Guida alla Programmazione in C#

1. Cos'è C#?

C# è un linguaggio di programmazione orientato agli oggetti, sviluppato da Microsoft, ideale per lo sviluppo di applicazioni desktop, web e mobile, in particolare all'interno dell'ecosistema .NET. È progettato per essere semplice, moderno e robusto.

2. Installazione e Ambiente di Sviluppo

Per iniziare a programmare in C#, è necessario:

- **Installare .NET SDK**: Il .NET SDK contiene gli strumenti per sviluppare applicazioni .NET. Può essere scaricato dal sito ufficiale di Microsoft.
- **Ambiente di sviluppo**: Visual Studio è l'IDE più utilizzato per C#, ma esistono alternative come Visual Studio Code, Rider, o semplici editor di testo.

3. Struttura di un Programma C#

Ecco un esempio della struttura base di un programma in C#:

csharp
Copia codice

```
using System;

namespace PrimoProgramma
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

- **Namespace**: Rappresenta uno spazio per organizzare le classi. In questo caso, `PrimoProgramma`.
- **Classe `Program`**: Il punto di partenza del programma è la classe `Program`.
- **Metodo `Main`**: È il metodo principale, punto di ingresso del programma.
- **Console.WriteLine()**: Stampa un messaggio a schermo.

4. Tipi di Dati e Variabili

C# supporta vari tipi di dati:

- **Numerici**: `int`, `double`, `float`, `decimal`, `long`
- **Booleano**: `bool` (valori `true` o `false`)
- **Caratteri**: `char`
- **Stringhe**: `string`
- **Altri**: `DateTime`, `object` (tipo base per tutti gli oggetti)

Esempio:

csharp
Copia codice
```
int numero = 10;
double decimale = 5.5;
bool verità = true;
char carattere = 'A';
string testo = "Ciao, C#";
```

5. Operatori

Gli operatori principali in C# includono:

- **Aritmetici**: `+`, `-`, `*`, `/`, `%`
- **Assegnazione**: `=`, `+=`, `-=`, `*=`, `/=`
- **Confronto**: `==`, `!=`, `>`, `<`, `>=`, `<=`
- **Logici**: `&&`, `||`, `!`

Esempio:

csharp
Copia codice
```
int a = 10;
int b = 5;
int somma = a + b;
bool maggiore = a > b;
```

6. Strutture di Controllo

Le strutture di controllo dirigono il flusso del programma.

Condizionali: `if`, `else if`, `else`
csharp
Copia codice
```
if (a > b)
```

```
{
    Console.WriteLine("a è maggiore di b");
}
else
{
    Console.WriteLine("a non è maggiore di b");
}
```

-

Ciclo `for`:
csharp
Copia codice
```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}
```

-

Ciclo `while` e `do-while`:
csharp
Copia codice
```
int i = 0;
while (i < 10)
{
    Console.WriteLine(i);
    i++;
}
```

-

7. Metodi

I metodi organizzano il codice in blocchi riutilizzabili.

Esempio di metodo:

csharp
Copia codice
```
public class Calcolatrice
{
    public int Somma(int a, int b)
    {
        return a + b;
```

```
    }

    public static void Main(string[] args)
    {
        Calcolatrice calc = new Calcolatrice();
        Console.WriteLine(calc.Somma(5, 3));
    }
}
```

8. Programmazione Orientata agli Oggetti (OOP)

C# è un linguaggio orientato agli oggetti (OOP), basato su:

- **Classi e Oggetti**: Le classi definiscono il comportamento degli oggetti.
- **Ereditarietà**: Una classe può ereditare le proprietà di un'altra classe.
- **Incapsulamento**: Consiste nel nascondere i dettagli interni di una classe.
- **Polimorfismo**: Consente di trattare oggetti di classi diverse in modo uniforme.
- **Astrazione**: Permette di rappresentare solo i dettagli essenziali.

9. Esempio di Classe con Incapsulamento

csharp
Copia codice

```
public class Persona
{
    private string nome;
    private int età;

    public Persona(string nome, int età)
    {
        this.nome = nome;
        this.età = età;
    }

    public string GetNome()
    {
        return nome;
    }

    public int GetEtà()
    {
        return età;
    }
```

```
    public void SetNome(string nome)
    {
        this.nome = nome;
    }

    public void SetEtà(int età)
    {
        this.età = età;
    }
}
```

10. Gestione delle Eccezioni

La gestione delle eccezioni è importante per evitare crash dovuti a errori runtime.

Esempio di blocco `try-catch`:

csharp
Copia codice

```
try
{
    int divisione = 10 / 0; // genera un'eccezione
}
catch (DivideByZeroException e)
{
    Console.WriteLine("Errore: divisione per zero");
}
```

11. Input/Output di Base

C# consente di leggere input dall'utente con `Console.ReadLine()` e di stampare output con `Console.WriteLine()`.

csharp
Copia codice

```
using System;

public class InputExample
{
    public static void Main(string[] args)
    {
```

```
        Console.Write("Inserisci il tuo nome: ");
        string nome = Console.ReadLine();
        Console.WriteLine("Ciao, " + nome);
    }
}
```

12. Array e Liste

Array: Un array è una struttura di dati a dimensione fissa.
csharp
Copia codice
```
int[] numeri = {1, 2, 3, 4, 5};
```

-

Liste (dalla libreria `System.Collections.Generic`): una lista ridimensionabile.
csharp
Copia codice
```
using System.Collections.Generic;

List<string> lista = new List<string>();
lista.Add("Elemento 1");
lista.Add("Elemento 2");
```

-

13. Compilare ed Eseguire un Programma C#

- **Compilazione e Esecuzione**: Salva il file come `.cs` (es. `Program.cs`). Da terminale, puoi compilare con `dotnet build` e eseguire con `dotnet run`.

14. Conclusione

Questa guida rappresenta una base introduttiva alla programmazione in C#. Man mano che avanzi, potrai esplorare concetti avanzati come le librerie .NET, l'accesso a database, la creazione di interfacce grafiche e molto altro. Buon lavoro!

4o