Prefazione

Imparare il linguaggio C è come fare viaggio in una terra lontana e sconosciuta; è dunque sia un viaggio meraviglioso, affascinante e gratificante, ricco di sorprese e scoperte, sia un viaggio non semplice, che richiede perciò tanta pazienza e il giusto tempo.
In ogni caso, al termine del viaggio, la ricompensa ricevuta sarà elevata: si sarà appreso non un linguaggio di programmazione qualsiasi ma *il* linguaggio di programmazione per eccellenza che già antecedentemente alla sua prima standardizzazione, avvenuta nel lontano 1989 (ANSI C), aveva iniziato a entusiasmare una vasta platea di programmatori.
Come avremo modo di verificare durante tutto il percorso di apprendimento che il libro intende offrire, C è un linguaggio estremamente espressivo e sintetico che dà grande "fiducia" e libertà operativa al programmatore, il cui unico limite potrà essere dato solo dalla poca fantasia o dalla scarsa preparazione sulle regole sintattiche dei costrutti o sulla semantica delle operazioni.
Dal punto di vista più pratico, e forse meno filosofico, imparare C consente di sviluppare programmi di uso generale, davvero a 360 gradi, programmi cioè utili per qualsiasi ambito applicativo (sistemi operativi, robotica, database, networking, grafica e così via). Esso quindi si dimostra, ancora oggi, dove è presente una pletora di altri linguaggi di programmazione che promettono di essere più *easy* e più *safe*, lo strumento di eccellenza adoperato da milioni di programmatori che apre le porte del mondo della programmazione *reale*, sicuramente più *hard* ma anche più appagante.
Last but not least, C è un impressionante e imprescindibile strumento didattico usato soprattutto dalle università più giudiziose per insegnare sia i fondamenti della programmazione ma anche come è fatto, a "basso livello", un calcolatore elettronico (si pensi ai puntatori, la cui disamina non può prescindere da una spiegazione approfondita di cos'è e come è organizzata la memoria di un elaboratore).
Desideriamo, inoltre, spendere qualche parola sui principi ispiratori che hanno guidato l'autore nella scrittura del presente 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; allo stesso modo solo dopo aver trattato anche delle funzioni nel Capitolo 6, si parlerà dei puntatori nel Capitolo 7 e di tutte le loro *relazioni* e utilizzi con le variabili, gli array e le funzioni. 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 sull'istruzione `printf`), induce a distrazioni e fa perdere inutilmente tempo nella 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 del linguaggio. Questo non è un libro del tipo "Impariamo C in 24 ore" oppure "C For Dummies". I libri che pretendono di insegnare C in quel modo molto probabilmente sono solo uno specchio per le allodole; illudono, dando false promesse. C è un linguaggio complesso e ricco di sfumature; per insegnarlo, ci vogliono "serietà" e il giusto rigore; per impararlo, ci vogliono pazienza e disciplina.
- I listati e gli snippet di codice sono stati pensati in modo che diano una chiara indicazione pratica dei relativi argomenti teorici; sono piuttosto brevi, autoconclusivi e supportati da ulteriori commenti che danno, talune volte, ulteriori spiegazione teoriche.

Organizzazione del libro

Il libro è organizzato nei capitoli elencati di seguito.

- Capitolo 1, "Introduzione al linguaggio C": introduciamo il lettore ad alcuni concetti propedeutici del calcolatore elettronico e dello sviluppo di un programma in C. Redigiamo anche un primo programma e mostriamo come compilarlo ed eseguirlo.
- Capitolo 2, "Variabili, costanti, letterali e tipi": parliamo delle variabili, delle costanti e dei tipi di dato fondamentali del linguaggio. Chiudiamo con una trattazione delle conversioni di tipo e di un confronto tra `typedef` e `#define`.
- Capitolo 3, "Array": analizziamo l'importante struttura dati array nella sua forma monodimensionale e multidimensionale. Parliamo altresì degli array di lunghezza variabile, degli array costanti e dell'applicazione dell'operatore `sizeof` per ottenere la dimensione di un array.
- Capitolo 4, "Operatori": mostriamo tutti gli operatori che il linguaggio mette a disposizione per manipolare i dati; dai semplici operatori aritmetici a quelli complessi *bitwise*. Chiude un'utile tabella di precedenza degli operatori.
- Capitolo 5, "Strutture di controllo": vediamo come gestire il flusso di esecuzione di un programma attraverso le istruzioni di selezione, di iterazione e di salto.
- Capitolo 6, "Funzioni": trattiamo del fondamentale costrutto di funzione. Vediamo come essa si dichiara e si definisce una funzione, cosa sono i parametri formali, come si invoca e cosa sono i parametri attuali o argomenti. Analizziamo in dettaglio l'importante istruzione `return` e cos'è la ricorsione.
- Capitolo 7, "Puntatori": parliamo in grande dettaglio dei puntatori e della loro relazione con gli array. Vediamo anche cosa sono i puntatori a puntatori, i puntatori

a funzione, i puntatori a `void` e i puntatori nulli. Infine illustriamo in che modo le keyword `const` e `restrict` influenzino un puntatore e la conversione tra puntatori.

- Capitolo 8, "Strutture, unioni ed enumerazioni": descriviamo le strutture, le unioni e le enumerazioni evidenziando le loro differenze e quali sono i comuni casi di utilizzo.
- Capitolo 9, "Dichiarazioni": approfondiamo i concetti legati alle variabili, ai blocchi, allo scope e al linkage. Diamo uno sguardo di insieme agli specificatori della classe di memorizzazione, ai qualificatori di tipo, agli specificatori di tipo, agli specificatori di funzione e agli specificatori di allineamento.
- Capitolo 10, "Il preprocessore": parliamo di una delle peculiarità di C, ossia il suo preprocessore, indicando tutte le direttive che mette a disposizione, dalla più comune `#define` a quella più *esoterica* `#pragma`.
- Capitolo 11, "La libreria standard": analizziamo tutte le funzionalità della libreria standard del linguaggio attraverso un excursus completo di tutti i suoi header, anche qui da quelli più usati come `<string.h>`, `<stdio.h>` e `<stdlib.h>` a quelli più particolari come `<fenv.h>`, `<locale.h>` e `<signal.h>`.
- Appendice A, "Installazione e utilizzo di GCC": mostriamo come installare la suite di compilazione GCC e come effettuare le comuni operazioni di compilazione, linking e dubugging del codice.
- Appendice B, "Installazione e utilizzo di NetBeans": trattiamo di NetBeans, un potente IDE multipiattaforma che consente di creare, compilare ed eseguire codice C in un ambiente di facile utilizzo e ricco di funzionalità.
- Appendice C, "Sistemi numerici: cenni": forniamo un'introduzione ai sistemi numerici decimale, ottale, esadecimale e binario e a come eseguire le conversioni tra di essi. Mostriamo anche come effettuare le operazioni aritmetiche binarie di addizione, sottrazione, moltiplicazione e divisione.

Struttura del libro e convenzioni

Gli argomenti del libro sono, ovviamente, organizzati in capitoli. Ogni capitolo è numerato in ordine progressivo e denominato significativamente nel suo obiettivo didattico (per esempio, Capitolo 2, "Variabili, costanti, letterali e tipi"). I capitoli sono poi suddivisi in paragrafi di pertinenza.
All'interno dei paragrafi possiamo avere dei blocchi di testo o di grafica, a supporto alla teoria, denominati, per esempio, come segue:

- *Listato NrCapitolo.NrProgressivo Descrizione…* per i listati del codice sorgente;
- *Snippet NrCapitolo.NrProgressivo Descrizione…* per un frammento di codice sorgente;
- *Sintassi NrCapitolo.NrProgressivo Descrizione…* per la sintassi di un costrutto del linguaggio;
- *Shell NrCapitolo.NrProgressivo Descrizione…* per un comando di shell;
- *Output NrCapitolo.NrProgressivo Descrizione…* per l'output di un programma;

- *Figura NrCapitolo.NrProgressivo Descrizione…* per una figura;
- *Tabella NrCapitolo.NrProgressivo Descrizione…* per una tabella.

Per esempio, il blocco denominato "Listato 6.10 VariableArgumentsList.c (VariableArgumentsList)" indica il listato di codice numero 10 del Capitolo 6 avente come descrizione il nome del file `.c` di codice sorgente e tra parentesi il nome del progetto NetBeans.
Per quanto attiene ai listati, abbiamo adottato la seguente convenzione: i puntini di sospensione (…) eventualmente presenti indicano che in quel punto sono state omesse alcune parti del listato. Ovviamente, le medesime parti sono presenti nei relativi file `.c` allegati al libro. Gli stessi caratteri possono talvolta trovarsi anche negli output di un programma eccessivamente lungo.

Codice sorgente e progetti

All'indirizzo `http://www.apogeonline.com/libri/9788850333288/scheda` è possibile scaricare un archivio ZIP che contiene tante cartelle quanti sono i capitoli del libro. Ciascuna cartella, denominata `Cap01`, `Cap02` e così via, ha le seguenti cartelle strutturate e denominate come segue:

- `Listati`
  - ◊ `CON_NetBeans`
    - – `Linux`
      - • *[nome cartella del progetto; es. PrimoProgramma]*
      - • …
    - – `Windows`
      - • *[nome cartella del progetto; es. PrimoProgramma]*
      - • …
  - ◊ `SENZA_NetBeans`
    - • *[nome file .c del listato; es. PrimoProgramma.c]*
    - • …
- `Snippet`
  - ◊ `CON_NetBeans`
    - – `Linux,`
      - • *[nome cartella del progetto; es. 2.1]*
      - • …
    - – `Windows`
      - • *[nome cartella del progetto; es. 2.1]*
      - • …
  - ◊ `SENZA_NetBeans`
    - • *[nome file .c dello snippet; es. 2.1.c]*
    - • …

All'interno, dunque, della cartella `SENZA_NetBeans` sono presenti, in modo indipendente, tutti i file sorgente `.c` del capitolo di pertinenza con le eventuali risorse complementari; nella cartella `CON_NetBeans` sono invece presenti, sia per Linux sia per Windows, delle sottocartelle direttamente correlate a specifici progetti caricabili con l'IDE NetBeans.

Compilare ed eseguire direttamente i listati e gli snippet di codice

Per rendere agevole la compilazione e l'esecuzione dei listati e gli snippet di codice, indipendentemente dall'IDE NetBeans, riteniamo utili i consigli riportati di seguito.
Se si utilizza Windows, creare le seguenti strutture di directory:

- per i sorgenti, `C:\MY_C_SOURCES`;
- per gli eseguibili, `C:\MY_C_BINARIES`;
- per i file oggetto, `C:\MY_C_OBJECTS`;
- per i file include, `C:\MY_C_INCLUDE`;
- per i file delle librerie statiche, `C:\MY_C_STATIC_LIBRARIES`;
- per i file delle librerie dinamiche, `C:\MY_C_SHARED_LIBRARIES`;
- per altri file, `C:\MY_C_FILES`.

Se si utilizza GNU/Linux, creare le seguenti strutture di directory, dove `$HOME` rappresenta la propria home directory (per esempio `/home/thp`):

- per i sorgenti, `$HOME/MY_C_SOURCES`;
- per gli eseguibili, `$HOME/MY_C_BINARIES`;
- per i file oggetto, `$HOME/MY_C_OBJECTS`;
- per i file include, `$HOME/MY_C_INCLUDE`;
- per i file delle librerie statiche, `$HOME/MY_C_STATIC_LIBRARIES`;
- per i file delle librerie dinamiche, `$HOME/MY_C_SHARED_LIBRARIES`;
- per altri file, `$HOME/MY_C_FILES`.

Leggere altresì e in via preliminare l'Appendice A.

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

Se lo si desidera, è possibile utilizzare l'IDE NetBeans per editare, compilare, eseguire e "debuggare" il codice sorgente presente nel libro. A tal fine consultare l'Appendice B, per una spiegazione in merito all'uso introduttivo di tale IDE e alla creazione e all'impiego dei progetti.

NOTA

I progetti NetBeans degli snippet di codice hanno delle opzioni di compilazione "pedanti" in modo che il lettore possa essere avvisato su cosa riporta il compilatore durante la fase di compilazione. È anche importante avvisare che se da Windows si spostano le cartelle dei progetti NetBeans in GNU/Linux, sarà apposto in automatico su un file denominato

`.dep.inc`, presente nella root di ogni progetto, l'attributo "Sola lettura", il quale darà problemi di compilazione sotto Windows se le stesse cartelle saranno spostate nuovamente su sistemi Microsoft (o se saranno spostate direttamente da GNU/Linux in caso sia avvenuta qui una loro prima copia). Lo stesso problema, comunque, non sarà mai rilevato in GNU/Linux.