Introduzione

Imparare un linguaggio di programmazione e nello specifico C#, è come compiere viaggio in una terra lontana e sconosciuta; un viaggio meraviglioso, affascinante e gratificante, ricco di sorprese e scoperte, ma anche un viaggio non semplice, che richiede perciò tanta pazienza e il giusto tempo. In ogni caso, al termine del viaggio, la ricompensa sarà elevata: si sarà infatti appreso un linguaggio di programmazione, potente e flessibile come C#, che permetterà di utilizzare API di qualsiasi tipo, da quelle più *tediose*, per interfacciarsi a un database, a quelle più *divertenti*, per scrivere videogiochi, e software di qualsiasi tipo. C#, come avremo modo di verificare durante tutto il percorso di apprendimento che il libro intende offrire, è un linguaggio di programmazione estremamente espressivo e ricco di molteplici costrutti sintattici, che dà al programmatore una grande libertà operativa, il cui unico limite sarà la sua fantasia o la sua scarsa preparazione sulle regole sintattiche dei costrutti o sulla semantica delle operazioni del linguaggio.
Desideriamo, inoltre, spendere anche qualche parola sui principi ispiratori che hanno guidato l'autore nella scrittura del testo.

- Gli argomenti propri di ogni capitolo hanno un preciso e chiaro ordine. Ogni capitolo esprimerà compiutamente il relativo obiettivo didattico e non si sovrapporrà o comprenderà contenuti di altri capitoli. Per esempio, se nel Capitolo 2 si parlerà delle variabili, solo nel successivo Capitolo 3 si parlerà degli array. Quanto detto può apparire abbastanza ovvio ma non lo è. Infatti, oggi, si sta assistendo alla proliferazione di testi con argomenti scritti "a spirale" dove cioè un argomento può contenere riferimenti iniziali ad altri argomenti i quali saranno poi trattati approfonditamente solo nel capitolo di pertinenza. Questo, a parere dell'autore, laddove non strettamente necessario e comunque non legato ai costrutti del linguaggio (è evidente che per mostrare il valore di un variabile bisogna dire qualcosa di propedeutico sul metodo `Console.WriteLine`) induce a distrazioni e fa perdere inutile tempo per la corretta comprensione della corrente unità didattica.
- Il modo espositivo seguito è rigoroso, laddove necessario piuttosto formale, e si è prestata molta attenzione al corretto uso della terminologia propria di C#. Questo non è un libro del tipo "impariamo C# in 24 ore" oppure "C# *per tutti*". C# è un linguaggio complesso e ricco di costrutti; per insegnarlo, ci vuole serietà e il giusto rigore; per impararlo, ci vuole pazienza e disciplina.

- I listati e gli *snippet* (frammenti) di codice sono stati concepiti in modo da dare una chiara indicazione pratica dei relativi argomenti teorici: sono piuttosto brevi, autoconclusivi e decorati da ulteriori commenti che danno, talune volte, ulteriori spiegazione teoriche.
- Non sono trattate API specifiche del .NET Framework, ma solo quelle, diciamo, "connesse" con il linguaggio. Questo è un libro su C# ossia sul linguaggio di programmazione, non un libro sulla programmazione di Windows. Ha senso spendere centinaia di pagine per parlare della programmazione di GUI, di database, del Web, e così via, elencando di fatto una pletora di API? Ha senso sottrarre centinaia di pagine a ulteriori dettagli sui costrutti del linguaggio per "donarle" a quelle API? Crediamo di no. Crediamo che un libro su C# sia un libro su C#, che dedica centinaia di pagine a spiegare in modo compiuto, dettagliato e rigoroso cosa sia la OOP, la programmazione funzionale e via discorrendo per gli altri argomenti pertinenti. In definitiva, crediamo che un testo così strutturato dia al lettore, paziente e volenteroso, una marcia in più per poter affrontare con la giusta comprensione e consapevolezza le API che avrà interesse apprendere e che, nel caso di Microsoft, godono di un'abbondante documentazione (anche in italiano).

Organizzazione del libro

Il libro è organizzato idealmente nelle seguenti parti.

- *Prima parte – Concetti e costrutti fondamentali*. È costituita dai seguenti capitoli: *Capitolo 1 – Introduzione al linguaggio*; *Capitolo 2 – Variabili, costanti, letterali e tipi*; *Capitolo 3 – Array*; *Capitolo 4 – Operatori*; *Capitolo 5 – Strutture di controllo*; *Capitolo 6 – Metodi*. Questa parte enuclea le nozioni essenziali e fondamentali che sono propedeutiche di un qualsiasi corso sulla programmazione; dal concetto di variabile e array, passando per gli operatori e le strutture di controllo del flusso di esecuzione del codice, per concludere con un dettaglio sul modo in cui scrivere blocchi di codice attraverso il costrutto di *metodo*.

  NOTA

  Nell'organizzazione dei capitoli si è preferito introdurre i metodi prima del costrutto di classe poiché si tratta di "strutture" sintattiche più semplici e in modo da affrontare i concetti essenziali del linguaggio in ordine crescente di complessità. Filosoficamente, questa scelta si sposa bene con l'inquadramento del paradigma a oggetti (basato sulle classi) inteso come estensione del paradigma procedurale (basato sulle procedure). Non a caso, le strutture di controllo della programmazione a oggetti, che vengono poi utilizzate all'interno dei metodi, sono le stesse del paradigma procedurale.

- *Seconda parte – Paradigmi, stili di programmazione e gestione degli errori*. È costituita dai seguenti capitoli: *Capitolo 7 – Programmazione basata sugli oggetti*; *Capitolo 8 – Programmazione orientata agli oggetti*; *Capitolo 9 – Programmazione generica*; *Capitolo 10 – Programmazione funzionale*; *Capitolo 11 – Eccezioni software*. Questa parte illustra come avvantaggiarsi, nella costruzione di sistemi software, dei più noti paradigmi di programmazione; da quello maggiormente utilizzato, proprio della *OOP*, a quello,

definito *funzionale*, che oggi sta riscuotendo una profonda rivalutazione e un certo successo. Analizza anche come impiegare nei programmi uno stile di programmazione definito *generico*, che fa uso di tipi e funzionalità che sono in grado di compiere una medesima operazione su un insieme di tipi di dato differenti. Spiega, infine, come intercettare e gestire adeguatamente gli errori software, grazie all'utilizzo del meccanismo di *gestione delle eccezioni*.

- *Terza parte – Concetti e costrutti supplementari e avanzati*. È costituita dai seguenti capitoli: *Capitolo 12 – Namespace*; *Capitolo 13 – Assembly*; *Capitolo 14 – Attributi*; *Capitolo 15 – Documentazione del codice sorgente*; *Capitolo 16 – Direttive di preprocessing*; *Capitolo 17 – Programmazione unsafe e puntatori*; *Capitolo 18 – Programmazione dinamica e reflection*. Questa parte tratta i seguenti argomenti. Come organizzare, tramite gli *spazi dei nomi*, in modo raggruppato, strutturato e gerarchico, dei tipi che siano connessi a una particolare funzionalità applicativa. Come è strutturato un *assembly* ossia un file `.exe` o `.dll`. Come associare, tramite dei *metadati* (attributi), delle informazioni descrittive agli elementi di un programma. Come far elaborare al compilatore, durante l'esecuzione dell'analisi lessicale, dei particolari "comandi" di *preprocessing*. Come utilizzare un contesto non sicuro (*unsafe context*) per scrivere codice non sicuro (*unsafe code*) che si avvale della potenza dei tipi *puntatore*. Come usare i costrutti sintattici e semantici propri di una *tipizzazione dinamica* e come esaminare, a runtime e tramite la *riflessione*, i tipi di un programma; come poi utilizzarne le funzionalità e ciò indipendentemente dal fatto che quei tipi siano disponibili a *compile time*.
- *Quarta parte – Introduzione ai tipi e alle librerie essenziali*. È costituita dai seguenti capitoli: *Capitolo 19 – Stringhe*; *Capitolo 20 – Espressioni di query*; *Capitolo 21 – Collezioni*; *Capitolo 22 – Multithreading*. Questa parte analizza le stringhe, LINQ, le collezioni di dati e la programmazione concorrente;
- *Appendici*. L'*Appendice A – Ambienti di sviluppo integrato* e l'*Appendice B – Sistemi numerici: cenni* mostrano come usare gli IDE Visual Studio e MonoDevelop e fornisce un'introduzione ai sistemi numerici decimale, ottale, esadecimale e binario.

Struttura del libro e convenzioni

Gli argomenti del libro sono organizzati in capitoli. Ogni capitolo è numerato in ordine progressivo e denominato significativamente nel suo obiettivo didattico. I capitoli sono poi suddivisi in paragrafi, al cui interno possiamo avere dei blocchi di *testo* o di *grafica*, a supporto alla teoria, denominati in accordo con il seguente *pattern*.

- *`Tipologia`* `NrCapitolo.NrProgressivo Descrizione`. Dove *`Tipologia`* può esprime: un listato di codice sorgente (`Listato`); un frammento di codice sorgente (`Snippet`); la sintassi di un costrutto C# (`Sintassi`); dei comandi di shell (`Shell`); l'output di un programma (`Output`); una figura (`Figura`); una tabella (`Tabella`). Per esempio, il blocco denominato "Listato 6.2 ByValue.cs (ByValue).", indica il secondo listato di codice del Capitolo 6, avente come descrizione il nome del file `.cs` di codice sorgente e, tra parentesi, il nome del progetto dell'IDE.

Per quanto attiene ai listati, abbiamo adottato anche la seguente convenzione: i puntini di sospensione (...) eventualmente presenti indicano che in quel punto sono state omesse alcune parti del listato, presenti però nei relativi file `.cs` allegati al libro.

Gli stessi puntini di sospensione possono talvolta trovarsi anche negli output di un programma eccessivamente lungo.

Codice sorgente e progetti

All'indirizzo `http://www.apogeonline.com/libri/9788850333691/scheda` è possibile scaricare un archivio ZIP contenente il codice degli esempi del libro e organizzato come segue:

- `[Windows | Linux | MacOS] -> CapNr -> [Listati | Snippet | Sorgenti]`. Avremo, cioè, tre differenti cartelle per ciascun sistema operativo, denominate `Windows`, `Linux` o `MacOS` e che conterranno, ognuna, le cartelle dei capitoli (`Cap01`, `Cap02` e così via). Queste ultime cartelle conterranno, a loro volta, le cartelle dei progetti dei listati (`Listati`) o degli snippet di codice (`Snippet`) per gli IDE Visual Studio o MonoDevelop; la cartella dei sorgenti (`Sorgenti`) conterrà, invece, i sorgenti `.cs` per chi non desiderasse usare uno degli IDE menzionati.

Compilare ed eseguire direttamente i listati e gli snippet di codice

Per agevolare la compilazione e l'esecuzione dei listati e degli snippet di codice, indipendentemente dall'IDE Visual Studio o MonoDevelop, diamo il seguente consiglio.

- Per Windows, creare in `C:` le cartelle `MY_C#_SOURCES`, `MY_C#_EXE`, `MY_C#_LIBRARIES` e `MY_C#_FILES`.
- Per GNU/Linux o OS X, creare in `$HOME` le directory `MY_C#_SOURCES`, `MY_C#_EXE`, `MY_C#_LIBRARIES` e `MY_C#_FILES`.

Compilare ed eseguire con gli IDE i listati e gli snippet di codice

Se lo si desidera, è possibile utilizzare l'IDE Visual Studio o MonoDevelop per editare, compilare, eseguire ed effettuare il debugging del codice sorgente presente nel libro. A tal fine consultare l'*Appendice A – Ambienti di sviluppo integrato* per una spiegazione in merito all'utilizzo introduttivo di tali IDE e alla creazione e all'impiego dei progetti.