Основи програмної інженерії

- Лекції, лабораторні роботи – к.т.н., доц. Каплієнко Тетяна Ігорівна
- Лабораторні роботи – ас. Качан Олександр Іванович
- Форма контролю - **залік**

Microsoft Visual Studio

Microsoft Visual Studio – набір продуктів компанії Майкрософт, які включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні застосунки, так і застосунки із **графічним інтерфейсом**, а також веб-сайти, програми для роботи із офісними застосунками та іншими видами програмного забезпечення.

Visual Studio дозволяє розробляти ПЗ на наступних мовах:

- Visual Basic
- Visual C++
- Visual C#
- Visual F#

Мова програмування C#

Основні поняття мови

Visual C#

C# (вимовляється як сі шарп) – об`єктно-орієнтована мова прогрмування, призначена для розробки різноманітних застосунків, виконуваних у середовищі **.NET Framework**. На мові C# можна розробляти звичайні клієнтські застосунки Windows, консольні програми, веб-сайти, застосунки виду «сервер-клієнт», застосунки для роботи з базами даних і багато інших.

Склад мови

- **Символи:**
  - букви: A-Z, a-z, _, букви нац. алфавітів
  - цифри: 0-9, A-F
  - спец. символи: +
  - пробільні символи, *, ,, …

■**Лексеми:**

- ■константи: 2 0.11 “Вася”
- ■імена: Vasia a
- ■ключові слова: _11 double
- ■знаки операцій: do if =
- ■роздільники: ; [ ] ,

■**Вирази**

- ■вираз – правило обчислення значення: a + b

■**Оператори**

- ■виконувані: c = a + b;
- ■опис: double a, b;

Константи
си

Вид	Приклади
Булеві	true false
Цілі десяткові шістнадцяткові	 8 19922 0xA 0x1B8 0X00FF
Дійсні з точкою з порядком	 5.7 .001f 0.2E6 .11e–3 5E10
Символьні	'A' '\x74' '\0' '\uA81B'
Рядкові	"Тут був Vasia“ "Тут був \u0056\u0061" @"C:\temp\file1.txt"
Константа null	null

Імена (ідентифікатори)

- ім’я повинно починатися з букви або _;
- ім’я повинно містити тільки букви, знаки підкреслення і цифри;
- **прописні та малі букви відрізняються**;
- довжина імені практично не обмежена;
- імена не повинні співпадати із ключовими словами, однак допускається: @if, @float…

Приклади правильних імен: Vasia, Вася, _13, @while.

Приклади неправильних імен: 2late, Big aaa, Б#г

Нотації (рекомендації по найменуванню змінних)

Зрозумілі та узгодженні між собою імена – основа хорошого стилю. Існує кілька нотацій – узгоджень про правила створення імен.
У C# для найменування різних видів програмних об`єктів найчастіше використовують дві нотації:

- *Нотація Паскаля* – кожне слово починається із прописної букви:
  - MaxLength, MyFuzzyShooshpanchik
- *Camel notation* – кожне слово починається із прописної букви, крім першої:
  - maxLength, myFuzzyShooshpanchik

Ключові слова, знаки операцій, роздільники

- *Ключові слова* — ідентифікатори, які мають спеціальне призначення для компілятора. Їх можна використовувати тільки у тому сенсі, у якому вони визначені.
  - Наприклад, для оператора переходу визначено слово goto.
- *Знак операції* — один або більше символів, які визначають дію над операндами. Всередині знаку операції пробіли не допускаються.
  - Наприклад, додавання +, ділення/, складне присвоєння +=.
- Операції поділяються на унарні (з одним операндом), *бінарні* (з двома) и *тернарні* (з трьома).
- *Роздільники* використовуються для розділення або, навпаки, групування елементів. Приклади роздільників: дужки, крапка, кома.

Ключові слова C#

abstract as base bool break byte case catch
char checked class const continue decimal
default delegate do double else enum event
explicit extern finally fixed float for
foreach false goto implicit in int
interface if is lock long namespace
new null internal out override params
private object operator ref return
protected public readonly
sbyte sealed sizeof stackalloc static string
switch short throw true try typeof struct ulong
this unsafe ushort uint void
unchecked using while virtual
volatile

Типи даних

Концепція типу даних

Тип даних визначає:

- внутрішнє представлення даних => множина можливих значень, які може приймати змінна

- допустимі дії над даними => операції і функції

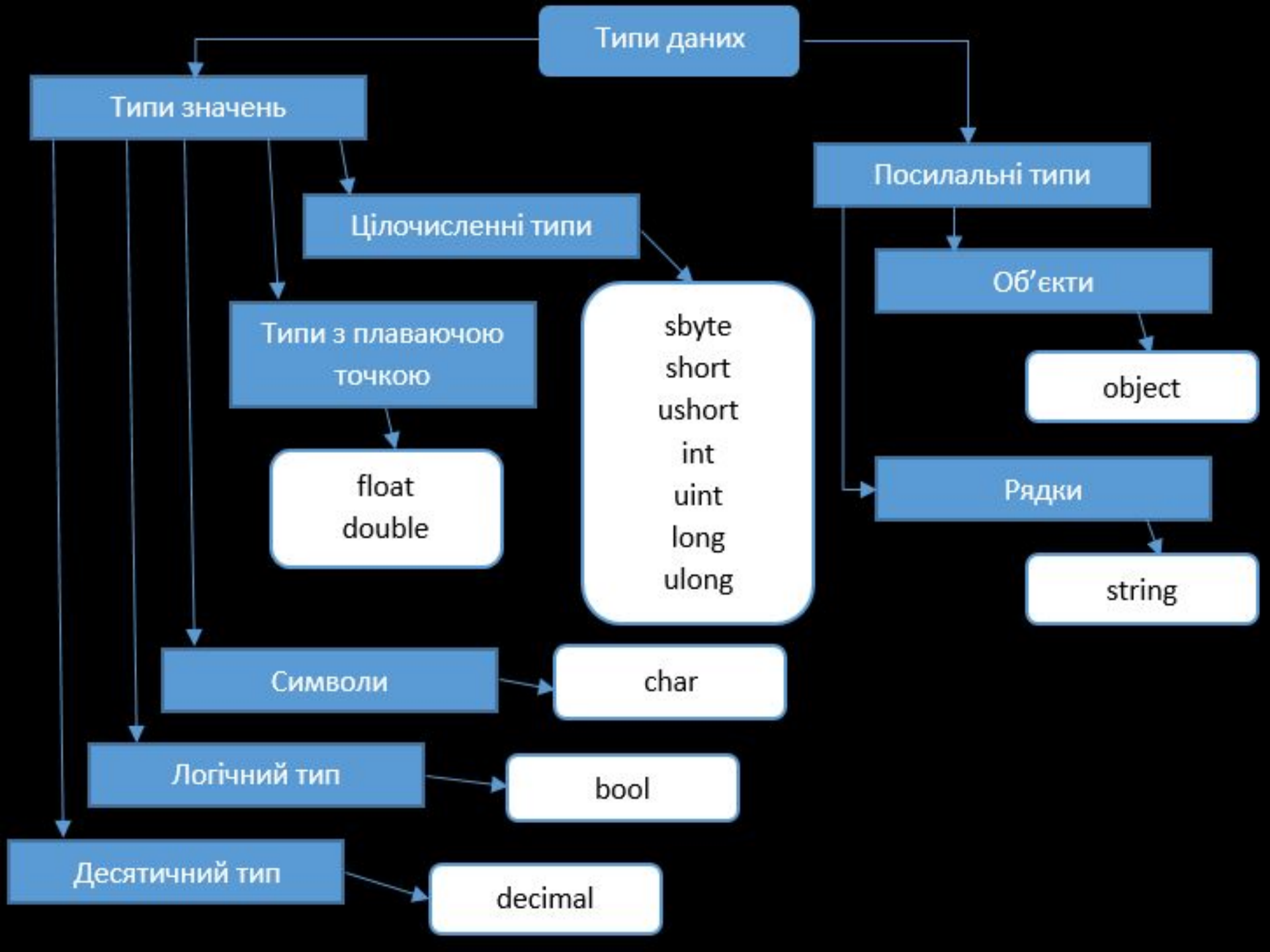
*Типи даних мають особливе значення в C#, так як це **суворо класифікована мова**.*
*Це значить, що всі операції піддаються суворому контролю зі сторони компілятора на **відповідність типів**, причому недопустимі операції не компілюються.*
Отже, суворий контроль типів дозволяє виключати можливі помилки та підвищити надійність програм. Для забезпечення контролю типів всі змінні, вирази та значення повинні належати до певного типу.

Типи значень і посилальні типи

В C# наявні дві загальні категорії вбудованих типів даних:

- типи значень(value type);
- посилальні типи (reference type).

Вони відрізняються за вмістом змінної. Концептуально різниця між ними полягає у тому, що перший тип зберігає дані безпосередньо, в той же час як посилальний тип зберігає посилання на значення.


Типи даних
Типи значень
Посилальні типи
Цілочисленні типи
Об’єкти
Типи з плаваючою точкою
sbyte
short
ushort
int
uint
long
ulong
object
float
double
Рядки
string
Символи
char
Логічний тип
bool
Десятичний тип
decimal

Логічний і цілі типи даних

Назва	Ключове слово	Тип .NET	Діапазон значень	Опис	Розмір, біт
Логічний	bool	Boolean	true, false		
Цілі	sbyte	SByte	–128 — +127	знаковий	8
	byte	Byte	0 — +255	беззнаковий	8
	short	Int16	–32768 — +32767	знаковий	16
	ushort	UInt16	0 — +65535	беззнаковий	16
	int	Int32	$\approx(-2*10^9 — +2*10^9)$	знаковий	32
	uint	UInt32	$\approx(0 — +4*10^9)$	беззнаковий	32
	long	Int64	$\approx(-9*10^{18} —+ 9*10^{18})$	знаковий	64
	ulong	UInt64	$\approx(0— 18*10^{18})$	беззнаковий	64

Інші

Символьний	char	Char	U+0000 — U+ffff	символ Unicode	16
Дійсний	float	Single	$-3.4*10^{33}$ — $+3.4*10^{33}$	точність – 7 розрядів	32
	double	Double	$-1.7*10^{306}$ — $+1.7*10^{306}$	точність – 16 розрядів	64
Десятковий	decimal	Decimal	$-7.79*10^{28}$ — $+7.9*10^{28}$	точність – 28 розрядів	128
Рядковий	string	String	Довжина обмежена об`ємом доступної пам`яті	Рядок із символів Unicode	
object	object	Object	Об`єктний тип	Загальний предок	

Лінійні програми

Поля (властивості) і методи вбудованих типів

- Кожний вбудований тип C# побудований на базі стандартного класу бібліотеки .NET. Це значить, що у вбудованих типів даних C# є методи і поля. За допомогою їх можна, наприклад, отримати:
  - double.MaxValue (або System.Double.MaxValue) — максимальне число типу double;
  - uint.MinValue (або System.UInt32.MinValue) — мінімальне число типу uint.
- В дійсних класах наявні елементи:
  - Додатна (плюс) нескінченність PositiveInfinity – значенням цієї константи є результат ділення додатного числа на нуль. Ця константа повертається у випадку, якщо результат операції більше, ніж MaxValue;
  - від`ємна (мінус) нескінченність NegativeInfinity;
  - «не є числом»: NaN – константа, яка повертається в результаті роботи функції у випадку, якщо не вказано результат операції. Наприклад, результат при діленні на нуль.

Змінні

- *Змінна* — це величина, яка під час роботи програми може змінювати своє значення.
- Всі змінні, які використовуються у програмі, повинні бути оголошені.
- Для кожної змінної задається її ім’я та тип:

```
int     number;
float   x, y;
char    option;
```

- Тип змінної обирається на базі діапазону і потрібної точності представлення даних.

Область дії і час життя змінних

- Змінні оголошуються всередині будь-якого блоку (класу, методу або блоку всередині методу)
  - Блок — це код, укладений в фігурні дужки. Основне призначення блоку — групування операторів.
  - Змінні, оголошені безпосередньо всередині класу, називаються полями класу.
  - Змінні, оголошені всередині методу класу, називаються локальними змінними.
- Область дії змінної – область програми, де можна використовувати змінну.
- Область дії змінної починається з місця її оголошення і продовжується до кінця блоку, всередині якого вона оголошена.
- Час життя: змінні створюються при вході до їхньої області дії (блоку) і знищуються при виході.

Оголошення змінних

- При оголошенні можна присвоювати змінній початкове значення (ініціювати).

```
int     number = 100;
float   x      = 0.02;
char    option = ’я’;
```

- При ініціюванні можна використовувати не тільки константи, а й вирази – головне, щоб на момент оголошення вони були обчислюваними, наприклад:

```
int b = 1, a = 100;

int x = b * a + 25;
```

- *Якщо змінними при їх створенні явно не присвоювати будь-які значення, то вони ініціюються «значенням по замовчуванню» (0, null).*

Іменовані константи

Замість значень констант можна (і потрібно!) використовувати у програмі їх імена.

Це полегшує читабельність програми і внесення в неї змін:

```
const    float          weight
const    = 61.5;  int   n
const             = 10;
         float  g   = 9.8;
```

Вирази

- *Вирази* — правило обчислення значення.
- У виразі беруть участь *операнди*, об`єднані знаками операцій.
- Операндами виразів можуть бути константи, змінні і виклики функцій.
- Операції виконуються у відповідності із *пріоритетами*.
- Для зміни порядку виконання операцій використовують *круглі дужки*.
- Результатом виразу завжди є значення певного типу, який визначається типами операндів.
- Величини, які беруть участь у виразі, повинні бути *сумісними типами*.

- t + Math.Sin(x)/2 * x

результатом є дійсний тип

- a <= b + 2

результатом є логічний тип

- x > 0 && y < 0

результатом є логічний тип

Пріоритети операцій C#

1.	Первинний	(), [], new, …
2.	Унарний	~, !, ++, --, -, …
3.	Типу множення (мультиплікативні)	*, /, %
4.	Типу складання (адитивні)	+, -
5.	Зсуву	<<, >>
6.	Відношення і перевірки типу	<, >, is, …
7.	Перевірка на рівність	==, !=
8.	Порозрядні логічні	&, ^, |
9.	Умовні логічні	&&, ||
10.	Умовна	?:
11.	Присвоєння	=, *=, /=,…

Тип результату виразу

- Якщо операнди, що входять у вираз, належать типу, і операція для цього типу визначена, то результат виразу буде мати той же тип.
- Якщо операнди різного типу і (або) операція для цього типу не визначена, перед обчисленням виконується автоматичне перетворення типів за правилами, які забезпечують отримання більш коротких типів до більш довгих для збереження значимості і точності.
- Автоматичне (*неявне*) перетворення можливе не завжди, а тільки якщо при цьому не може статися втрата значимості.
- Якщо неявне перетворення із одного типу в інший не існує, програміст може задати явне перетворення типу, наприклад, за допомогою операції (тип)x.

Інкремент і декремент

```
private void button1_Click(object sender, EventArgs e)
    {
        int x = 3, y = 3;
        textBox1.Text = "Значення префіксного виразу: ";
        textBox1.Text += (++x) + "\r\n";
        textBox1.Text += "Значення x після перетворення:
        ";
        textBox1.Text += x   + "\r\n";

        textBox1.Text += "Значення постфіксного виразу: ";
        textBox1.Text += (y++) + "\r\n";
        textBox1.Text += "Значення y після перетворення: ";
        textBox1.Text += y   + "\r\n";
    }
```

Результат роботи програми:

```
Значення префіксного виразу: 4
Значення x після перетворення: 4
Значення постфіксного виразу: 3
Значення y після перетворення: 4
```

Операції заперечення

```
using System;
namespace ConsoleApplication1
{     class Class1
   {  static void Main()
      {
         sbyte a = 3, b = -63, c = 126;
         bool  d = true;
         Console.WriteLine( -a );  // Результат -3
         Console.WriteLine( -c );  // Результат -126
         Console.WriteLine( !d );  // Результат false
         Console.WriteLine( ~a );  // Результат -4 = -3-1
         Console.WriteLine( ~b );  // Результат 62 =
         Console.WriteLine( ~c );  -(-63)-1
}}}                                // Результат -127 =
                                   -126-1
```

Множення

- *Операція множення* (*) повертає результат перемноження двох операндів.
- Стандартна операція множення реалізована для типів int, uint, long, ulong, float, double і decimal.
- До величин інших типів її можна використовувати, якщо для них можливе неявне перетворення для відповідних типів. Тип результату операції дорівнює «найбільшому» із типів операндів, але не менше int.
- Якщо обидва операнди цілочисельны або типу decimal і результат операції занадто великий для представлення за допомогою вказаного типу, генерується виключення System.OverflowException

Прикла
д

```
using System;
namespace ConsoleApplication1
{     class Class1
   {     static void Main()
      {
         int x = 11, y = 4;
         float z = 4;
         Console.WriteLine( z * y );          // Результат 16
         Console.WriteLine( z * 1e308 );     // Рез.
         Console.WriteLine( x / y );          "нескінченність"
         Console.WriteLine( x / z );          // Результат 2
         Console.WriteLine( x % y );          // Результат 2,75
         Console.WriteLine( 1e-324 / 1e-324 ); // Результат
}}}                                                  NaN
```

Операції відношення і перевірки на рівність

- *Операції відношення* (<, <=, >, >=, ==, !=) порівнюють перший операнд із другим.
- Операнди повинні бути арифметичного типу.
- Результат операції — логічного типу, дорівнює true або false.

x == y -- true, якщо x дорівнює y, інакше false

x != y -- true, якщо x не дорівнює y, інакше false

x < y -- true, якщо x менше y, інакше

false x > y -- true, якщо x більше y,

інакше false

x <= y -- true, якщо x менше або дорівнює y, інакше false

Операції присвоєння

Присвоєння – це заміна старого значення змінної на нове. Старе значення безслідно стирається.

Операція може використовуватися у програмі як закінчений оператор.

змінна = вираз;

```
a = b + c;
x = 1;
x = x + 0.5;
```

Правий операнд операції присвоєння повинен мати неявне перетворення до типу лівого операнду, наприклад:

Дійсна змінна = цілий вираз;

Введення-виведення в C#

Виведення на консоль

```
using System;
namespace A
{
  class Class1
  {
    static void Main()
    {
      int    i = 3;  double y = 4.12;  decimal d = 600;  string s = "Вася";

      Console.Write( i );
      Console.Write( " y = {0} \n d = {1}", y, d );
      Console.WriteLine( " s = " + s );
    }
  }
}
```

Результат роботи програми: 3 y = 4,12
d = 600 s = Вася

Введення з консолі

```
using System;
namespace A
{       class
Class1  static void Main()
        {
          string s = Console.ReadLine();        // введення рядку

          char c = (char)Console.Read();        // введення
          Console.ReadLine();                   символу

          string buf;                           // буфер для введення
          buf = Console.ReadLine();             чисел
          int i = Convert.ToInt32( buf );       // перетворення в ціле

          buf = Console.ReadLine();
          double x = Convert.ToDouble( buf ); // перетворення в дійсне

          buf = Console.ReadLine();
          double y = double.Parse( buf );       // перетворення в дійсне
        }
    }
}
```

Створення Windows-застосунку

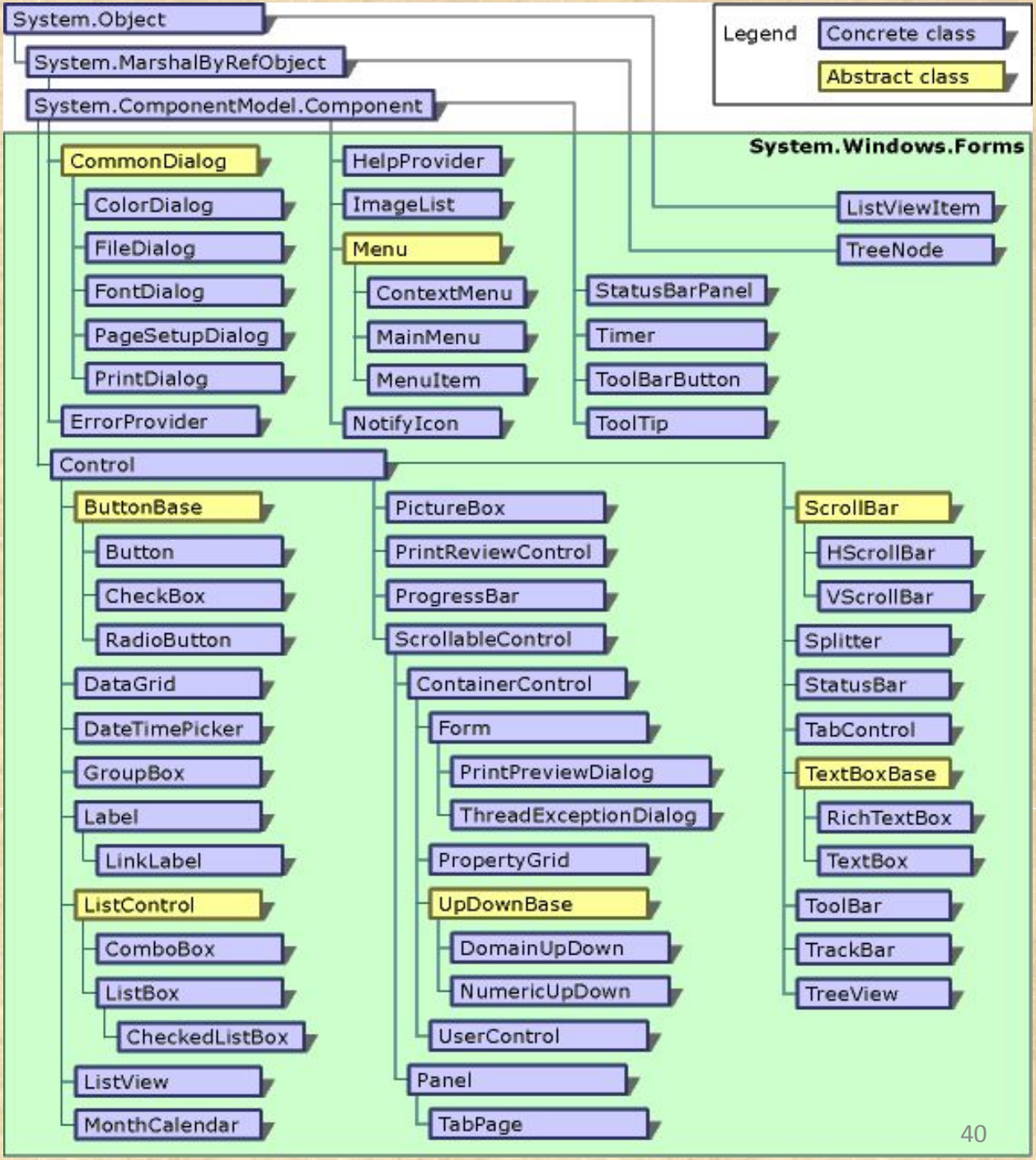
Основні особливості Windows-застосунку

- У застосунку Windows Forms форма є візуальною зоною, в якій відображається інформація для користувача.
- Як правило, створення застосунку Windows Forms відбувається шляхом додавання на форму елементів управління та написання відповідей (обробників подій) на дії користувача, такі як кліки мишки або натискання клавіш.
- Windows Forms включає широкий набір елементів керування, які можна додавати на форми: текстові поля, кнопки, випадаючі списки, перемикачі і навіть веб-сторінки.

Основні типи Windows.Forms

Application
ButtonBase, Button, CheckBox, ComboBox, DataGrid, GroupBox, ListBox, LinkLabel, PictureBox
Form
ColorDialog, FileDialog, FontDialog, PrintPreviewDialog
Menu, MainMenu, MenuItem, ContextMenu
Clipboard, Help, Timer, Screen, ToolTip, Cursors
StatusBar, Splitter, ToolBar, ScrollBar

Ієрархія елементів керування Windows.Forms



Створення Windows-застосунку

- При створенні нового проєкту C# у середовищі Visual Studio обирається один із можливих типів проєктів, в том числі Windows Application, Class Library, Web Control Library, ASP.NET, Application і ASP.NET Web Service. На базі зробленого вибору автоматично створюється каркас проєкту.
- Visual Studio.NET – це не тільки середовище для розробки програм на мові C#. Visual Studio.NET дозволяє створювати програми на мові VB, C#, C++, формувати Setup (інсталяційний пакет) розроблених програм і багато іншого.

Створення Windows-застосунку

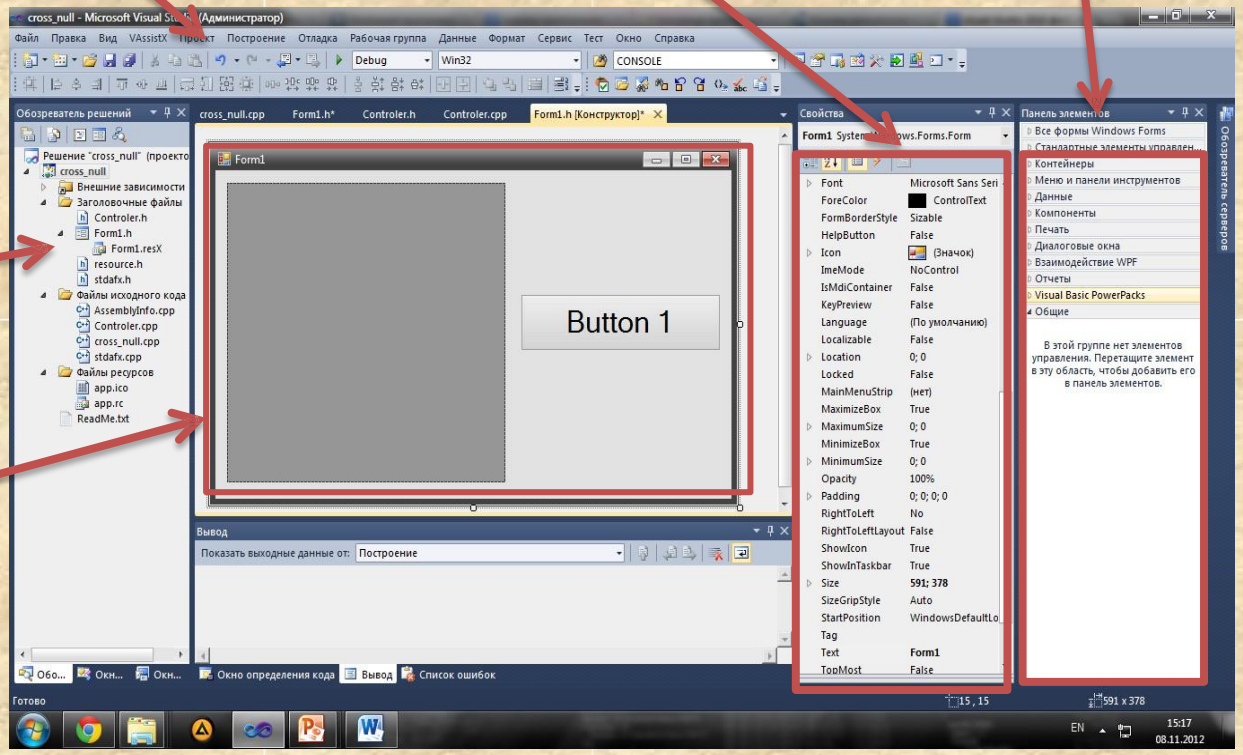
Панель меню

Панель властивостей (Properties)

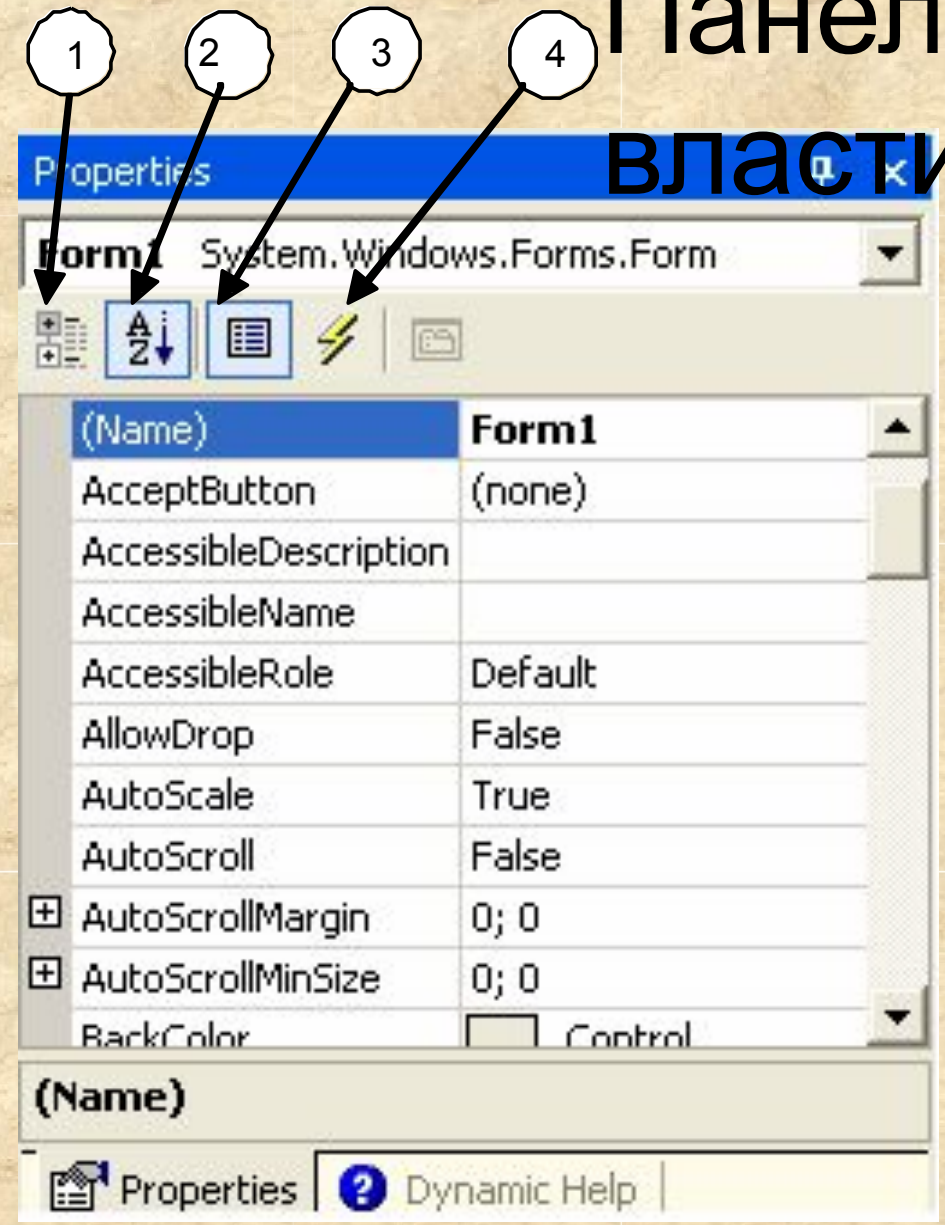
Панель елементів (ToolBox)

Провідник рішень (Solution explorer)

Робоча область



Панель властивостей



Встановлення властивостей виконується або вибором наявних у списку варіантів, або введенням потрібних значень із клавіатури. Якщо поблизу імені властивості знаходиться значок +, це значить, що властивість включає інші властивості. Вони стають доступними після натискання на значок.

1,2 – Тип сортування властивостей

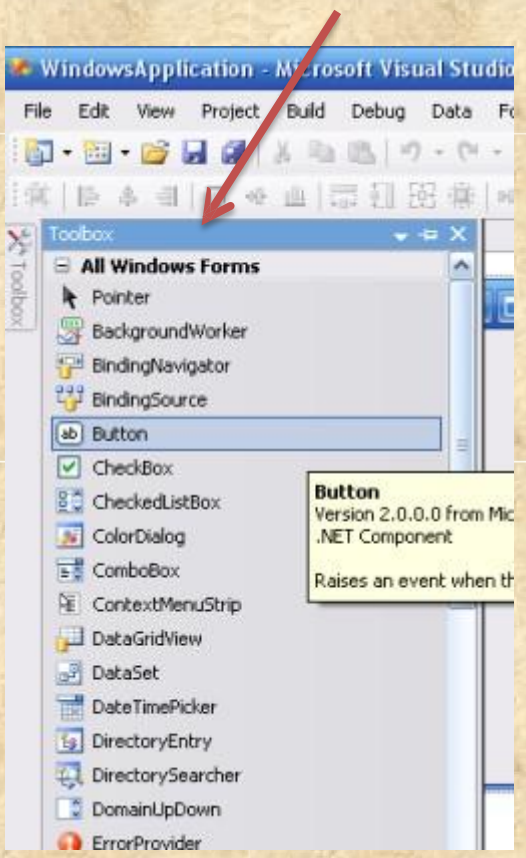
(за алфавітом, за категоріями)

3 – Властивості елементу керування

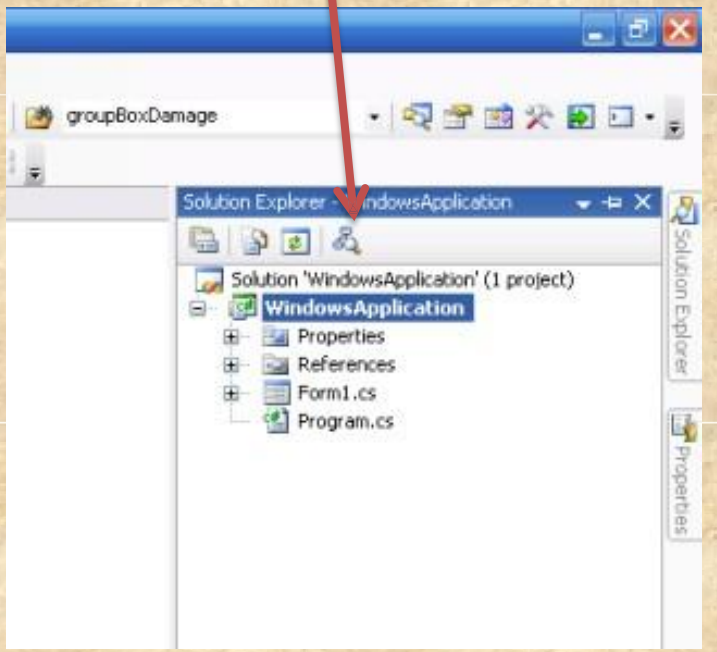
4 – Подія, яка виникає на елементах керування.

Панель ToolBox и Solution Explorer

ToolBox – панель елементів. Тут перераховані готові компоненти, які можна додавати на форму.



Solution Explorer відображає всі файли проєкта і використовуваний простір імен у розділі References.

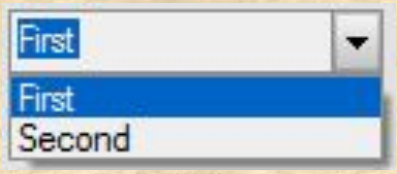


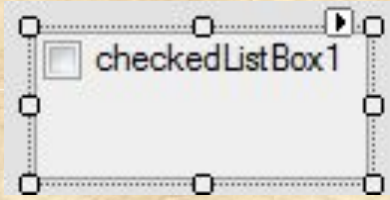
Основні елементи керування

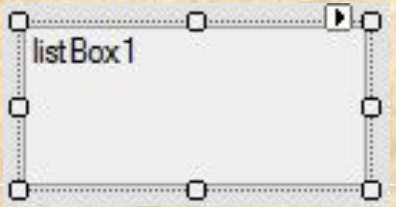
- Кнопка (button)
- Радіокнопка (RadioButton)
- Мітка (Label)
- Текстове поле (TextBox)
- Чекбокс (прапорець) (CheckBox)
- Розширене текстове поле (RichTextBox)

Основні елементи керування

- Випадаючий список (ComboBox)
- Список прапорців (CheckListBox)
- Список (ListBox)







Основні елементи керування

- Кнопка (Button)
  - Взаємодія користувача обмежується однією дією – натисканням. Все, що вам необхідно зробити при роботі із кнопкою, це помістити її на потрібне місце форми і назначити їй відповідний обробник.
- Радіокнопка (RadioButton)
  - Перемикач служить для вибору одного із декількох варіантів і тому розміщуються завжди групами. Щоб виділити їх в ізольовані групи, поміщають у контейнери: Panel або GroupBox.
- Мітка (Label)
  - Мітка найчастіше використовується для інформування користувача. Текст вноситься програмно і можливість редагування у користувача відсутня.
- Текстове поле (TextBox)
  - Використовується для введення і редагування тексту користувачем.
- Розширене текстове поле (RichTextBox)
  - Дозволяє розміщати графіку, абзаци, таблиці, формули та інше.

Основні елементи керування

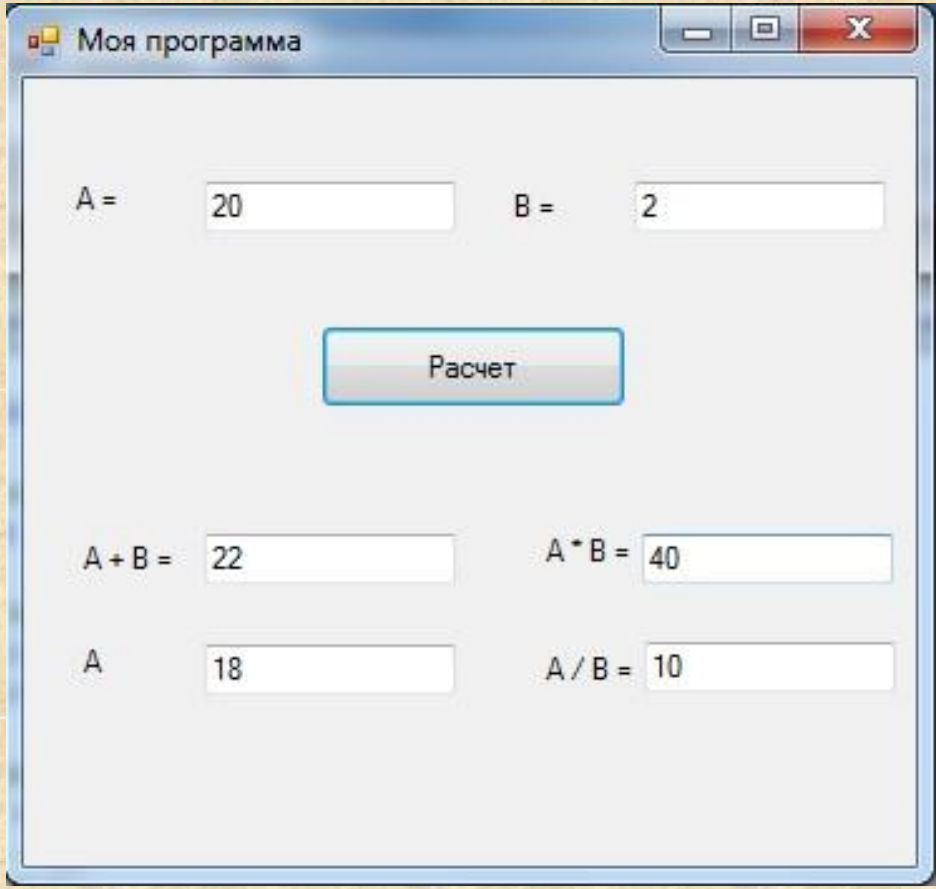
- Прапорець (CheckBox)
  - Виконує функції аналогічні радіокнопкам, але є повністю незалежними між собою у плані вибору декількох варіантів.
- Випадаючий список (ComboBox)
  - Список із можливістю вибору одного елементу із доступних варіантів при розкритті.
- Список прапорців (CheckListBox)
  - Дозволяє обрати потрібні прапорці із доступного списку.
- Список (ListBox)
  - Надає список, у якого можна обирати необхідні елементи.

Найбільш часто використовувані події

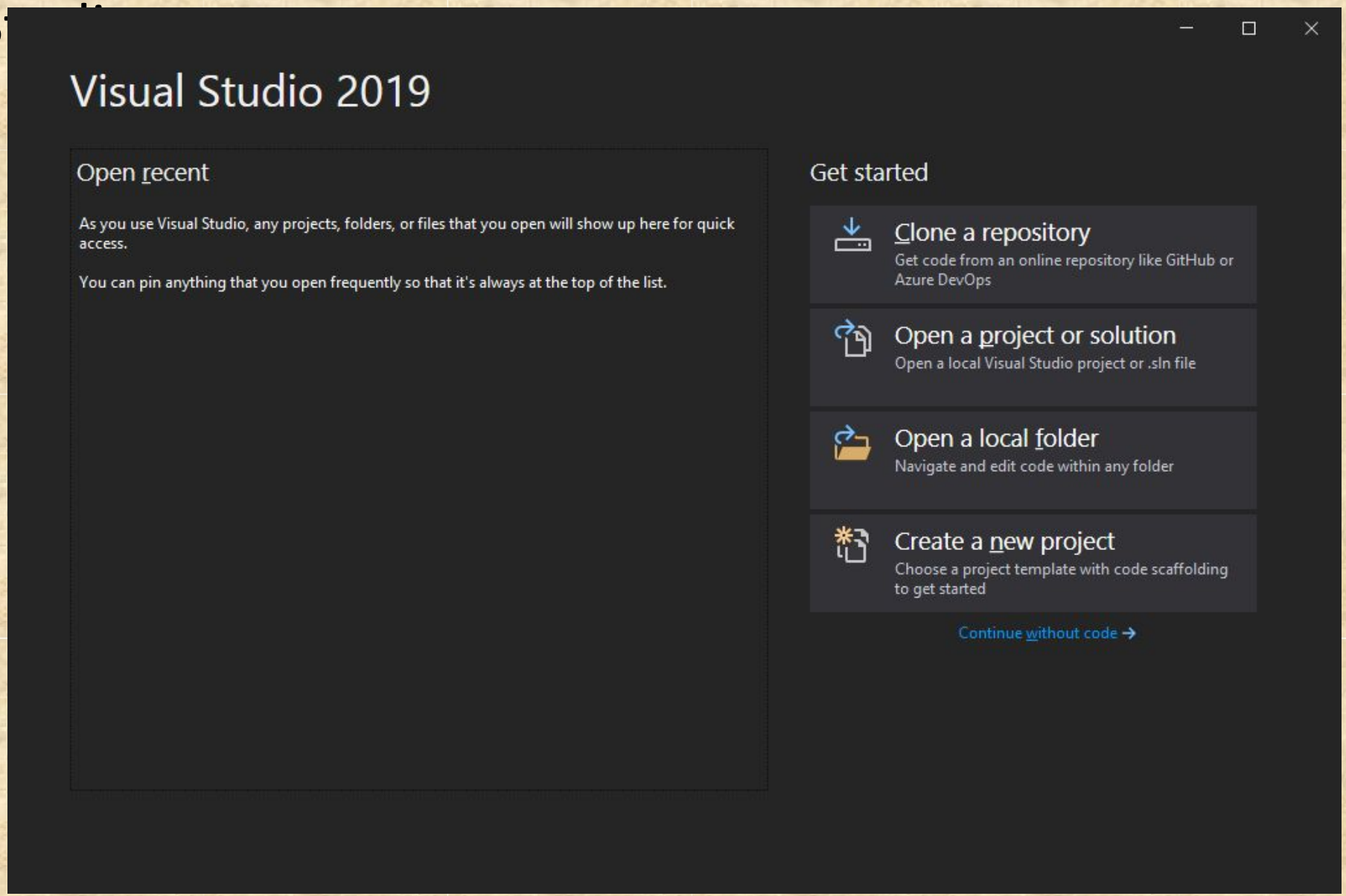
- Activated — отримання формою фокусу введення;
- Click, DoubleClick — одинарний і подвійний кліки мишки;
- Closed — закриття форми;
- Load — завантаження форми;
- KeyDown, KeyUp — натискання і відпускання будь-якої клавіші;
- MouseDown, MouseUp — натискання і відпускання кнопки миші;
- MouseMove — переміщення миші.

Створення Windows-застосунків у Visual Studio C#

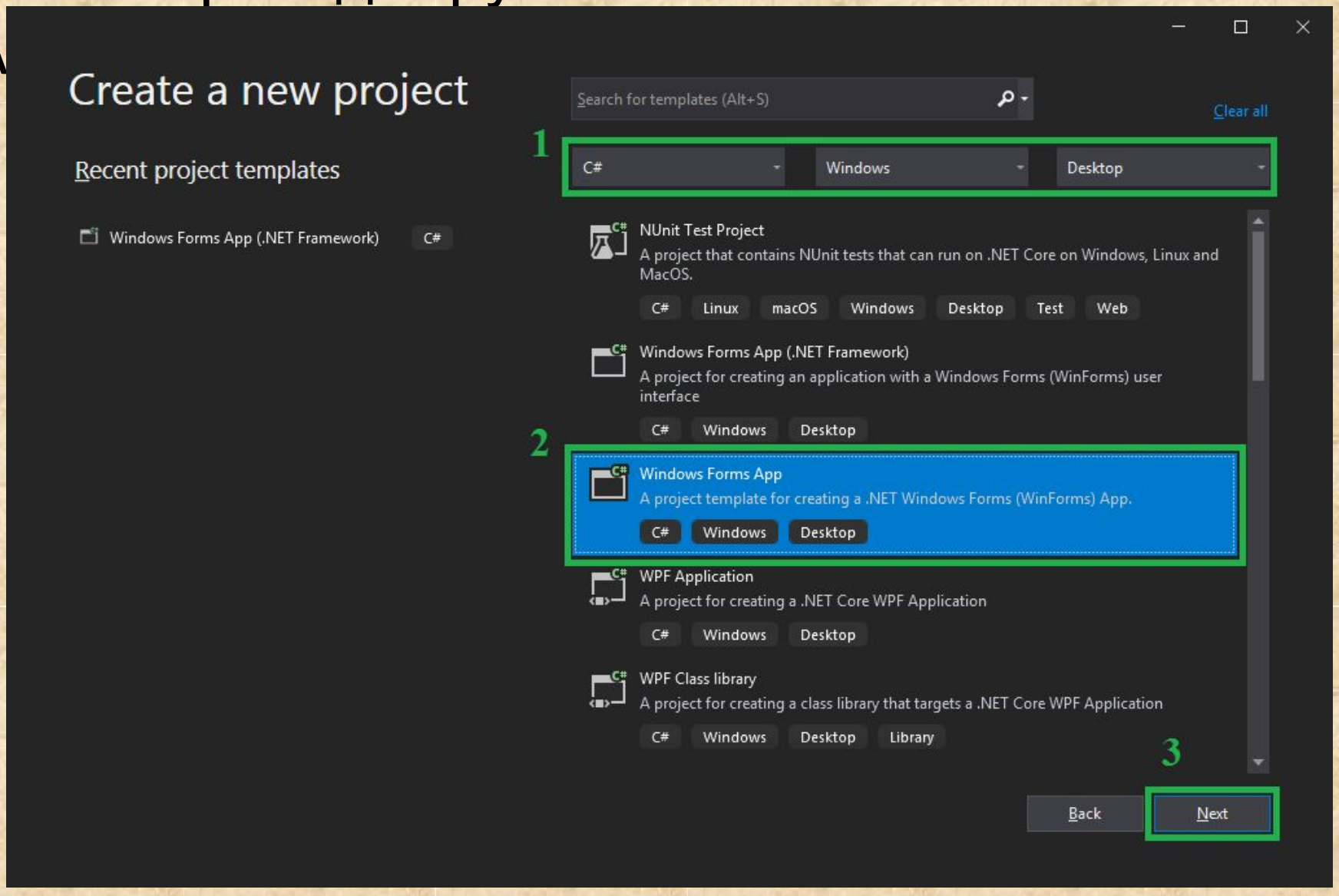
Розглянемо у цій темі на прикладу процес розробки програми-калькулятора, який дозволяє вводити 2 числа, і після натискання на кнопку «Обчислити» видавати результат операцій +, –, *, /.



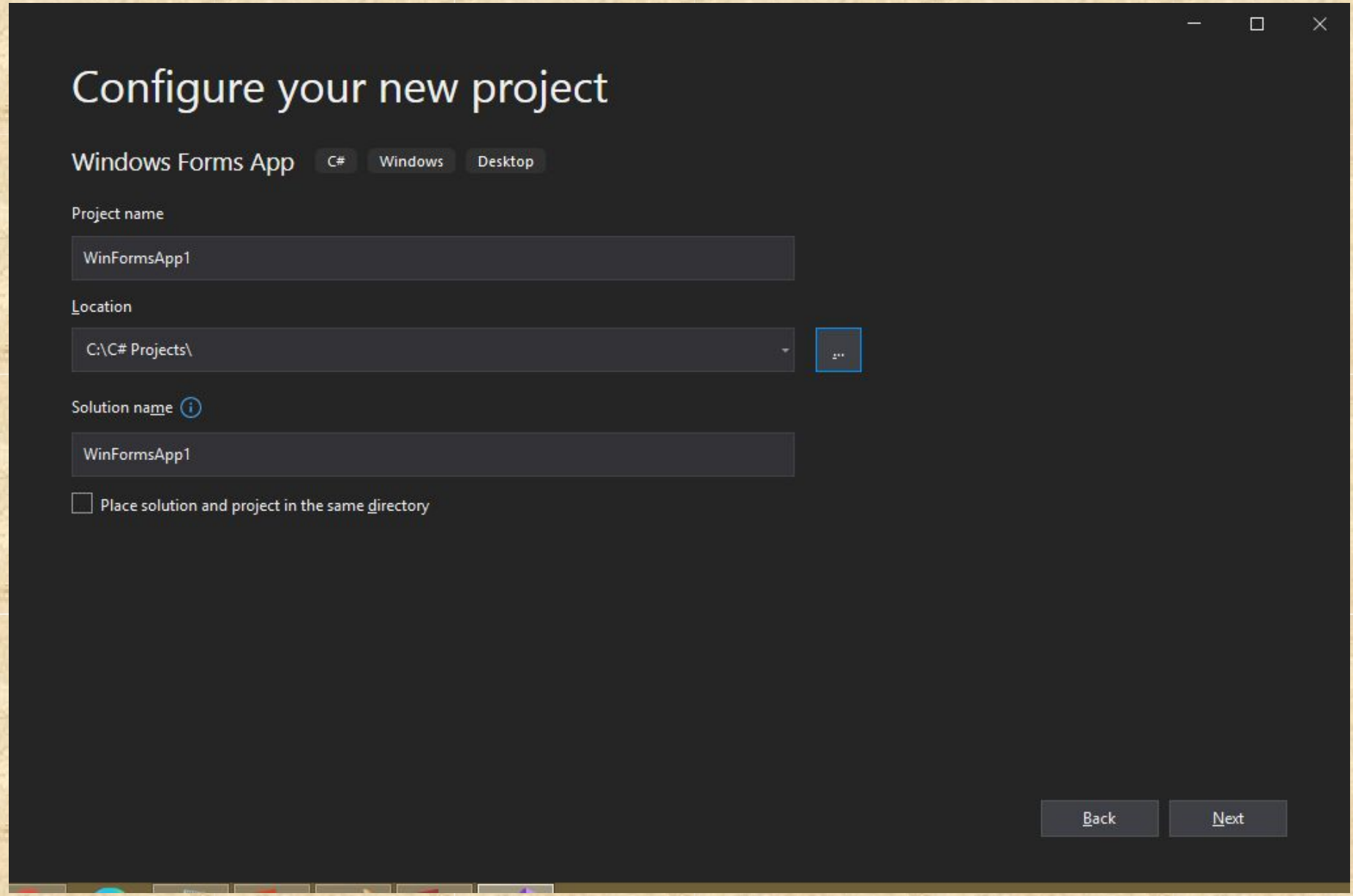
1. Запустити Microsoft Visual S



2. Створити новий проєкт (Create New Project -> Фільтри відбору->Windows Forms A



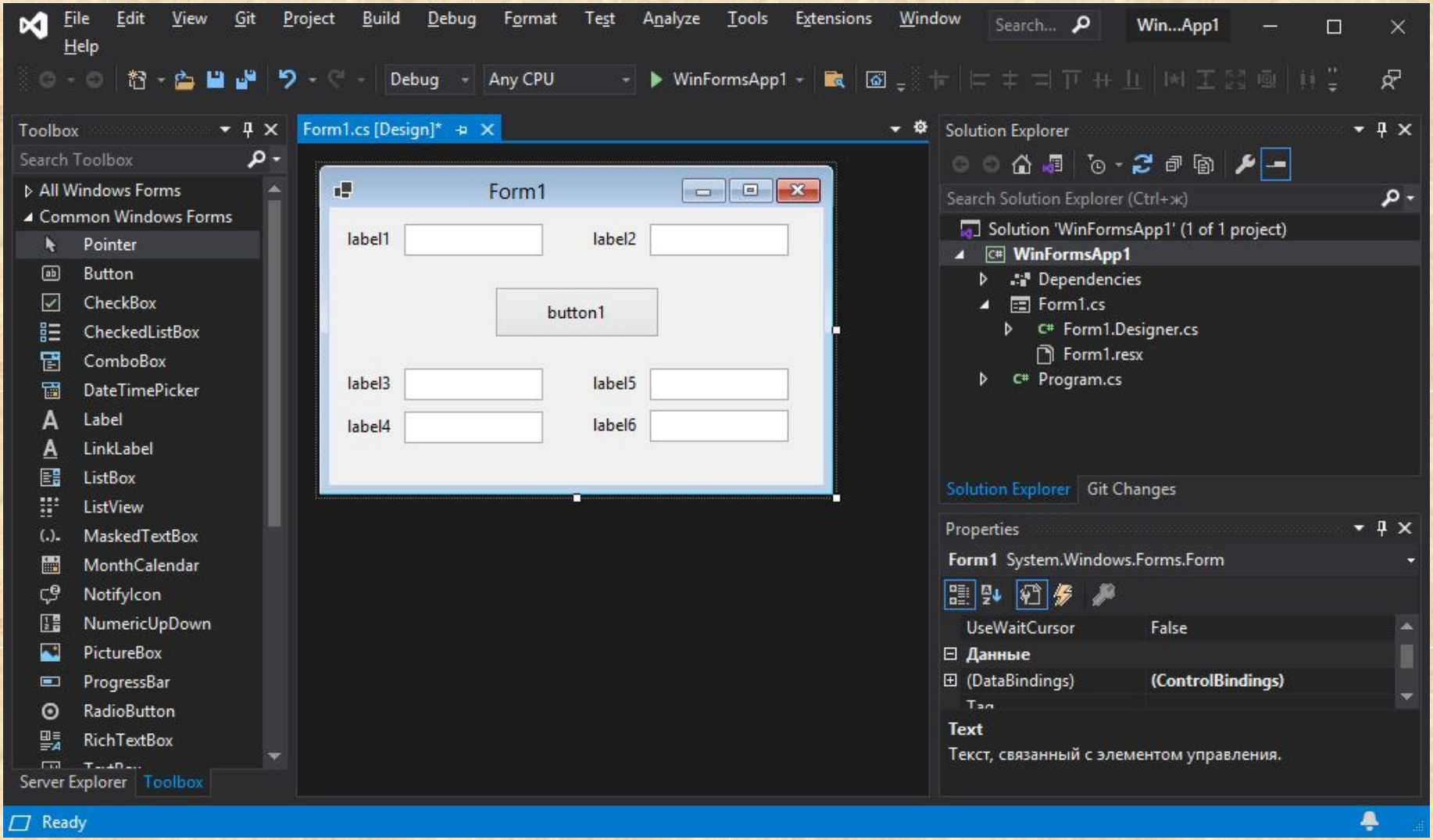
3. Вказати ім’я проєкту та вказати шлях



4. На формі (Form1) розташувати необхідні візуальні компоненти (панель Toolbox; у випадку її відсутності обрати пункт меню View –> Toolbox), створивши інтерфейс користувача. Додати, наприклад:

— текстові поля (TextBox) – для введення користувачем інформації (вхідних даних) або для її виведення (вихідних даних);
— мітку (Label) – для відображення (без можливості редагування) описової інформації;
— кнопку (Button) – для можливості запуску деяких дій (розрахунків, відкриття файлів і т.д.)

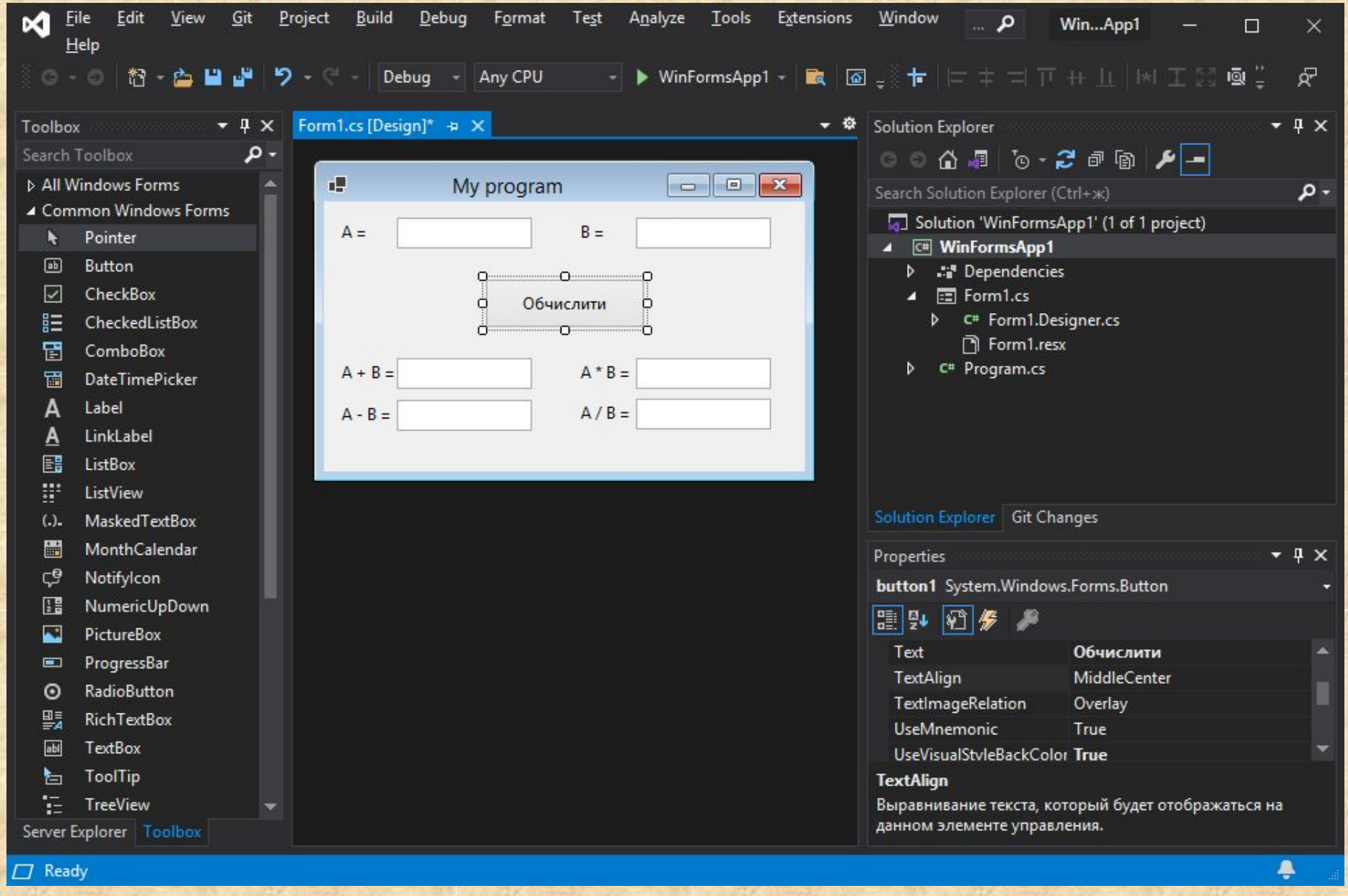
4.


File Edit View Git Project Build Debug Format Test Analyze Tools Extensions Window Help
Search...
Win...App1
Debug
Any CPU
WinFormsApp1
Toolbox
Search Toolbox
All Windows Forms
Common Windows Forms
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
Server Explorer
Toolbox
Form1.cs [Design]*
Form1
label1
label2
button1
label3
label5
label4
label6
Solution Explorer
Search Solution Explorer (Ctrl+ж)
Solution 'WinFormsApp1' (1 of 1 project)
WinFormsApp1
Dependencies
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Solution Explorer
Git Changes
Properties
Form1 System.Windows.Forms.Form
UseWaitCursor
False
Данные
(DataBindings)
(ControlBindings)
Text
Текст, связанный с элементом управления.
Ready

5. Змінити властивості деяких візуальних компонентів (панель Properties). Наприклад:

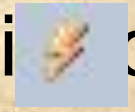
– Text – текст, видимий текст;
– Size – розмір об’єкта;
– Visible – видимість;
– Enabled – доступність;
– Font – шрифт.

5.

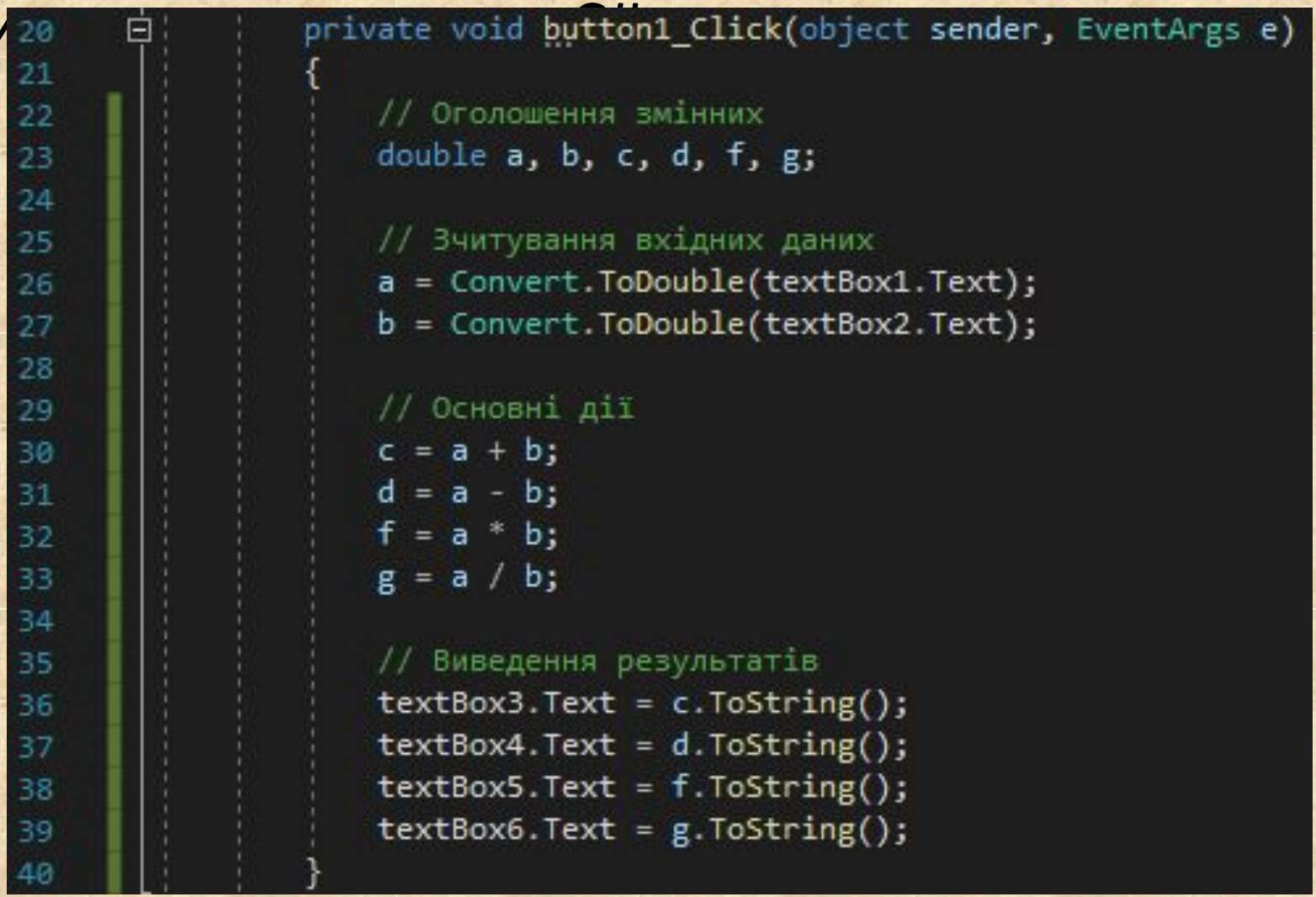

File Edit View Git Project Build Debug Format Test Analyze Tools Extensions Window Help
Win...App1
Debug
Any CPU
WinFormsApp1
Toolbox
Search Toolbox
All Windows Forms
Common Windows Forms
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
Server Explorer
Toolbox
Form1.cs [Design]*
My program
A =
B =
Обчислити
A + B =
A * B =
A - B =
A / B =
Solution Explorer
Search Solution Explorer (Ctrl+ж)
Solution 'WinFormsApp1' (1 of 1 project)
WinFormsApp1
Dependencies
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Solution Explorer
Git Changes
Properties
button1 System.Windows.Forms.Button
Text
Обчислити
TextAlign
MiddleCenter
TextImageRelation
Overlay
UseMnemonic
True
UseVisualStyleBackColor
True
TextAlign
Выравнивание текста, который будет отображаться на данном элементе управления.
Ready

6. Створити необхідні обробники подій.

– Якщо подія для об’єкта є «подією за замовчуванням» (наприклад, для кнопки – звичайне одинарне натискання – Click), то для створення обробника події необхідно двічі клацнути по відповідному компоненту. Після цього з’являється можливість для написання тексту програми на мові C# для обробки відповідної події;

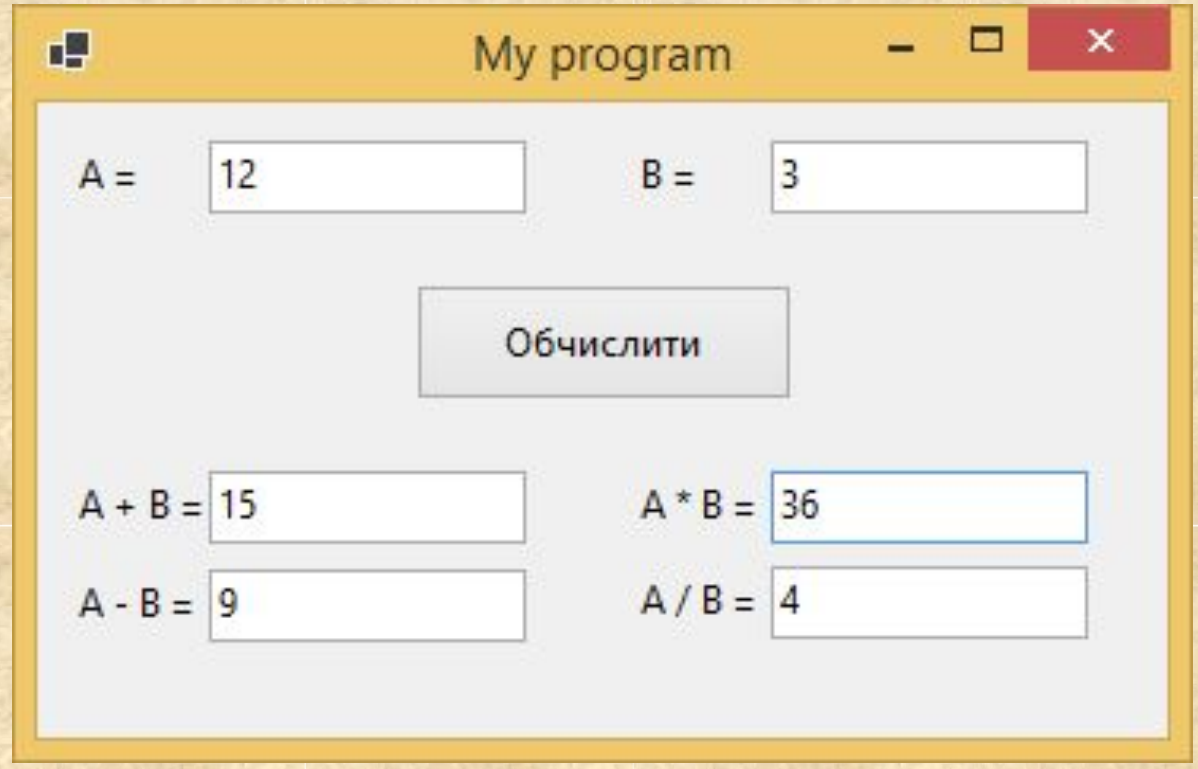
– для створення обробника інших подій необхі  о на панелі Properties натиснути Events і зі списку обрати відповідну

7. Написати текст для відповідного обробника подій у відповідності з си



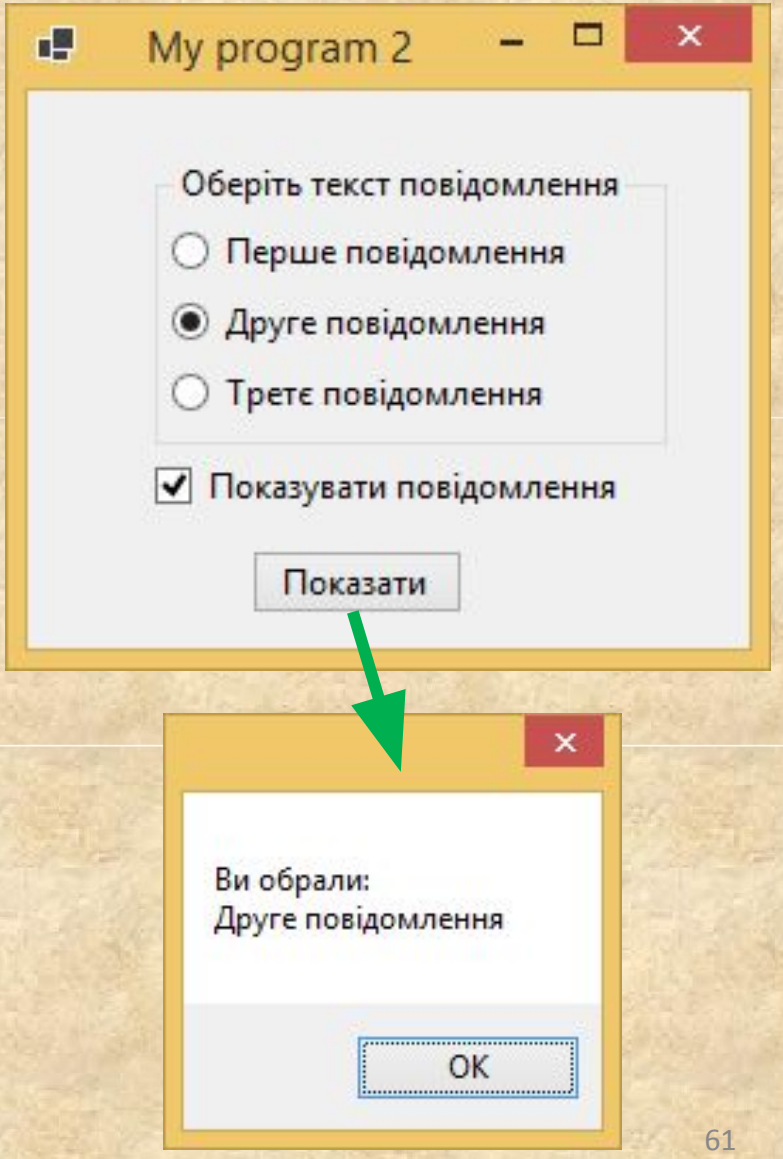
```
20      private void button1_Click(object sender, EventArgs e)
21      {
22          // Оголошення змінних
23          double a, b, c, d, f, g;
24
25          // Зчитування вхідних даних
26          a = Convert.ToDouble(textBox1.Text);
27          b = Convert.ToDouble(textBox2.Text);
28
29          // Основні дії
30          c = a + b;
31          d = a - b;
32          f = a * b;
33          g = a / b;
34
35          // Виведення результатів
36          textBox3.Text = c.ToString();
37          textBox4.Text = d.ToString();
38          textBox5.Text = f.ToString();
39          textBox6.Text = g.ToString();
40      }
```

8. Запустити проєкт на виконання (АБО натиснути f5 на клавіатурі, АБО кно у , АБО обрати пункт меню Debug –> Start Debugging)



Приклад Windows-застосунку 2

Розроблювана у прикладі програма буде виконувати наступні функції: радіокнопки встановлюють текст повідомлення, яке буде виводитися після натискання на кнопку. Прапорець повинен визначати – виводити повідомлення чи ні.



Приклад Windows-застосунку 2

- Додайте на форму один елемент керування GroupBox, три елемента RadioButton, один елемент CheckBox і один елемент Button. Всі три радіокнопки повинні бути поміщеними в один GroupBox. Інакше вони не будуть пов`язані між собою.
- Тепер змініть деякі властивості доданих елементів:
  - button1: Text - Показати
  - groupBox1: Text - Оберіть текст повідомлення
  - radioButton1: Text – Перше повідомлення
  - radioButton2: Text – Друге повідомлення
  - radioButton3: Text – Третє повідомлення
  - checkBox1: Text – Показувати повідомлення, Checked - True

Обробник Windows-застосунку 2

```
20      private void button1_Click(object sender, EventArgs e)
21      {
22          // Змінна для зберігання обраного повідомлення
23          string strmessage = "";
24          if (radioButton1.Checked == true)
25          {
26              // якщо обрано саме цю кнопку
27              // то копіюємо текст кнопки у змінну
28              strmessage = radioButton1.Text;
29          }
30          // перевіряємо другу радіокнопку
31          else if (radioButton2.Checked == true)
32          {
33              strmessage = radioButton2.Text;
34          }
35          else if (radioButton3.Checked == true)
36          { strmessage = radioButton3.Text; }
37          // перевіряємо, чи прапорець відмічений
38          // якщо так, то виводимо обране повідомлення на екран
39          if (checkBox1.Checked == true)
40          {
41              MessageBox.Show("Ви обрали: \n" + strmessage);
42          }
43      }
```

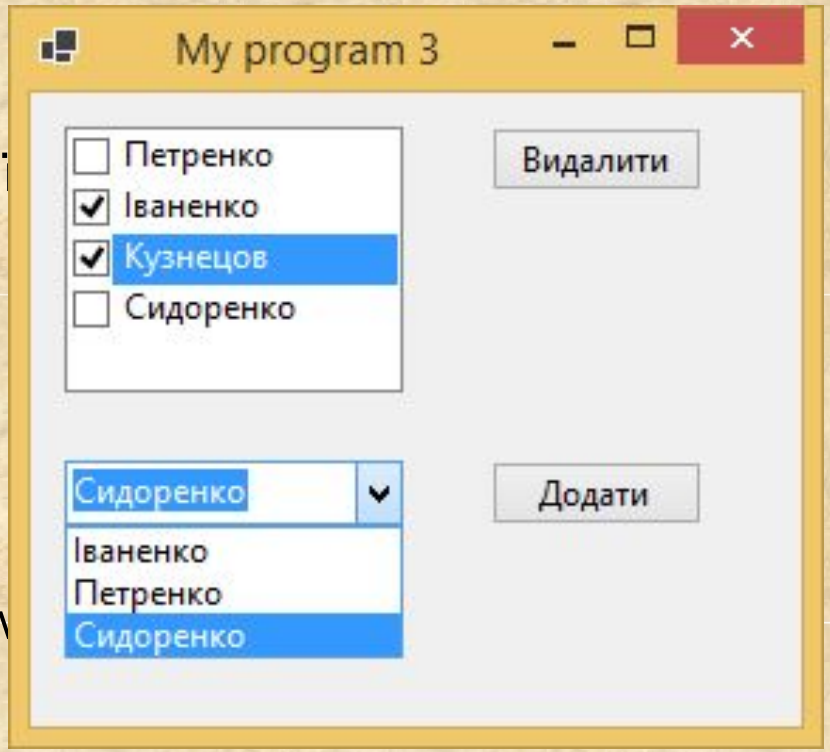
Приклад Windows-застосунку 3

Напишемо програму призначену для обліку даних про учасників змагань. Програма буде містити два списки:

–ComboBox – для введення інформації про учасників;

–CheckedListBox – для зберігання і обробки даних.

За допомогою списку ComboBox користувач буде обирати прізвища людей, яких необхідно додати в список учасників. Дві кнопки на форм будуть додавати або видаляти користувачів із списку.

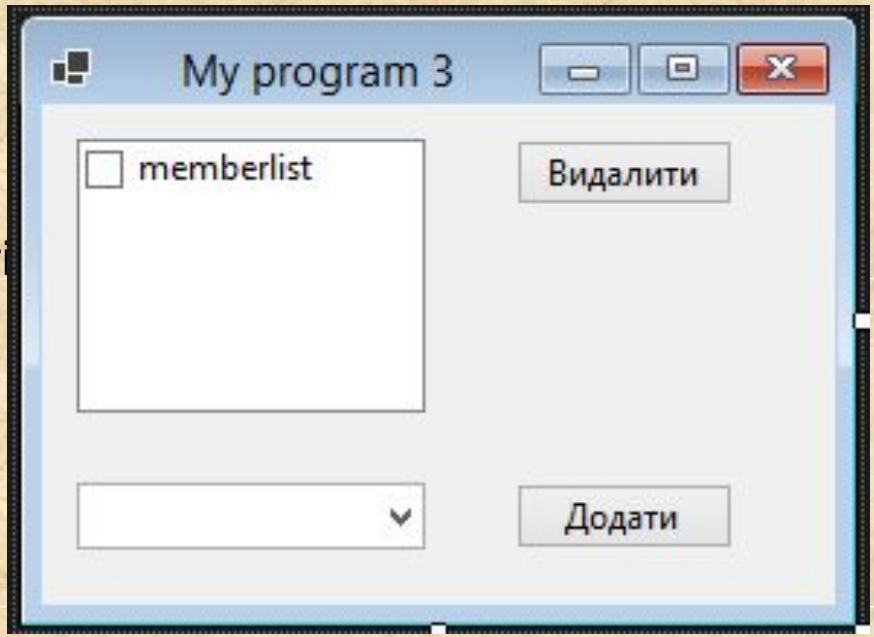


Приклад Windows-застосунку 3

Створіть наступну форму

- Змініть деякі властивості створеної форми:
  - Text - «робота із списками»

- Тепер змінимо властивості елементі керування
  - CheckedListBox: Name - memberlist
  - comboBox1: Name - peoplelist ; Text - «»
  - button1: Name – buttonAdd; Text - «Додати»
  - button2: Name – buttonDelete; Text - «Видалити»



Приклад Windows-застосунку 3

- Створимо елемент керування ComboBox, який має ім`я peoplelist, списком прізвищ передбачуваних учасників змагань. Для цього у вікні властивостей peoplelist оберіть властивість Items. Відкрийте вікно String Collection Editor, натиснувши на кнопку із трьома точками в поле Items.
- Додайте обробники для кнопок «Додати» і «Видалити», двічі клацнувши лівою кнопкою миші по кожній із кнопок.

Приклад Windows-застосунку 3. Обробник подій для кнопки «Додати»

```
20      private void buttonAdd_Click(object sender, EventArgs e)
21      {
22          //Працюємо із списком для введення прізвища (peoplelist)
23          //Перевіряємо, чи обраний якийсь елемент списку
24          if (peoplelist.Text.Length != 0)
25          {
26              //Якщо елемент вибрано, то додаємо його у список учасників
27              memberlist.Items.Add(peoplelist.Text);
28          }
29          else
30          {
31              //Якщо елемент не обрано, то виводимо повідомлення
32              MessageBox.Show("Оберіть елемент із списку або введіть новий");
33          }
34      }
```

Приклад Windows-застосунку 3. Обробник подій для кнопки «Видалити»

```
36          private void buttonDelete_Click(object sender, EventArgs e)
37          {
38              //Поки список відмічених елементів непорожній
39              while (memberlist.CheckedIndices.Count > 0)
40              {
41                  //Видаляємо із списку учасників по одному елементу
42                  //при цьому список відмічених елементів автоматично оновлюється
43                  //Таким чином, кожний раз нульовий елемент із CheckedIndices
44                  //буде містити індекс першого обраного у списку об`єкта
45                  memberlist.Items.RemoveAt(memberlist.CheckedIndices[0]);
46                  //При видаленні із списку останнього позначеного елементу
47                  // CheckedIndices.Count буде дорівнювати нулю
48                  //і цикл автоматично завершиться
49              }
50          }
```