Лекція № 1
ОСНОВНІ ЗАСАДИ ПРОГРАМУВАННЯ МЕХАНІЗМІВ БЕЗПЕКИ

Лектор:
Копитін Юрій Вікторович

Основні нормативні документи, які регламентують порядок створення, впровадження, супроводження та модернізації засобів технічного захисту інформації від несанкціонованого

- НД ТЗІ 1.1-002-99 Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу.
- НД ТЗІ 1.1-003-99 Термінологія в галузі захисту інформації в комп’ютерних системах від несанкціонованого доступу.
- НД ТЗІ 3.6-001-2000 Технічний захист інформації. Комп’ютерні системи. Порядок створення, впровадження, супроводження та модернізації засобів технічного захисту інформації від несанкціонованого доступу.
- НД ТЗІ 2.5-004-99 Критерії оцінки захищеності інформації в комп’ютерних системах від несанкціонованого доступу.
- НД ТЗІ 2.5-005-99 Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу.
- Положення про державну експертизу в сфері технічного захисту інформації. Наказ Адміністрації Держспецзв’язку від 16.05.2007 № 93, зареєстрований в Міністерстві юстиції України 16.07.2007 за № 820/14087 із змінами, затвердженими наказом Адміністрації Держспецзв’язку від 10.10.2012 № 567, зареєстрованим в Міністерстві юстиції України 06.11.2012 за № 1863/22175.
- НД ТЗІ 2.7-009-09 Методичні вказівки з оцінювання функціональних послуг безпеки в засобах захисту інформації від несанкціонованого доступу.
- НД ТЗІ 2.7-010-09 Методичні вказівки з оцінювання рівня гарантій коректності реалізації функціональних послуг безпеки в засобах захисту інформації від несанкціонованого доступу.
- НД ТЗІ 2.6-001-11 Порядок проведення робіт з державної експертизи засобів технічного захисту інформації від несанкціонованого доступу та комплексних систем захисту інформації в інформаційно-телекомунікаційних системах.
- ДСТУ 3918-99 Інформаційні технології. Процеси життєвого циклу програмного забезпечення.
- ГОСТ 19.101-77 Единая система программной документации. Виды программ и программных документов.
- ГОСТ 19.501-78 Единая система программной документации. Формуляр. Требования к содержанию и оформлению.

Поняття «механізм безпеки»

Механізм забезпечення безпеки – взаємопов'язана сукупність організаційних, апаратних, програмних та програмно-апаратних засобів, способів, методів, правил та процедур, використовуваних для реалізації вимог безпеки мережі.

Механізми безпеки (відповідно до рекомендації ITU-T X.800)

Шифрування

Електронний цифровий підпис

Механізм управління доступом

Механізм контролю цілісності даних

Механізм аутентифікації

Механізм доповнення трафіку

Механізм управління маршрутизацією

Механізм нотарізації

Механізми безпеки (відповідно до НД ТЗІ 2.5-004-99)

Критерії конфіденційності
Довірча конфіденційність
Адміністративна конфіденційність
Повторне використання об'єктів
Аналіз прихованих каналів
Конфіденційність при обміні

Критерії цілісності
Довірча цілісність
Адміністративна цілісність
Відкат
Цілісність при обміні

Критерії доступності
Використання ресурсів
Стійкість до відмов
Гаряча заміна
Відновлення після збоїв

Критерії спостереженості
Реєстрація
Ідентифікація і автентифікація
Достовірний канал
Розподіл обов'язків
Цілісність комплексу засобів захисту
Самотестування
Ідентифікація і автентифікація при обміні
Автентифікація відправника
Автентифікація одержувача

Процес розробки програмного забезпечення

Процес розробки програмного забезпечення складається із:

- чіткої поставки завдання;
- обрання мови програмування для розробки;
- обрання оптимальної структури для представлення даних;
- розробки алгоритму;
- написання і документування надійної та легко модифікованої програми;
- забезпечення її вичерпного тестування.

Середовища розробки для C#

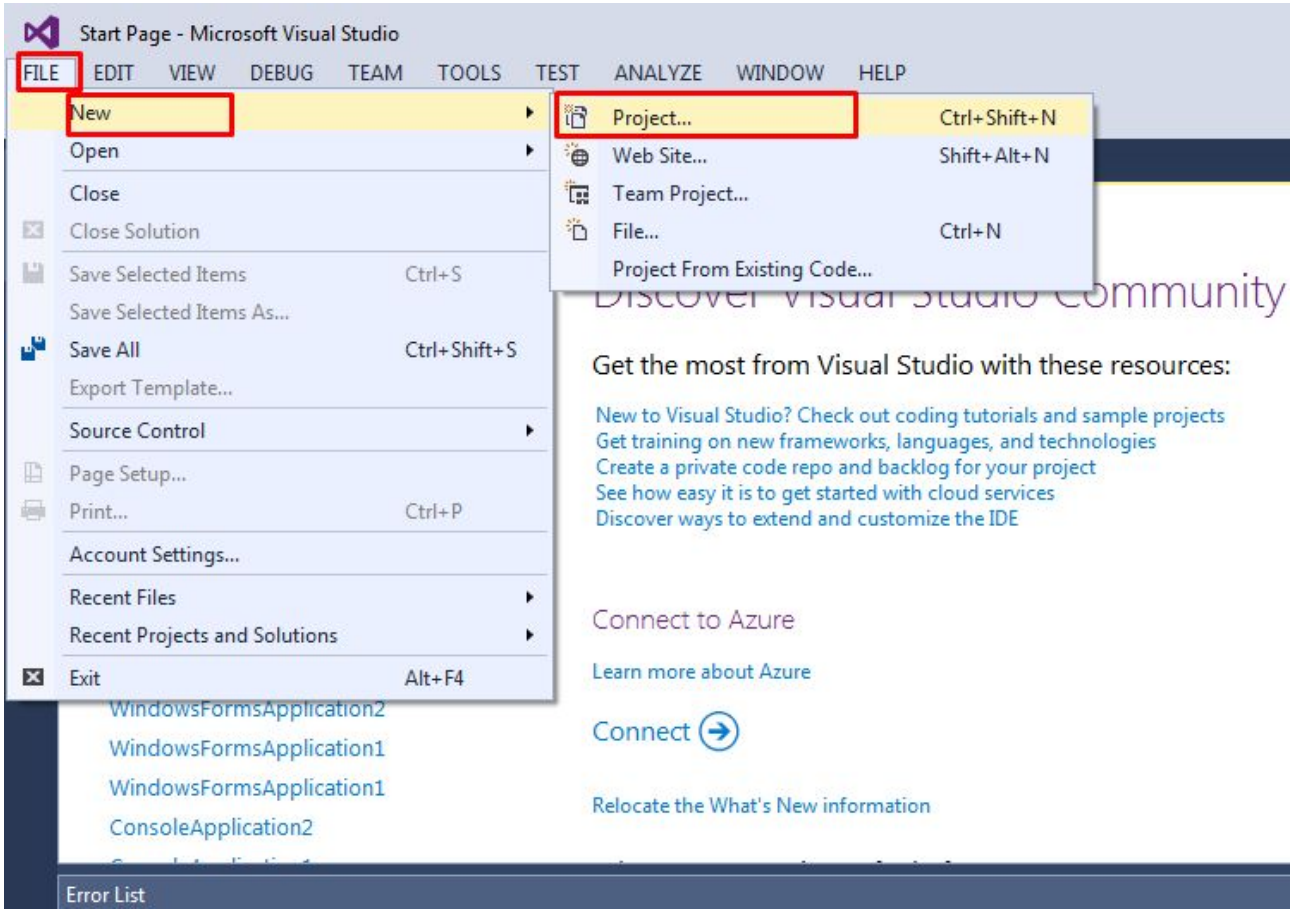
Visual Studio, SharpDevelop, Xamarin Studio

Типи проектів у середовищі розробки Visual Studio

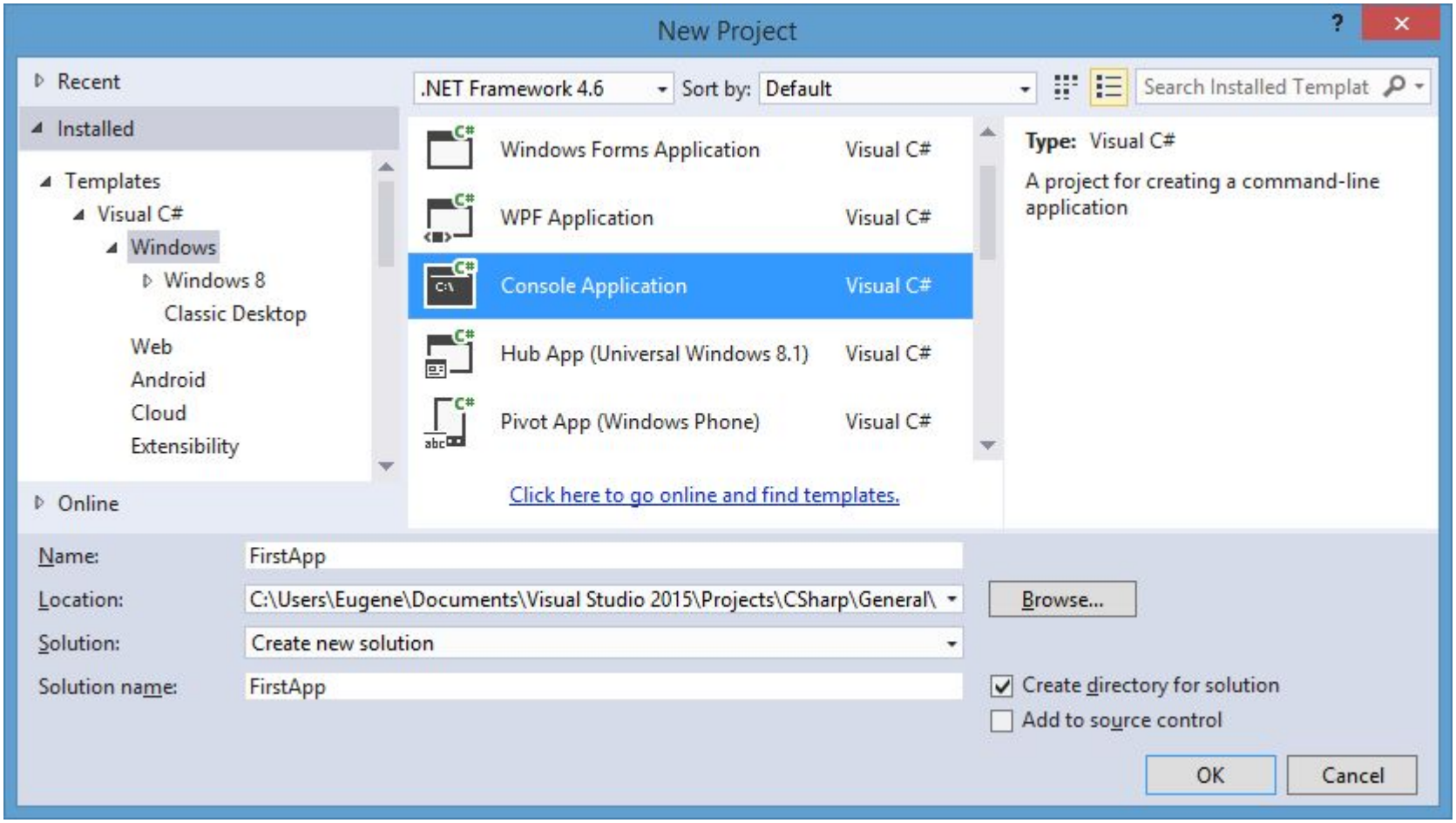
Windows-застосування	- це застосування, яке використовує елементи інтерфейсу Windows, включаючи форми, кнопки, прапорці тощо.
Консольне застосування	- це застосування, яке виконує виведення «на консоль».
Бібліотека класів	- об'єднує класи, які призначені для використання в інших застосуваннях.
Веб-застосування	- це застосування, доступ до якого виконується через браузер та яке по запиту формує веб-сторінку та відправляє її клієнту мережею.
Веб-сервіс	- це компонент, методи якого можуть викликатися через Інтернет.

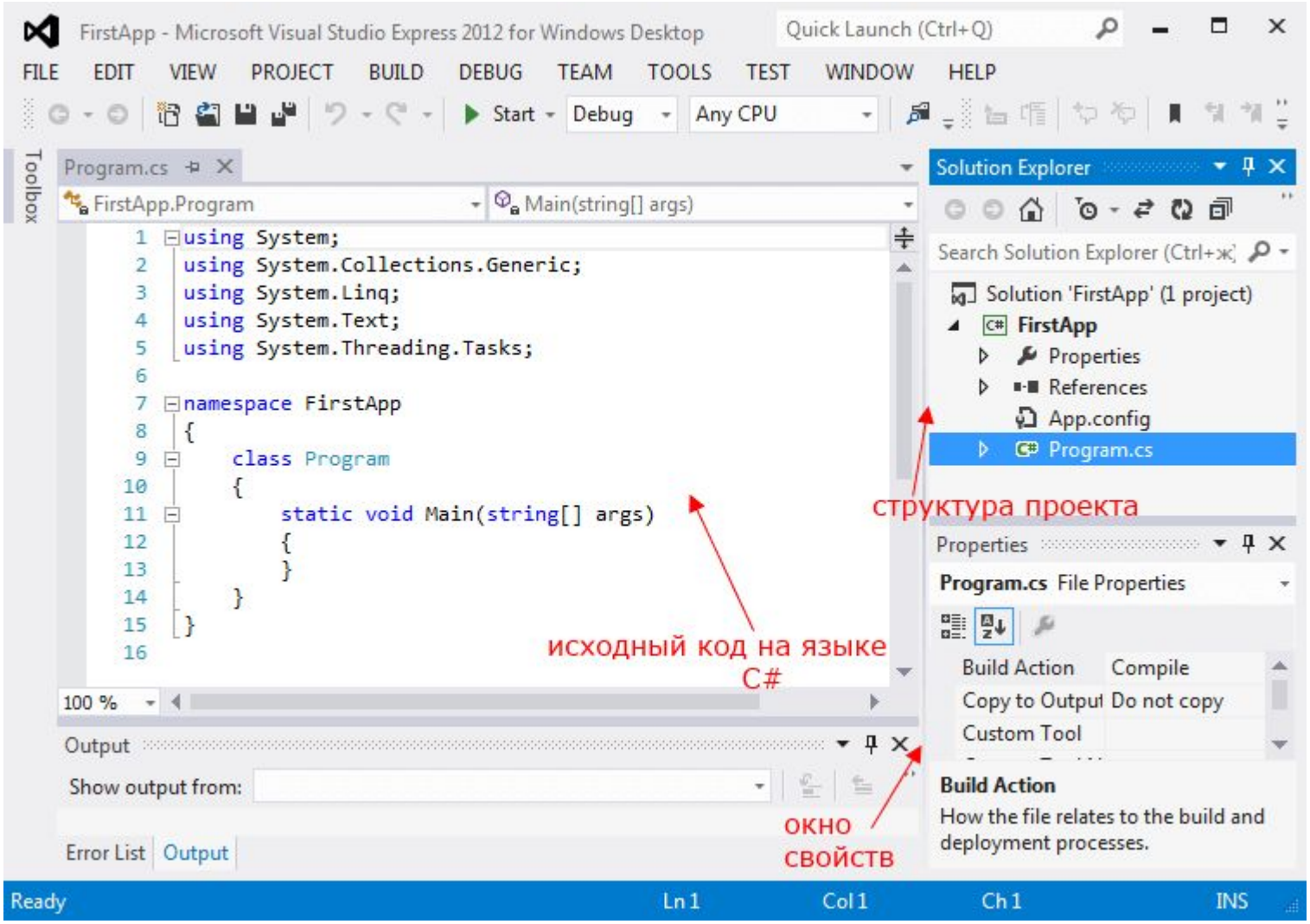
Перше консольне застосування

File->New->Project…



Оберіть пункт «Console Application»




FirstApp - Microsoft Visual Studio Express 2012 for Windows Desktop
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP
Start
Debug
Any CPU
Toolbox
Program.cs
FirstApp.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace FirstApp
{
class Program
{
static void Main(string[] args)
{
}
}
}
100 %
исходный код на языке C#
Output
Show output from:
Error List
Output
Solution Explorer
Search Solution Explorer (Ctrl+ж)
Solution 'FirstApp' (1 project)
FirstApp
Properties
References
App.config
Program.cs
структура проекта
Properties
Program.cs File Properties
Build Action Compile
Copy to Output Do not copy
Custom Tool
Build Action
How the file relates to the build and deployment processes.
окно свойств
Ready
Ln 1
Col 1
Ch 1
INS

```
/* Початок секції просторів імен, що підключаються */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/* Кінець секції просторів імен, що підключаються */

namespace FirstApp /* оголошення нового простору імен */
{
    class Program  /* оголошення нового класу */
    {
        static void Main(string[] args) /* оголошення нового методу */
        {

        } /* кінець оголошення нового методу */


    } /* кінець оголошення нового класу */

} /* кінець оголошення нового простору імен */
```

Простір імен – це декларативна область, в рамках якої визначаються різні ідентифікатори (імена типів, функцій, змінних тощо).
Простори імен використовуються для організації коду у вигляді логічних груп та з метою уникнення конфліктів імен, які можуть виникнути, особливо в таких випадках , коли база коду включає декілька бібліотек.

Клас – це спеціальна конструкція, яка використовується для групування пов'язаних змінних та функцій.
При цьому глобальні змінні класу називаються полями даних, а функції – методами класу. Створений та ініціалізований екземпляр класу називають об'єктом класу. На основі одного класу, можна створити безліч об'єктів, що відрізнятимуться один від одного своїм станом (значеннями полів).

```
using System;

namespace FirstApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Calculator calc = new Calculator(); // створення об'єкту нового класу
            calc.Add(2, 3); // виклик методу Add нового класу
        }
    }

    // оголошення нового класу
   class Calculator
   {
       public void Add(int x, int y)
       {
           int z = x + y;
            Console.WriteLine("Сумма {0} и {1} равна {2}", x, y, z);
           Console.ReadLine();
       }
   }
}
```

Методи класу Console

Beep: подача звукового сигналу

Clear: очищення консолі

WriteLine: виведення рядку тексту, включаючи символ повернення каретки (тобто з переведенням на новий рядок)

Write: виведення рядку тексту, але без символу повернення каретки

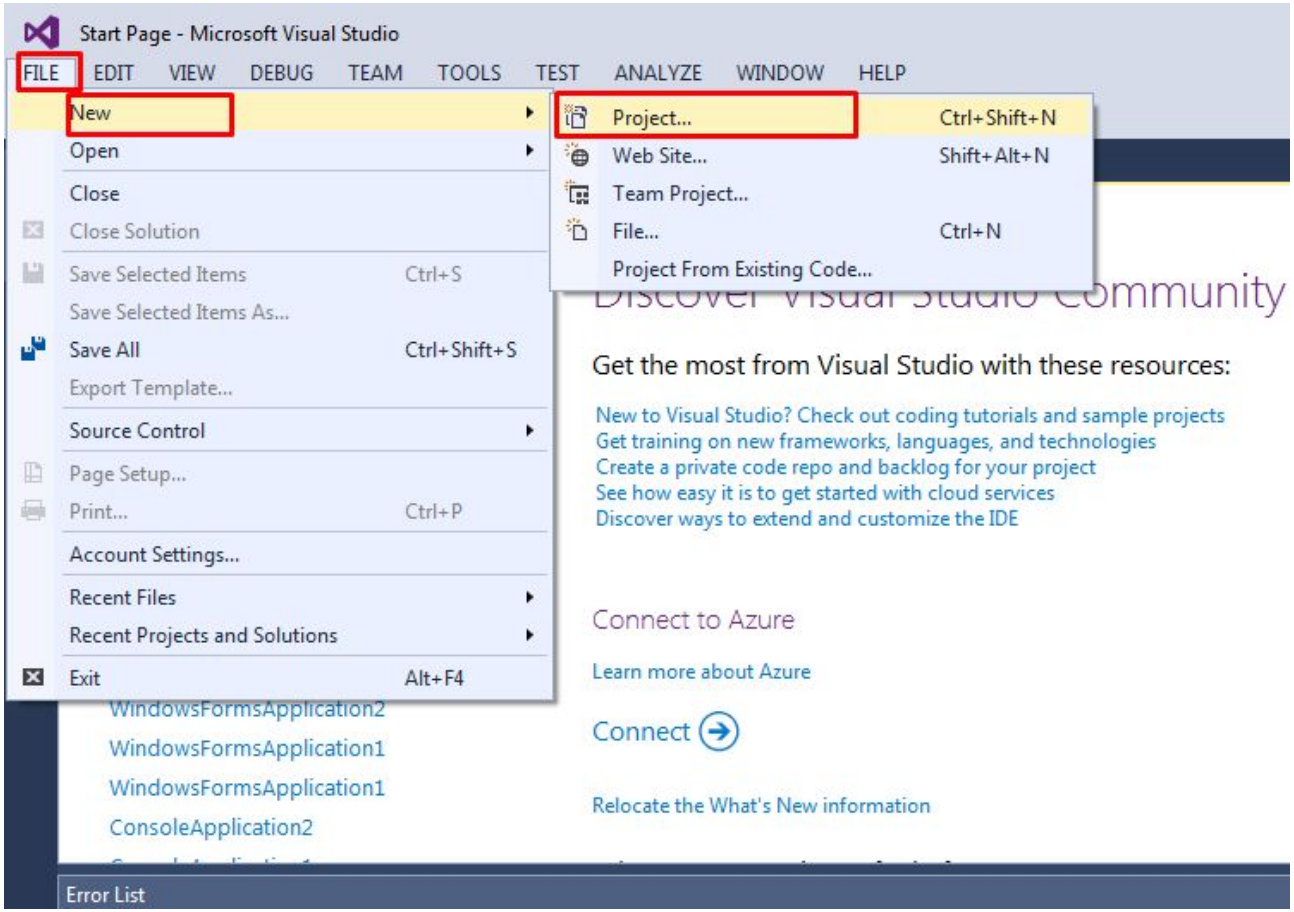
ReadLine: зчитування рядуа тексту із вхідного потоку

Read: зчитування введеного символу у вигляді числового коду даного символу.

ReadKey: зчитування натиснутої клавіші клавіатури

Перше застосування Windows Forms

File->New->Project…



У вікні, що з’явилось, слід обрати Windows Forms Application

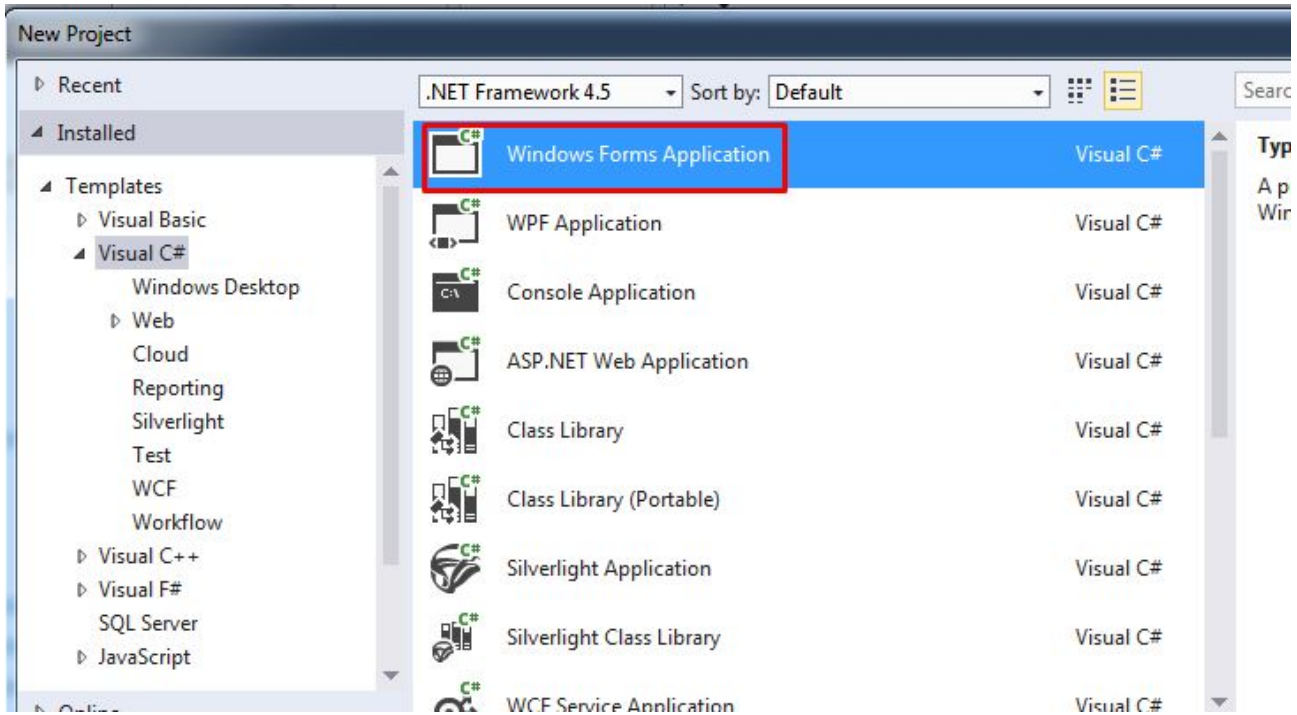
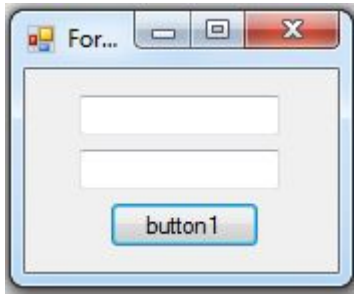


Рис. 4 Створення нового проекту (2-й крок)

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

namespace FirstApp
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         string s;
         s = textBox1.Text;
         s += " some info";
         textBox2.Text = s;
      }
   }
}
```



Коментар до коду

// однорядковий коментар

/* багаторядковий
коментар */

Вбудовані типи даних

bool: зберігає значення true або false.
byte: зберігає ціле число від 0 до 255 і займає 1 байт.
sbyte: зберігає ціле число від -128 до +127 і займає 1 байт.
short: зберігає ціле число від -32768 до 32767 і займає 2 байта.
ushort: зберігає ціле число від 0 до 65535 і займає 2 байта.
int: зберігає ціле число від -2147483648 до 2147483647 і займає 4 байта.
uint: зберігає ціле число від 0 до 4294967295 і займає 4 байта.
long: зберігає ціле число від -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 і займає 8 байт.
ulong: зберігає ціле число від 0 до $18 * 10_{18}$ і займає 8 байт.
float: зберігає число з плаваючою крапкою від $1.5 * 10^{-45}$ до $3.4 * 10_{38}$ і займає 4 байта.
double: зберігає число з плаваючою крапкою від $5.0 * 10^{-324}$ до $1.7 * 10_{308}$ і займає 8 байта.
decimal: зберігає десяткове дробове число від $1.0 * 10^{-28}$ до $7.9 * 10_{28}$ і займає 16 байт.
char: зберігає одиночний символ в кодуванні Unicode і займає 2 байта.
string: зберігає набір символів Unicode.

Оголошення змінних

```
тип_даних
назва_змінної
```

В якості ім'я змінної може виступати довільна назва, яке задовольняє наступним вимогам:

- ім'я повинне містити не більше 255 символів;

- ім'я може містити будь-які цифри, букви і символ підкреслення, при цьому перший символ в імені повинен бути буквою або символом підкреслення;

- в імені не повинно бути знаків пунктуації та прогалин;

- ім'я не може бути ключовим словом мови C #.

```
bool isEnabled = true;
```

Математичні операції

int x1 = 2 + 4; // результат 6

int x2 = 10 - 6; //результат 4

int x3 = 10 * 6; //результат 60

double x4 = 10.0 / 4.0; //результат 2.5

double x5 = 10.0 % 4.0; //результат 2

int y1 = 5;

int z1 = ++y1; // (префіксний інкремент) результат z1=6;

y1=6

int y2 = 5;

int z2 = y2++; // (постфіксний інкремент) результат z2=5;

y2=6

int y3 = 5;

int z3 = --y3; // (префіксний декремент) z3=4; y3=4

int y4 = 5;

int z4 = y4--; // (постфіксний декремент) z4=5; y4=4

Операції порівняння

a==b

a!=b

a<b

a>b

a<=b

a>=b

Умовна конструкція if / else

```
if (умова) {дія}
else if (умова) {дія}
else {дія}
```

```
int num1 = 8;
int num2 = 6;
if(num1 > num2)
{
   Console.WriteLine("Число {0} больше числа {1}", num1, num2);
}
else if (num1 < num2)
{
   Console.WriteLine("Число {0} меньше числа {1}", num1, num2);
}
else
{
   Console.WriteLine("Число num1 равно числу num2");
}
```

Умовна конструкція switch/case

```
Console.WriteLine("Нажмите Y или N");
string selection = Console.ReadLine();
switch (selection)
{
   case "Y":
      Console.WriteLine("Вы нажали букву Y");
      break;
   case "N":
      Console.WriteLine("Вы нажали букву N");
      break;
   default:
      Console.WriteLINE("Вы нажали неизвестную букву");
      break;
}
```

Цикл for

```
for ([ініціалізація лічильника]; [умова]; [зміна лічильника])
{
   // дія
}
```

```
for (int i = 0; i < 9; i++)
{
   Console.WriteLine("Квадрат числа {0} дорівнює {1}", i, i * i);
}
```

Цикл foreach

Цикл foreach призначений для перебору елементів в контейнерах.

```
foreach (тип_данних назва змінної in контейнері)
{
   // дія
}
```

Наприклад:

```
int[] array = new int[] { 1, 2, 3, 4, 5 };
foreach (int i in array)
{
   Console.WriteLine(i);
}
```

Цикл while

Цикл while перевіряє істинність деякої умови, і якщо умова істина, то код циклу виконується.

```
int i = 6;
while (i > 0)
{
    Console.WriteLine(i);
    i--;
}
```

Цикл do

В циклі do спочатку виконується код циклу, а потім відбувається перевірка умови в інструкції while.

```
int i = 6;
do
{
   Console.WriteLine(i);
   i--;
}
while (i > 0);
```