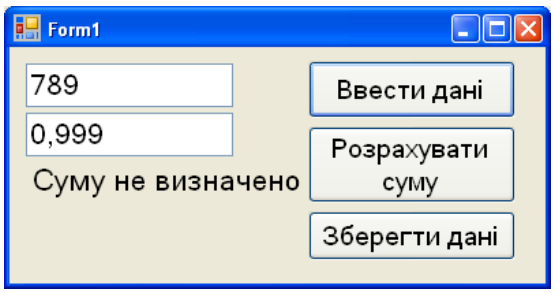


6. Додаємо до форми ще дві кнопки і призначаємо їм такі функції-обробники подій:

```
private void button2_Click(object sender, EventArgs e)
 { // Введення даних з файлу
     string[] s1 = File.ReadAllLines(@"data1.txt");
     textBox1.Text=s1[0];
     textBox2.Text=s1[1];
     label1.Text = "Суму не визначено";
 }

 private void button3_Click(object sender, EventArgs e)
 { // Запис даних у файл
     File.WriteAllText (@"data1.txt",textBox1.Text+"\n");
     File.AppendAllText(@"data1.txt",textBox2.Text+"\n");
      }
```

7. Запускаємо проект на виконання, вводимо контрольні дані і натискаємо на третю кнопку. Потім закриваємо форму і знов запускаємо проект на виконання. Збережені дані будуть зчитані з файлу:



Відмітимо наступне:

- Дані є текстовими і записуються у файл в текстовому вигляді.
- Дані записуються рядок за рядком і завершуються роздільником рядків „**\n**”.
- Перше число записується методом **File.WriteAllText,** щоб знищити попередню версію файлу.
- Друге число записується методом **File.AppendAllText**, щоб додати другий рядок без знищення файлу.
- Створений файл зберігається, за замовченням, у поточній папці. У режимі налагодження це папка \bin\Debug даного проекту.
- Дані зчитуються з файлу всі разом методом **File.ReadAllLines.** При цьому створюється новий масив рядків і посилання на нього присвоюється змінній **string[] s1.**

ДОВІДКОВІ МАТЕРІАЛИ

4.1. Розробка windows-застосувань

Створення Windows-застосувань, або, як їх іноді ще називають, "товстих клієнтів", можливо в Visual Studio з використанням двох ключових компонентів .NET Framework - Windows Forms (з'явився в .NET Framework 1.0) і Windows Presentation Foundation (WPF, з'явився в листопаді 2006 р. в складі .NET Framework 3.0).

Загалом Microsoft Visual Studio - це інтегроване середовище розробки (Integrated Development Environment (IDE)) для створення, документування, запуску і налагодження програм, написаних на мовах .NET. Це потужний інструмент професійної розробки складних застосувань.

Запускаємо Visual Studio.NET, вибираємо File/New/Project (або CreateProject) - з'являється діалогове вікно, у якому вибираємо Visual C# й Windows Form Application

У поле Name задаємо ім'я проекту - Form2 і зберігаємо його в папку, обумовлену полем Location. Папку можна згодом перемістити на інший комп'ютер і продовжити роботу - у ній будуть перебувати всі створювані файли цього проекту. На екрані з'явилася порожня Windows-форма. Після того, як створили новий проект, можна бачити основні частини середовища розробки.

4.1.1. Команди головного меню

Головне вікно Visual Studio.NET, подібно іншим додаткам Windows, містить рядок меню, що включає в себе наступні категорії (коли перебуваємо на Start Page, частина категорій не видима - вона з'явиться пізніше, коли буде створений проект).

У цих категоріях розташовані (або можуть бути) наступні команди:

- File - відкриття, створення, додавання, закривання, друк тощо;
- Edit - стандартні команди виправлення: копіювання, вставка, вирізання тощо;
- View - команди для приховання й відображення всіх вікон і панелей інструментів;
- Project - команди для роботи із проектом: додавання елементів, форм, посилань тощо;

- Build - команди компіляції програми;
- Debug - команди для настроювання програми;
- Data - команди для роботи з даними;
- Format - команди форматування розташовуваних елементів (вирівнювання, інтервал тощо);
- Tools - команди додаткових інструментів і настроювання Visual Studio .NET;
- Test - група команд тестування;
- Window - керування розташуванням вікон;
- Help – довідка

4.1.2. Вікно Solution Explorer

Вікно Solution Explorer (провідник проекту, View/Solution Explorer) містить компоненти, що входять до складу проекту. Пункти контекстного меню цього вікна (викликаються натисканням правою кнопкою миші) дозволяють змінювати вміст проекту, а також додавати нові компоненти

При створенні нового проекту Solution Explorer містить компоненти, створені шаблоном.

Папка **References** містить посилання на класи, використовувані в проекті за замовчуванням. Подвійне натискання миші на підпапках **References** запускає вікно **Object Browser** (провідник об'єктів, **View/Object Browser**). Вікно **Object Browser**, у свою чергу, є вичерпним засобом одержання інформації про властивості об'єктів, як абстрактний клас **brush** успадковується від класу **System.MarshalByRefObject** і містить методи **Clone, Dispose(bool**), **Dispose й Finalize**.

4.1.3. Вікно Object Browser

В ньому можна одержати короткий опис будь-якого методу, класу або властивості, клацнувши на ньому. При цьому на інформаційній панелі негайно відобразиться коротка довідка.

5.1.4. Вікно Class View

Вікно **Class View** - (огляд класів, **View/Class View**), дозволяє переміщатися в коді по обраному об'єкту; містить методи, класи, дані всього лістингу проекту. Для переходу, наприклад, у метод **Main** необхідно клацнути на відповідній назві у вікні **Class View**.

5.1.5. Вікно Properties

Вікно властивостей Properties - основний інструмент настроювання форми та її компонентів. Вміст цього вікна являє собою весь список властивостей обраного в цей момент компонента або форми. Викликається це вікно декількома способами – у меню View слід вибирати пункт Properties Window (або Alt+Enter), на обраному об'єкті клацнути правою кнопкою миші і у контекстному меню вибирати пункт Properties.

Коли проект вже створений, у вікні Properties відображаються властивості форми.

Опис інтерфейсу вікна Properties

Елемент	Опис
Object name	У поле цього списку виводиться назва даного обраного об'єкта, який є екземпляром якого-небудь класу. Тут Form1 - назва форми за замовчуванням, що успадковується від класу System .Windows. Forms. Form
Categorized	При натисканні на цю кнопку проводиться сортування властивостей обраного об'єкта за категоріями. Можна закривати категорію, зменшуючи число видимих елементів. Коли категорія схована, ви бачите знак (+), коли розкрита – (-)
Alphabetic	Сортування властивостей і подій об'єкта за абеткою
Properties	При натисканні на цю кнопку відображається перерахування властивостей об'єкта
Events	При натисканні на цю кнопку відображається перерахування подій об'єкта
Description Panel	Панель, на яку виводиться інформація про обрану властивість

Вікно Properties дозволяє визначати дизайн форми та її елементів керування. У наступній таблиці наведено опис деяких властивостей форми, звичайно обумовлених у режимі дизайну. При виборі значення властивості, відмінного від прийнятого за замовчуванням, воно виділяється жирним шрифтом, що спрощує надалі визначення змін.

Деякі властивості форми

Властивість	Опис	Значення за замовчуванням
Name	Назва форми в проекті. Це не заголовок форми, що видний при запуску форми, а назва форми усередині проекту, який буде використовуватися в коді	Forml, Form2 і т.д.
AcceptButton	Встановлюється значення кнопки, що буде спрацьовувати при натисненні клавіші Enter. Для того, щоб ця властивість була активною, необхідна наявність принаймні однієї кнопки, розташованої на формі	None
BackColor	Кольори форми. Для швидкого перегляду різних варіантів просто натисніть прямо на назві BackColor	Control
Background Image	Зображення на задньому тлі	None
CancelButton	Встановлюється значення кнопки, що буде спрацьовувати при натисненні клавіші Esc. Для того, щоб ця властивість була активною, необхідна наявність принаймні однієї кнопки, розташованої на формі	None
ControlBox	Встановлюється наявність або відсутність трьох стандартних кнопок у верхньому правому куті форми: Згорнути, Розгорнути й Закрити	
Cursor	Визначається вид курсору при його положенні на формі	Default
DrawGrid	Встановлюється наявність або відсутність сітки із точок, що допомагає форматувати елементи керування. У кожному разі сітка видна тільки на стадії створення додатка	True
Font	Форматування шрифту, що використовується для відображення тексту на формі в елементах керування	Microsoft Sans Serif; 7,8pt

Властивість	Опис	Значення за замовчуванням
Icon	Зображення іконки, яка розташована в заготовку форми. Підтримуються формати .ico	
Maximize Box	Визначається активність стандартної кнопки Розгорнути у верхньому правому куті форми	True
Maximum Size	Максимальний розмір ширини і висоти [юрми, що задається в пікселях. Форма буде приймати зазначений розмір при натисканні на стандартну кнопку Розгорнути	0;0 (На весь екран)
MinimízeBox	Зазначається активність стандартної кнопки 1 »горнути у верхньому правому куті форми	Crue
MinimumSize	Мінімальний розмір ширини і висоти форми, що задається в пікселях. Форма буде приймати зазначений розмір при зміні її меж користувачем (якщо властивість FormBorderStyle має значення за замовчуванням Sizable)	0;0
Size	Ширина і висота форми	300; 300
FormBorder Style	Визначення виду границь форми. Можливі варіанти: - None - форма без границь і рядка заголовка; - FixedSingle - тонкі границі без можливості зміни розміру користувачем; - Fixed3D - границі без можливості зміни розміру з тривимірним ефектом; - FixedDialog - границі без можливості зміни, без іконки додатка; - Sizable - звичайні границі: користувач може змінювати розмір границь; - FixedToolWindow - фіксовані границі, є тільки кнопка закриття форми. Такий вид мають панелі інструментів у додатках; - SizableToolWindow - границі з можливістю зміни розмірів, є тільки кнопка закриття форми	Sizable
Startposition	Визначення розташування форми при запуску додатка. Можливі наступні значення: - Manual - форма з'являється у верхньому лівому куті екрана; - CenterScreen - у центрі екрана; - WindowsDefaultLocation - розташування форми за замовчуванням. Якщо користувач змінив розміри форми, то при наступному її запуску вона буде мати той самий вид і розташування; -WindowsDefaultBounds - межі формі приймають фіксований розмір; - CenterParent - у центрі батьківської форми	Windows Default Location
Text	Заголовок форми. На відміну від властивості Name, це саме назва форми, яка ні використовується в коді	Forml, Form 2 і т.д.
Windows State	Визначення положення форми при запуску. Можливі наступні значення: - Normal - форма запускається з розмірами, зазначеними у властивості Size; - Minimized - форма запускається з мінімальними розмірами, зазначеними у властивості MinimumSize; - Maximized - форма розвертається на весь екран	Normal

Кнопка вікна властивостей **Events** (Події) перемикає вікно **Properties** у режим керування оброблювачами різних подій (наприклад, миші, клавіатури) і одночасно виводить список всіх подій компонента. Подвійне натискання миші в полі значення події генерує оброблювач для нього і перемикає в режим коду.

5.7.6. Вікно Toolbox

Вікно **Toolbox** (панель інструментів, **View/Toolbox**, або сполучення клавіш **Ctrl+Alt+X**) містить компоненти Windows-форм, що називаються також елементами керування, які розміщаються на формі. Воно складається з декількох закладок: **Common Controls, Containers, All Windows Forms** й **General**.

Найбільш часто вживаною закладкою є **All Windows Forms**. Для розміщення потрібного елемента керування досить просто клацнути на ньому у вікні **Toolbox** або, схопивши, перетягнути його на форму.

Для відновлення значень за замовчуванням необхідно вибрати в контекстному меню будь-якої закладки пункт **Reset Toolbox**.

5.1.7. Режими дизайну та коду

При створенні нового проекту запускається режим дизайну – форма являє собою основу для розміщення елементів керування. Для роботи із програмою варто перейти в режим коду. Це можна зробити декількома способами: клацнути правою кнопкою миші в будь-якій частині форми і у меню, що з'явилося, вибрати **View Code**. Після хоча б однократного переходу в режим коду в цьому проекті з'явиться вкладка **Form1.cs*,** натискаючи на яку теж можна переходити в режим коду. Для переходу в режим коду також можна використати клавішу **F7**, а для повернення в режим дизайну – сполучення **Shift+F7**. Перейдемо в режим коду та розглянемо деякі блоки.

Даний блок визначає, які простори імен використовуються в цьому проекті:

```
using System;
using System.Collections.Generic:
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
```

Для перегляду інформації про вміст кожного із цих просторів можна скористатися вікном **Object Browser**.

Далі визначається власний простір імен, ім'я якого збігається з назвою проекту:

```
namespace form2
```

При необхідності цю назву можна міняти. Клас форми **Form**., наслідується від **System.Windows.Forms**. Всередині цього класу перебуває **конструктор форми**:

```
public Form1()
{
InitializeComponentQ;
}
```

Подія **Initiliaze** відбувається в момент запуску додатка; код, що слідує після **InitlalizeComponent**, може змінювати вміст форми або елементи керування в момент запуску. Клас форми **Forml,** містить у собі майже весь код:

```
partial class Forml
 {
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.iContainer components = null;
/// <summary>
/// Clean up any resources being used. /// </summary>
```

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

Область **Windows Form Designer generated code** містить код графічного інтерфейсу елементів керування і форми, який автоматично генерується середовищем. Порожня форма містить опис розмірів і заголовка. Клацніть на знак (+) для перегляду цієї області:

```
#region Windows Form Designer generated code /// <summary>
/// Required method for Designer support - do not modify /// the contents of this method with the code editor. /// </summary>
private void lnltializeComponent()
 {
this.SuspendLayoutQ;
//
// Forml
//
this.AutoScaleDimensions = new System.Drawing.Size(8F, 16F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(282, 257);
this.Name = "Forml"; thls.Text = "Forml"; thls.ResumeLayout(false);
}
#endregion
```

Можна міняти значення параметрів, створювані середовищем, і тоді зміни негайно відобразяться на графічному інтерфейсі.

Метод **Main** реалізує головну точка входу в програму - місце, звідки починається виконання написаного коду:

```
static void Main()
{
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(new Form 1());
}
```

При налагодженні великих програм зручно використовувати нумерацію рядків, яку можна включити в пункті меню **Tools/Options.../Text Editor/C#** - на формі **Display** - галочка **Line Numbers**. Крім цього, пункт меню **Tools/Options...** можна використовувати для установки інших параметрів (шрифт коду, кольори тла й т.д.).

Розглянемо властивості проекту. У вікні **Solution Explorer** виділяємо назву проекту - **form2**, клацаємо правою кнопкою миші і вибираємо в меню пункт **Properties**. У вікні, що з'явилося, містяться усі властивості поточного проекту

У вікні властивостей міститься досить багато параметрів. Розглянемо найбільш вживані. Вкладка **Properties/Application** містить у собі наступні властивості:

- **Assembly Name** - назва зборки;
- **Output Type** - тип компільованого додатка. За замовчуванням для Windows-застосувань встановлено **Windows Application**;
- **Default Namespace** - назва простору імен у коді. За замовчуванням збігається з ім'ям проекту;
- **Startup Object** - назва класу, що містить точку входу в програму - метод Main;
- **Application Icon** - шлях до файла з іконкою додатка;
- **Project File** - ім'я файла з інформацією про проект. Перебуває у папці із проектом;
- **Project Folder** - шлях до файла із проектом;

- **Output File** - назва файла, створюваного при компіляції, - вихідного файла. Збігається з ім'ям проекту.

На вкладці **Properties/Build** розглянемо деякі властивості:

- **Optimize Cod**e - оптимізація програми, значення цієї властивості **true** може значно збільшити продуктивність додатка;

- **Allow Unsafe Code Blocks** - дозволити використання ключового слова **unsafe** у коді проекту;

-**Warning Level** - рівень попереджень, відображуваних при компіляції програми;

- **Treat Warnings As Errors** - сприймати всі попередження як помилки. Якщо оголосити змінну в коді, але ніде не використати її, при значенні цієї властивості **False** застосування компілюється, при значенні **True** - ні;

- **Output Path** - шлях, де формується вихідний файл. Папка **bin** перебуває всередині папки проекту;

- **Generate Debugging Information** - виводити інформацію при налагодженні. Ця властивість повинна бути включеною: саме ці повідомлення допомагають виправляти код.

5.1.9. Компіляція програми

Для перевірки програми використовуються два способи налагодження. Перший спосіб - **Build Solution**, що перевіряє код і компілює програму, не запускаючи її. Це зручно, коли робота йде над окремим модулем розробки, і відсутня можливість перевіряти весь продукт в цілому. Для запуску цього способу вибираємо в головному меню пункт **Build/Build Solution**.

При цьому з'являється вікно **Error List**, у якому виводиться інформація про всі стадії компіляції. Якщо в коді є помилки, вікно виведе повідомлення про помилки, потім з'явиться їх список, причому для налагодження досить двічі клацнути на відповідній помилці для переходу до потрібної ділянки коду.

Інший спосіб компіляції програми – виконання режиму **Debug**, при якому перевіряється код, компілюється програма і формується інтерфейс користувача.

Для запуску цього способу натискаємо клавішу **F5**. На екрані знову з'являється вікно, що інформує про хід компіляції. Якщо застосування не містить помилок, то на екрані з'явиться готова форма.

При запуску додатка в папці **bin\Debug** проекту створюється файл **Form2.exe** і файли, необхідні для настроювання. Файл Form2.exe являє собою готовий застосування. Готовий застосування для поширення необхідно скомпілювати в режимі Release – тоді з'явиться папка **bin\Release**.

У меню **Debug** також розташовані всі засоби для покрокового налагодження коду.

5.2. Перший Windows-проект на C#

Побудуємо Windows-проект, повторюючи вже описані дії, у вікні нового проекту. Оберіть тип проекту Windows Form Application, і дайте йому ім'я WindowsHello.

За замовчуванням будується рішення, що містить єдиний проект з єдиним простором імен (всі три конструкції мають імена, що збігаються). У простір імен вкладений єдиний клас Forml.

Процедура **Main** містить рядок:

```
Application. Run(new Form1());
```

Клас Application викликає статичний метод Run, якому як фактичний аргумент передається об'єктний вираз new Forml(). При обчисленні цього виразу створюється перший об'єкт - екземпляр класу Forml. Цей об'єкт стає поточним. Для створення об'єкта викликається

конструктор класу. У процесі роботи конструктора здійснюється виклик методу InitializeComponent() (файл Forml Designer.cs). Метою цього виклику є поточний об'єкт - уже створений об'єкт класу Forml. Ні в конструкторі, ні у викликаному методі нові об'єкти не створюються. По завершенні роботи конструктора об'єкт класу Forml передається методу Run як аргумент.

Метод Run класу Application - відкриває форму - видимий образ об'єкта класу Forml, з яким тепер може працювати користувач. Але головна його робота полягає в тому, що він створює Windows-застосування, запускаючи цикл обробки повідомлень про події, що відбуваються. Повідомлення, що надходять, опрацьовуються операційною системою відповідно до черги і пріоритетів, викликаючи оброблювачі відповідних подій. Оскільки наша форма за замовчуванням не містить ніяких елементів керування, то вхідних повідомлень небагато. Все, що може робити користувач із формою, це переміщувати її по екрану, згортати та змінювати розміри. Звичайно, можна ще закрити форму. Це приведе до завершення циклу обробки повідомлень, до завершення роботи методу Run, методу Main, додатка.

Розширимо застосування, додавши віконця для введення і виведення інформації. Як уже говорилося, при створенні Windows-додатка за замовчуванням створюється не тільки об'єкт класу Forml - нащадок класу Form, але і його видимий образ - форма, з якою можна працювати в режимі проектування, додаючи на неї елементи керування. Додамо у форму наступні елементи керування (з Toolbox (All Windows Form)):

- текстове вікно і мітку. За замовчуванням вони одержать імена textBoxl та label1. Текстове вікно призначається для введення імені користувача; мітка, візуально пов'язана з вікном, дозволить вказати призначення текстового вікна. Установимо властивість Multiline для текстового вікна як true, властивість Text у мітки – Ваше имя;

- аналогічна пара елементів керування - textBox2 та label2 – призначені для виводу привітання. Оскільки вікно textBox2 призначено для виводу, то необхідно включити його властивість Readonly;

- додамо на форму командну кнопку (з Toolbox (Common Controls)), оброблювач події Click, який і буде організовувати читання імені користувача з вікна textBoxl і вивід привітання у вікно textBox2.

От як виглядає автоматично генерований і доданий в клас опис елементів керування:

```
private System.Windows.Forms.TextBox textBoxl;
private System.Wlndows.Forms.Label labeH;
private System.Windows.Forms.Label Iabel2;
private System.Windows.Forms.TextBox textBox2;
private System.Windows.Forms.Button button1;
```

Текст методу InitailizeComponent:

```
private void InitializeComponent()
{
this.textBox1 = new System.Windows.Forms.TextBox();
this.label1 B new System.Wlndows.Forms.Label();
this.label2 = new System.Windows.Forms.Label();
this.textBox2 = new System.Windows.Forms.TextBox();
this.button1 = new System.Windows.Forms.Button();
this.SuspendLayout();
// textBox1
this.textBox1.Location = new System.Drawing.Point(131, 29);
this.textBox1.Multiline = true; this.textBox1.Name = "textBox1";
this.textBox1.Size = new System.Drawing.Size(261, 22);
this.textBox1.TabIndex = 0;
// label1
this.label1 .AutoSize = true;
this.label1.Location = new System.Drawing.Point(12, 34);
```

```
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(73,17);
this.label1.TabIndex=1;
this.label1.Text = "Ваше имя";
// label2
this.label2. AutoSize = true;
this.label2.Location = new System.Drawing.Polnt(12,159);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(93,17);
thls.label2.Tablndex = 2;
this.label2.Text = "Приветствие";
//  textBox2
this.textBox2.Location * new System.Drawlng.Polnt(131,156);
this.textBox2.Name = "textBox2"; this.textBox2.Readonly = true;
this.textBox2.Size = new System.Drawing Size(261, 22);
this.textBox2.Tablndex = 3;
//  buttonl
this.buttonl.Location = new System.Drawing.Polnt(131, 81);
this.buttonl.Name = "buttonl";
this.buttonl .Size = new System.Drawing.Slze(134, 33);
this.buttonl.Tablndex = 4;
this.buttonl .Text = "Привет";
this.buttonl .UseVisualStyleBackColor = true;
// Form1
this.AutoScaleDimensions = new System.Drawing.Size(8F, 16F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMocle.Font;
this.ClientSize = new System. Drawing.Size(404, 211);
this.Controls.Add(this.buttonl);
this.Controls.Add(this.textBox2);
this.Controls.Add(this.label2);
this.Controls. Add(this.label1);
this.Controls.Add(this.textBoxl);
this.Name = "Forml";
this.Text = "Приветствие";
this.Load += new System.EventHandler(this.Form1_Load);
this.ResumoLayout(false);
this.PerformLayout();
}
#endregion
```

Додамо оброблювач подій командної кнопки. Для цього досить виділити потрібний елемент у формі, у вікні властивостей натиснути кнопку подій (зі значком блискавки), зі списку подій вибрати потрібну подію та клацнути на ній.

У даній ситуації все можна зробити простіше - подвійне натискання на кнопці включає подію і автоматично будується метод оброблювача події з потрібним ім'ям і параметрами. Так він виглядає:

```
private void button1_Click(object sender, EventArgs e)
{
}
```

Додамо в нього наступні рядки:

```
private void button1_Click(object sender, EventArgs e)
{
string temp; //Описали змінну
```

```
temp = textBox1.Text;       // Присвоїли їй значення з textBoxl
// Проаналізували її зміст
if (temp="")
textBox2.Text = "Здравствуй,:-)";
else
// Вивели значення temp в textBox2
textBox2.Text = "Здравствуй, " + temp + " !";
}
```