Academy .NET

Introduzione alla programmazione

Sommario

- Cos'è la programmazione?
- Il tuo primo programma C #
- Cos'è .NET Framework?
- Cos'è Visual Studio?
- Cos'è MSDN Library?

Cos'è la programmazione?

Programmazione: creazione di una **sequenza di istruzioni** per consentire al computer di eseguire un compito

Fasi della Programmazione

- Definizione di un'attività / problema
- Pianificazione della soluzione
- Ricerca dell'algoritmo adatto per risolverlo
- Ricerca di strutture dati adatte da utilizzare
- Scrittura del codice
- Risoluzione di eventuali errori del programma (bug)
- Rendi felice il tuo cliente

La programmazione orientata agli oggetti

C# fornisce il supporto completo alla **programmazione orientata agli oggetti.**

La progettazione stessa del linguaggio C# è stata basata per l'impiego che segue il paradigma **OOP**.

Questo non significa che C# non possa essere utilizzato anche per la **programmazione procedurale**.

OOP: Aspetti del Paradigma

Nella **OOP** ogni oggetto può inviare e ricevere messaggi da altri oggetti.

Un programma che segue il paradigma OOP non sarà pensato come una sequenza di istruzioni e calcoli da eseguire uno in seguito all'altro, ma un **insieme di oggetti** che **partecipano** e **collaborano** per realizzare un compito.

La programmazione procedurale

La programmazione procedurale è caratterizzata da un insieme di blocchi monolitici contenenti **le intere funzionalità:**

Svantaggi:

- Difficoltà di manutenzione
- Necessario conoscere l'intero programma per applicare modifiche
- Impossibilità nel suddividere il codice

OOP: Vantaggi

- Suddivisione del codice sorgente
- Incapsulamento funzionalità in moduli
- Ogni modulo può essere programmato da sviluppatori diversi

Isolamento e incapsulamento permettono la manutenzione del codice in modo semplificato, separando le funzionalità degli oggetti e l'accesso agli stessi dal codice esterno.

Primo sguardo a C#


```
using System;

class HelloCSharp
{
    static void Main()
    {
        Console.WriteLine("Hello, C#");
    }
}
```

Primo sguardo a C#

Include il namespace standard **«System»**

Definisce la classe chiamata **«HelloCSharp»**

```
using System;

class HelloCSharp
{
    static void Main()
    {
        Console.WriteLine("Hello, C#");
    }
}
```

Definisce il metodo **Main()** – entrypoint del programma

Stampa in console una stringa chiamando il metodo **WriteLine** della classe **Console**

Primo sguardo a C#

Il nome della classe, per convenzione, utilizza il **PascalCase**

```
using System;

class HelloCSharp
{
    static void Main()
    {
        Console.WriteLine("Hello, C#");
    }
}
```

Il simbolo **{** su una nuova riga

Il simbolo **}** su una nuova riga

Indentazione **statement**

Primo sguardo a C#

Linguaggio di programmazione:

Una **sintassi** che permette di dare istruzioni al computer

Funzionalità C #:

Nuovo linguaggio all'avanguardia

Estremamente potente

Facile da imparare

Facile da leggere e capire

Orientato agli oggetti

Di cosa si ha bisogno per programmare?

- Conoscenza di un linguaggio di programmazione
  C #
- Compito da risolvere
- Ambiente di sviluppo, compilatori, SDK
  Visual Studio, .NET Framework SDK
- Insieme di classi standard
  Microsoft .NET Framework FCL
- Documentazione
  MSDN Library

Struttura .NET Framework

C#	C++	VB.NET	J#	F#	PHP	Perl	Delphi	…
ASP.NET *Web Forms, MVC,* *Web API, SignalR*			Windows Forms		WPF / XAML		WinJS / Win8	
WCF and WWF (Communication and Workflow Tier)								
ADO.NET, EF, LINQ and XML (Data Tier)								
Base Class Library (BCL)								
Common Language Runtime (CLR)								
Operating System (OS)								

Common Language Runtime - CLR

- Ambiente di esecuzione gestito
  - Esegue applicazioni .NET
  - Controlla il processo di esecuzione
- Gestione automatica della memoria (**garbage collection**)
- Integrazione dei linguaggi di programmazione
- Supporto di più versioni per gli assembly
- Type safety e security integrati

Framework Class Library

La Libreria di classi Framework (**FCL**) fornisce funzionalità di base agli sviluppatori:

Applicazioni console
Applicazioni multimediali WPF e Silverlight
Applicazioni GUI Windows Forms
Applicazioni Web (siti Web dinamici)
Servizi Web, comunicazione e flusso di lavoro
Server e applicazioni desktop
Applicazioni per dispositivi mobili

Visual Studio

Visual Studio - Ambiente di sviluppo integrato (IDE)

Strumento di sviluppo che aiuta a:

- Scrivere codice
- Progettare l'interfaccia utente
- Compilare il codice
- Eseguire / testare / eseguire il debug delle applicazioni
- Gestisce i file del progetto

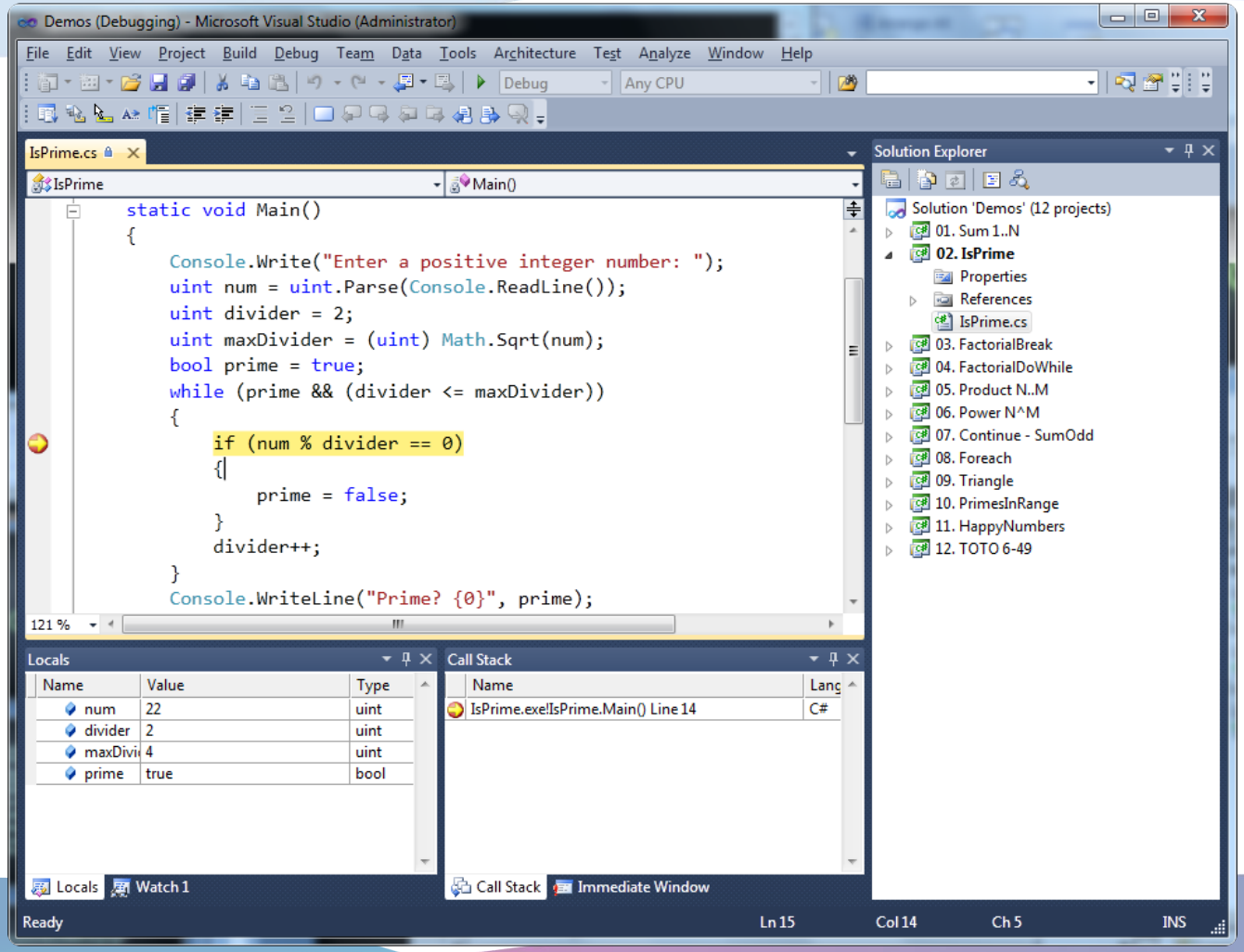
Visual Studio

- Strumento unico per:
  - Scrittura di codice in diversi linguaggi (C #, VB, ...)
  - Utilizzare diverse tecnologie (Web, WPF, ...)
  - Per diverse piattaforme (.NET CF, Silverlight, ...)

- **Integrazione completa** della maggior parte delle attività di sviluppo (**codifica**, **compilazione**, **test**, **debug**, **distribuzione**, **controllo della versione**, ...)

- Facile da usare!

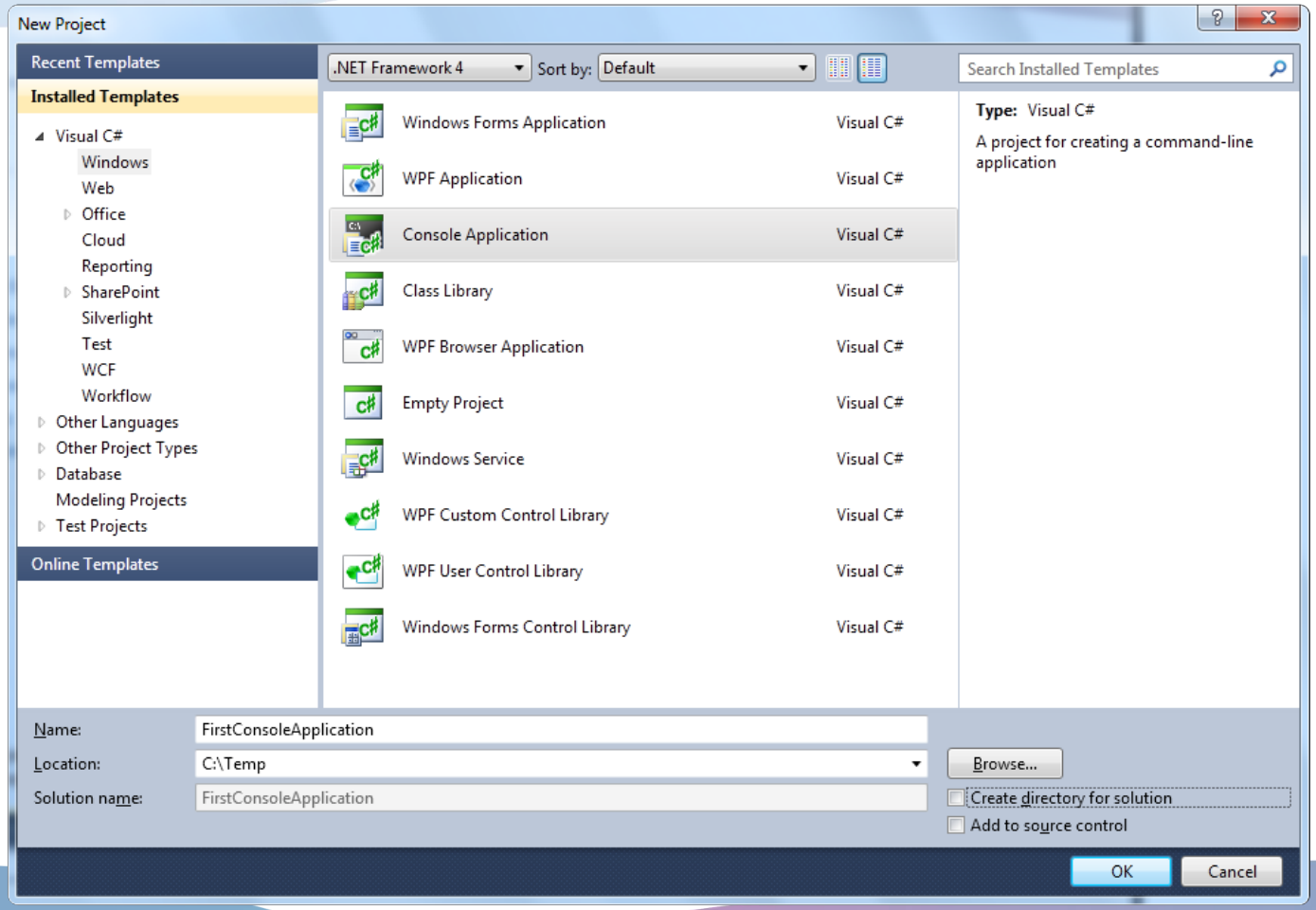
Visual Studio - Esempio



Creazione di una nuova applicazione Console

- File -> New -> Project ...
- -> **Console Application**
- -> **directory** e **nome progetto**

Creare una nuova applicazione Console



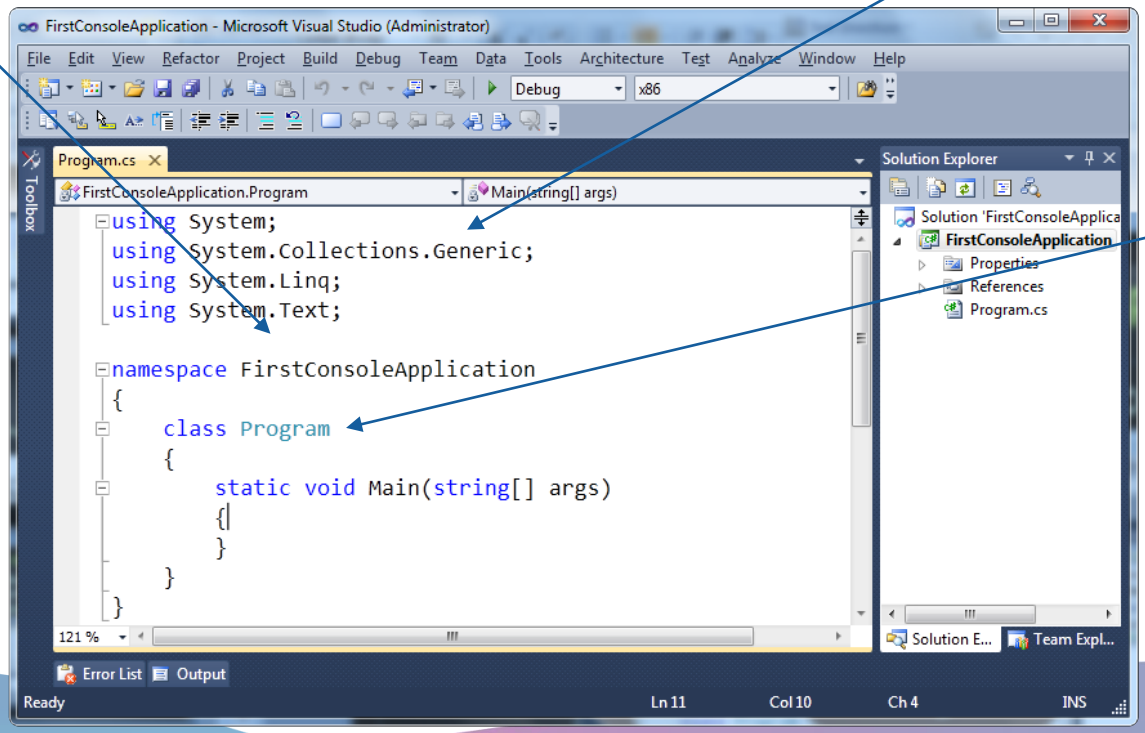
Creare una nuova applicazione Console

codice sorgente di base:

Namespace non obbligatorio

Evita import non necessari

Il nome della classe può essere cambiato



Compilazione del codice sorgente

Il processo di compilazione include:

- Controlli sintattici
- Controlli **Type Safety**
- Traduzione del codice sorgente in linguaggio di basso livello (**MSIL**)
- Creazione di file eseguibili (**assemblies**)

É possibile iniziare la compilazione da:

- Build-> Build Solution / Project
- Premendo [Ctrl + B] o [Maiusc + Ctrl + B]

Avviare un programma

Il processo di esecuzione dell'applicazione include:

- Compilazione (se progetto non compilato)
- Avvio dell'applicazione

Per eseguire l'applicazione:

- dal menu Debug-> Start
- Premendo [**F5**] o [**Ctrl + F5**] (senza debug)

* NOTA: non tutti i tipi di progetto possono essere avviati

Debug del codice

Il processo di **debug** dell'applicazione include:

- Individuazione di un errore
- Trovare le righe di codice che causano l'errore
- Correggere il codice
- Test per verificare se l'errore è scomparso e non ne vengono introdotti di nuovi

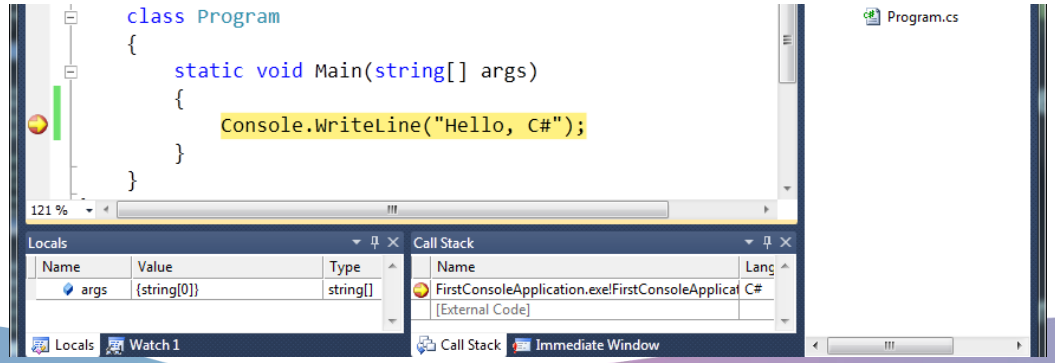
É un processo **iterativo** e **continuo**.

Debug in Visual Studio

Visual Studio dispone di un **debugger integrato.**

Fornisce:

- Punti di interruzione
- Capacità di tracciare l'esecuzione del codice
- Capacità di ispezionare le variabili in fase di esecuzione

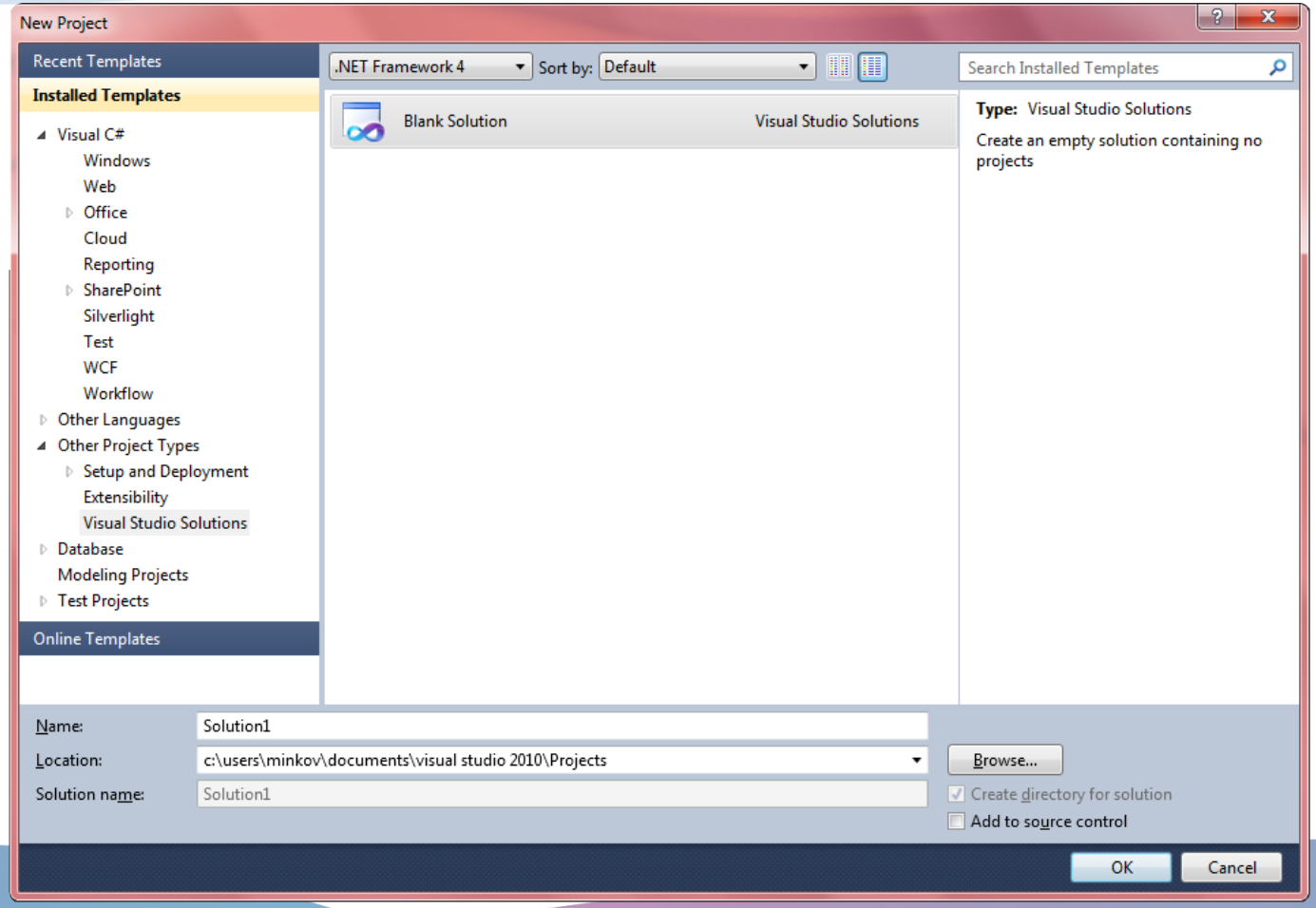


Visual Studio Soluzione Vuota

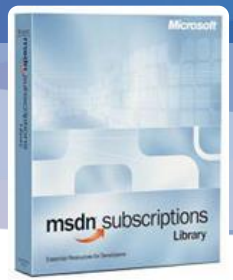
Soluzione vuota di Visual Studio:

- Soluzione senza progetti in essa
  - Progetto da aggiungere in seguito

Visual Studio Soluzione Vuota



Cosa è MSDN Library?



- Documentazione completa di tutte le classi e delle loro funzionalità
  - Con le descrizioni di tutti i metodi, proprietà, eventi, ecc.
  - Con esempi di codice

- Articoli Correlati
- Esempi di Codice
- Utilizzare la copia locale o la versione Web su Developer tools, technical documentation and coding examples | Microsoft Docs

Domande & approfondimenti

Academy .NET