Prefazione

Questo libro è in primo luogo una guida graduale e completa all'apprendimento della programmazione strutturata e modulare con il linguaggio C. Il percorso didattico non richiede alcun prerequisito in termini di conoscenze ed esperienze di computer e di programmazione. Proprio grazie a questa impostazione il testo ha riscosso sin dalla prima edizione un vasto e duraturo successo in ambito universitario e professionale. Particolare attenzione è posta sui principi e sulle tecniche di programmazione, il controllo del flusso di esecuzione, la rappresentazione dei dati, la definizione e l'utilizzo di funzioni e librerie, le strutture dati, le operazioni di ingresso e uscita. La prima metà dei capitoli è sufficiente per apprendere le basi del linguaggio e della programmazione strutturata e modulare. Il prosieguo dello studio permette di sviluppare, mettere a punto e verificare programmi di media complessità, acquisendo un importante bagaglio di conoscenze sugli algoritmi classici della letteratura del settore, e rende pronti ad affrontare problemi più articolati e a realizzare software di complessità superiore.

Questo libro è anche un introduzione alla programmazione orientata agli oggetti e al linguaggio Objective-C a cui sono dedicati per intero gli ultimi cinque capitoli. Grande cura è rivolta ai concetti di generalizzazione, classe, oggetto, istanza, ereditarietà e alla loro esemplificazione. Altrettanta attenzione è posta alle espressioni-messaggio proprie del linguaggio, all'impiego delle classi base già disponibili, a costruirne e utilizzarne di nuove, alla comunicazione fra le classi, ai meccanismi di delega e, non ultimo, al riutilizzo di tutto quanto si è già appreso del linguaggio C. L'Objective-C permette di programmare le *apps*, le applicazioni per smarthphone e tablet, per iPhone e iPad e in generale le applicazioni per i computer della Apple, ma non solo. Attraverso il supporto del compilatore open source *gcc* può essere impiegato in ogni contesto, e su i più diversi sistemi operativi quali Windows e Linux, per esempio per lo sviluppo di compilatori, per la programmazione di microprocessori e di sistemi in tempo reale.

È interessante notare come le scelte delle aziende e degli sviluppatori cadano molto spesso, e con rinnovata frequenza, su linguaggi con il "C dentro". Secondo **www.tiobe.com** (Tiobe, novembre 2012), che tiene una classifica ben accreditata con continui aggiornamenti, il linguaggio C è attualmente il più "scelto" al mondo, seguito da Java, Objective-C e C++, con l'Objective-C che si distingue per il maggior tasso di crescita. Tra i primi quattro, tre hanno il "C dentro".

Le novità della quinta edizione

La quinta edizione differisce dalla precedente per tre aspetti fondamentali: il nuovo standard internazionale del linguaggio C, la programmazione orientata agli oggetti e il linguaggio Ojective-C.

Il linguaggio C presentato nel libro è conforme allo standard ISO/IEC 9899:2011, più brevemente C11, che riunisce le numerose revisioni succedutesi negli anni del precedente C99. Sono state introdotte nuove parole chiave e librerie standard, è migliorato per esempio il supporto a *unicode*, alla matematica dei numeri complessi, al trattamento dei *thread*; alcune funzioni sono state abolite e sostituite da altre che migliorano la sicurezza delle applicazioni. Dando conto del nuovo standard si è tuttavia prestato attenzione al fatto che gli esempi di programmazione fossero supportati da implementazioni esistenti e largamente accessibili.

È stato aggiunto il Capitolo 26 che presenta il paradigma degli oggetti, i concetti di classe, istanza, ereditarietà e polimorfismo, mantenendo il consueto approccio diretto, esemplificativo e rigoroso.

I nuovi Capitoli 27-30 costituiscono un'introduzione graduale all'apprendimento del Objective-C, in cui si scoprono le caratteristiche del linguaggio prendendo le mosse da quanto già appreso sulla programmazione orientata agli oggetti e da quanto lungamente sperimentato con il C.

L'impostazione didattica del manuale

Ogni concetto è dapprima presentato con esemplificazioni, poi gradualmente sviluppato e approfondito. L'avvio introduce in modo organico e propedeutico allo studio del C e delle tecniche di programmazione attraverso argomenti di base quali: sistemi di elaborazione, sistemi operativi, algoritmi, linguaggi. Un'appendice è appositamente dedicata alla rappresentazione e al trattamento dell'informazione con una presentazione dei sistemi di numerazione e di codifica. In questo modo anche lo studente che si avvicina per la prima volta alla programmazione ha a portata di mano tutti gli strumenti concettuali che gli sono necessari, senza che debba ricorrere obbligatoriamente ad altre fonti da cui attingere informazioni frammentate e più difficilmente riconducibili a un quadro d'insieme.

Per garantire un approccio facilitato all'apprendimento è stata posta grande attenzione all'ordine di presentazione degli argomenti. Sin dall'inizio il lettore può scrivere e provare sull'elaboratore programmi completi. L'esperienza insegna infatti che uno dei modi migliori per apprendere un linguaggio è introdurre pochi concetti e su quelli concentrarsi, sviluppandoli subito con esempi. I primi problemi di programmazione sono concettualmente semplici, il loro scopo è di prendere confidenza con i costrutti sintattici del linguaggio: istruzioni, dati, operatori, strutture di controllo decisionali e iterative, funzioni di libreria. Successivamente sono presentati gli array e la creazione di proprie funzioni, quindi sono realizzati programmi di ricerca e ordinamento. Particolare cura è posta nella trattazione della programmazione modulare: sottoprogrammi, passaggio di parametri, visibilità

delle variabili. La scomposizione dei programmi in funzioni segue i dettami di modularità, coesione e accoppiamento richiesti dalla progettazione e programmazione strutturata; non si è voluto però inibire l'uso delle variabili globali, usate con parsimonia e cautela. Ai puntatori, notoriamente ostici, è dedicato un intero ampio capitolo in cui vengono introdotti con dovizia di esempi.

In modo graduale si propongono allo studente temi e problemi più complessi che richiedono un'analisi preliminare per valutare soluzioni alternative. Con la ricorsione vengono presentati numerosi interessanti esempi di calcolo combinatorio, gli stimolanti problemi della torre di Hanoi, delle otto regine, il metodo di ordinamento *quicksort*. Vengono poi introdotte le strutture, combinate con array e puntatori, che permettono di affrontare il Caso di studio per la realizzazione di un programma modulare di gestione anagrafica; quindi i file, con cui prende corpo il successivo Caso di studio per una nuova soluzione allo stesso problema, secondo una variante che prevede di archiviare i dati su memoria persistente. Si arriva così a trattare pile, code, alberi e grafi, presentando differenti implementazioni che coinvolgono array, liste lineari, liste multiple e soluzioni miste, con un duplice scopo: da una parte offrire un valido banco di prova per il programmatore C, spingendolo a sfruttare caratteristiche tipiche del linguaggio come puntatori, strutture, funzioni di allocazione di aree memoria e complesse chiamate di funzioni; dall'altra parte costruire un'introduzione completa alle strutture dati, che spesso vengono studiate solo sul piano teorico nei corsi di informatica. Un apposito capitolo è dedicato a riflessioni su semantica e correttezza dei programmi, temi messi a fuoco soffermandosi sul concetto di stato e impiegando i metodi delle asserzioni e delle specifiche per migliorare correttezza e affidabilità.

I Casi di studio costituiscono ampie esemplificazioni che percorrono l'intero testo. Per esempio, il problema della gestione di una sequenza del Caso di studio I consente di approfondire l'uso di funzioni, passaggio di parametri, array, metodi di ricerca e ordinamento. Il Caso di studio II aiuta a prendere confidenza con i puntatori, con il passaggio di parametri per riferimento e ancora con gli array. Infine il Caso di studio V permette di far pratica con l'allocazione dinamica della memoria, le strutture dati quali le liste lineari e ancora con i puntatori.

Il paradigma astratto degli oggetti viene introdotto attraverso esempi concreti usando uno pseudo-codice molto vicino alla sintassi del linguaggio Objective-C. In questo modo si ottengono insieme due risultati: si ha una livello di formalizzazione sufficientemente semplice da permettere una comprensione intuitiva dei nuovi argomenti, quali ereditarietà, istanza, polimorfismo, e si comincia a familiarizzare con i meccanismi del linguaggio. Il successivo approccio al vero codice Objective-C procede per esemplificazioni che inizialmente usufruiscono di alcune tra le più importanti classi di base che il linguaggio mette a disposizione; si lavora con stringhe, array, dizionari, facendo pratica di programmazione a oggetti, poi ci si emancipa costruendo proprie gerarchie di classi e imparando le diverse opzioni disponibili per la gestione della memoria. Nella trattazione si evidenziano i vantaggi ottenuti con questo livello di astrazione, più alto rispetto a quello della programmazione procedurale, seppur strutturata e modulare. D'altra parte si presentano le difficoltà dell'uso della sola ereditarietà nei sistemi complessi e quindi si introducono i meccanismi di comunicazione tra classi attraverso i protocolli formali e informali del linguaggio.

Il testo è strutturato in parti come indicato.

Parte A	Computer, Sistemi operativi, Algoritmi, Programmi.	Esempi in C
Parte B	Programmazione strutturata, Tipi dati, Algoritmi di base, Programmazione modulare.	Studio del linguaggio C
Parte C	Tecniche avanzate e Algoritmi notevoli.	
Parte D	Strutture dati, Progetti di sviluppo software, Uso esteso delle direttive del preprocessore. Semantiche e correttezza dei programmi.	
Parte E	Programmazione orientata agli oggetti: classi, oggetti, ereditarietà, polimorfismo.	Studio del linguaggio Objective-C
Parte F	Messaggi, array, dizionari, classi per composizione, metodi, gestione della memoria, protocolli, delega, categorie, introspezione, eccezioni.	

Contenuti

I primi quattro capitoli introducono i concetti generali teorici e pratici dell'informatica e gli argomenti che stanno alla base della programmazione, riflettendo inoltre sulla possibilità e i limiti dell'elaborazione automatica. L'obiettivo è duplice: approcciare la disciplina partendo da fondamenta solide e condivise e proiettare una luce sullo studio della programmazione in linguaggio C che svolgeremo di seguito, rendendoci più consapevoli nel nostro lavoro e forse più motivati.

In particolare il **Capitolo 1 Computer** presenta l'architettura del computer e la sua logica di funzionamento: unità di elaborazione, memoria centrale e cache, periferiche di Input/Output, memoria di massa. Successivamente partendo da una breve presentazione della storia degli elaboratori iniziata con i primi automatismi e la scansione del tempo, si passa a una possibile classificazione dei tipi di computer, le reti locali, geografiche, Internet, le architetture Client/Server e le tendenze attuali.

L'interazione degli utenti e dei programmi con la macchina passa attraverso il sistema operativo. Se in generale è bene averne coscienza, sapere i compiti che il sistema operativo svolge e avere un'idea di come li svolge è fondamentale per un programmatore C, per gestire la memoria centrale e di massa o per pilotare i dispositivi di Input/Output. Inoltre, non dobbiamo dimenticare che proprio in C sono realizzati la maggior parte degli stessi sistemi operativi e del software di base in generale.

Il **Capitolo 2 Sistemi operativi** presenta la gestione delle risorse hardware e software, l'interfaccia con l'utente, la schedulazione dei processi, lo swapping delle pagine, la gestione del file system, il concetto di directory, i cammini relativi e

assoluti, i comandi, gli script e alcune caratteristiche di Linux e Windows con cui invita a far pratica con esempi ed esercizi. Introduce i sistemi per smartphone e tablet quali iOS, Android e RIM.

Il **Capitolo 3 Algoritmi** tratta di semplici problemi esemplificativi e relativi algoritmi di risoluzione con l'uso implicito dei meccanismi primari del controllo del flusso: risulterà evidente la necessità della memoria e il significato di passi di esecuzione. Viene data una definizione rigorosa di algoritmo e ci si domanda quale sia l'effettiva potenza del calcolo meccanico, si discute la risposta data da Turing e Church.

Questo capitolo termina con l'introduzione dei concetti di complessità computazionale e spaziale, dimensione del problema, metriche – ci chiederemo come confrontare due algoritmi che risolvono la stessa classe di problemi – e correttezza di un algoritmo rispetto a una specifica.

Il **Capitolo 4 Programmi** introduce i linguaggi e la programmazione e dà concretezza alla distinzione tra linguaggi di alto e basso livello, mostrando l'interazione tra programma e parti fisiche della macchina per poi concentrarsi sulla programmazione strutturata. Ragioneremo graficamente sul problema del "salto" e sulle conseguenze negative di un suo uso deregolamentato. Comprenderemo la soluzione offerta dalla programmazione strutturata grazie al controllo del flusso garantito da sequenza, selezione, ripetizione e dal "blocco d'istruzioni". Le strutture di controllo sono spiegate nel dettaglio utilizzando un linguaggio C ancora informale. I paragrafi conclusivi accennano agli approcci top-down e alla programmazione modulare che verranno utilizzati nel testo.

Consigliamo il lettore che non abbia già familiarità con gli argomenti esposti a iniziare lo studio dal Capitolo 1, anche se la trattazione vera e propria della programmazione strutturata in C inizia dal Capitolo 5.

L'obiettivo del **Capitolo 5 Sequenza** è mettere il lettore immediatamente in grado di realizzare i primi programmi in linguaggio C. Impareremo a visualizzare informazioni sul monitor e accettare valori da tastiera, definire e utilizzare variabili e costanti di tipo intero. Comprenderemo il concetto di funzione e sapremo come far riferimento a funzioni già definite e come invocarle. Apprenderemo le varie fasi della programmazione: editing, precompilazione, compilazione, link, caricamento in memoria ed esecuzione, compilazione ed esecuzione dei primi programmi.

Gli esempi e gli esercizi del capitolo utilizzano una sola modalità di flusso, la sequenza, in cui le istruzioni vengono eseguite una dopo l'altra nell'ordine in cui sono state scritte. È utilizzato anche un solo tipo di valori, gli interi. Particolare attenzione viene posta all'operazione di assegnamento, e numerose riflessioni accompagnano l'introduzione del concetto di variabile e costante. Prende così corpo la struttura di un programma C.

Il **Capitolo 6 Tipi fondamentali** tratta un aspetto importantissimo della programmazione di alto livello: i tipi dati messi a disposizione dal linguaggio. Conosceremo e faremo pratica con i tipi fondamentali più usati che ci serviranno nei ca-

pitoli successivi (`char`, `int`, `float`, `double`), alcuni qualificatori (`short`, `long`) e le operazioni permesse. Sapremo definire, visualizzare, accettare in ingresso ed elaborare variabili e costanti dei tipi fondamentali.

Il **Capitolo 7 Selezione** percorre due tappe indispensabili verso l'apprendimento della programmazione strutturata in linguaggio C: costrutti di controllo decisionali e blocchi d'istruzioni. Grazie al costrutto decisionale `if` faremo eseguire un'istruzione piuttosto che un'altra in base al presentarsi di una certa condizione. Comprenderemo l'utilità dei blocchi d'istruzioni e degli `if` annidati. Effettueremo scelte multiple con il costrutto `switch-case`.

Negli esempi e gli esercizi del Capitolo 7 si utilizzano due modalità di flusso: la sequenza e la selezione, in cui un'istruzione o un blocco d'istruzioni vengono eseguiti se un'espressione risulta essere vera o meno. Si utilizzano inoltre due tipi di valori: gli interi e i caratteri. Particolare attenzione viene posta alla corrispondenza tra le istruzioni operative e le differenti vie aperte dai costrutti annidati e alla necessità o meno dei blocchi d'istruzioni. Viene presentata per la prima volta la possibilità di confrontare i caratteri in base al loro ordinamento.

Il **Capitolo 8 Operatori** si concentra sulla valutazione delle espressioni. Conosceremo e sperimenteremo il significato, la gerarchia e l'associatività degli operatori aritmetici, relazionali, logici, dell'operatore di assegnamento e dell'operatore condizionale. Durante il cammino cominceremo a costruire una prima, seppur parziale, tavola degli operatori con la gerarchia di priorità, sperimentata attraverso gli esempi e gli esercizi che lavorano con le espressioni, lo strumento più potente del linguaggio, mostrandone insidie e utilità per il calcolo e per prendere decisioni nelle strutture di controllo.

Il **Capitolo 9 Iterazione** costituisce un ulteriore passo fondamentale verso la programmazione strutturata, attraverso i costrutti di controllo iterativi. Grazie a `for`, `while` e `do-while` faremo ripetere l'esecuzione di un'istruzione o di un blocco d'istruzioni. Vedremo come gestire a piacimento il numero di iterazioni quale risultato della valutazione di un'espressione, che potrà essere un valore determinato a priori o dipendente dal presentarsi di una certa condizione maturata all'interno del ciclo stesso. Saremo in grado di utilizzare cicli annidati, incrementi e decrementi e l'operatore virgola. Definiremo e utilizzeremo variabili e costanti dei tipi in virgola mobile. Approfondiremo ed estenderemo l'uso di espressioni e operatori. Si affacciano anche le interruzioni `break`, `continue` ed `exit`.

Vengono utilizzate tutte le modalità di flusso della programmazione strutturata: sequenza, diramazione e iterazione congiuntamente con i blocchi d'istruzioni. Saremo in grado di operare con caratteri, numeri interi e in virgola mobile e diverrà chiara l'intera struttura di un programma C.

Presentando le variabili strutturate di tipo array, il **Capitolo 10 Array** permette di compiere un salto qualitativo nel trattamento dei dati. Apprenderemo a definire e utilizzare array mono e multidimensionali, a gestire gli indici di vettori e matrici. Approfondiremo l'uso di operatori e cicli annidati, e saremo in grado di assegnare valori d'inizializzazione alle variabili in fase di dichiarazione.

Gli esempi e gli esercizi mostrano la gestione di sequenze di dati omogenei, in particolare si consolida quanto appreso con la ricerca di massimo, minimo, media e con il prodotto di matrici.

Il **Capitolo 11 Funzioni** riguarda un argomento che costituisce un passaggio fondamentale: la programmazione modulare, che prevede la scomposizione di un problema in parti più semplici, il programma in sottoprogrammi. Dichiareremo, definiremo e lavoreremo con funzioni nostre. Predisporremo ed effettueremo il passaggio dei parametri, gestiremo il valore di ritorno di una funzione e il tipo `void`. Sapremo interpretare e utilizzare visibilità e mascheramento dei nomi.

Il **Capitolo 12 Ricerche e ordinamenti** è un "laboratorio" per sperimentare, affrontando alcuni dei problemi "classici" della programmazione, quello che è stato appreso, senza introdurre nuovi elementi del linguaggio. Implementeremo algoritmi di ricerca, ordinamento e fusione, ci impadroniremo delle tecniche per la gestione dei vettori, approfondiremo ulteriormente l'uso di condizioni logiche e cicli.

Grande importanza rivestono i metodi di *bubblesort*, ricerca binaria e *merge*, sia per se stessi sia per l'occasione che offrono di riflettere sulle logiche di programmazione. Metteremo in pratica alcune considerazioni teoriche fatte nel Capitolo 3 sulla complessità computazionale, applicandole agli algoritmi di ricerca e ordinamento.

Il **Caso di studio I Gestione di una sequenza** è introdotto dalla spiegazione del metodo in cui tutti i casi di studio vengono affrontati: modalità top-down, divisione del problema in sottoproblemi, risoluzioni dei sottoproblemi con specifici moduli e così via. Successivamente è presentato il problema e affrontato con l'analisi e il progetto della soluzione, il diagramma dell'interazione fra i moduli con evidenziato il passaggio esplicito e implicito dei parametri e lo sviluppo dell'intero programma adeguatamente commentato.

Acquisiremo dunque familiarità con la programmazione modulare e il linguaggio proponendo all'utente un menu di possibili opzioni fino a che non sceglie di chiudere l'applicazione e utilizzando funzioni quali quelle di immissione, ordinamento, ricerca.

Il **Capitolo 13 Stringhe** riprende la trattazione del linguaggio con i vettori di caratteri. Definiremo e utilizzeremo array di `char`, approfondiremo ed estenderemo l'uso dei valori alfanumerici.

Gli esempi e gli esercizi confrontano e concatenano stringhe o parti di esse, le convertono in valori numerici, applicano funzioni appositamente previste dalla libreria standard.

L'argomento trattato nel **Capitolo 14 Puntatori** è tanto ostico quanto fondamentale: per questa ragione gli viene dedicata un'intera ampia sezione introduttiva, ricca di esempi. Definiremo e utilizzeremo i puntatori, osserveremo le variabili strutturate di tipo array sotto una nuova luce e vi opereremo attraverso i puntatori, useremo l'aritmetica dei puntatori. Predisporremo ed effettueremo il passaggio

di parametri per indirizzo mentre approfondiremo l'uso delle funzioni. Apprenderemo il significato e l'uso dinamico della memoria, sapremo allocare e deallocare correttamente spazi di memoria centrale.

Il **Caso di studio II Gestione di una sequenza con i puntatori** offre la possibilità di apprendere le tecniche per operare con oggetti dinamici – creazione, accesso, rimozione – e di sperimentare ancora il linguaggio riprendendo la *gestione di una sequenza* di valori, realizzata questa volta con i puntatori.

L'approccio è lo stesso già presentato per il Caso di studio I e valido per tutti gli altri: problema, analisi e progetto, interazione fra i moduli, sviluppo, esercizi proposti di modifica ed estensione.

Il **Capitolo 15 Ricorsione** tratta le funzioni ricorsive, funzioni che richiamano in modo diretto o indiretto se stesse. Confronteremo iterazione e ricorsione, prenderemo confidenza con il passaggio dei parametri e il valore restituito dalle funzioni, ci eserciteremo estensivamente con il calcolo combinatorio (fattoriale, disposizioni, combinazioni), la successione di Fibonacci, la torre di Hanoi e le otto regine, un esempio di algoritmo di *backtracking*. Approfondiremo gli algoritmi di ordinamento con il metodo del *quicksort*.

L'argomento è interessante almeno per tre ragioni: costringe a seguire con attenzione flusso di esecuzione e visibilità delle variabili; induce a considerazioni e contrapposizioni tra eleganza ed efficienza degli algoritmi; permette un'ulteriore interessante sperimentazione delle funzioni e del passaggio di parametri.

Il **Capitolo 16 Strutture** presenta un altro importante elemento del linguaggio: variabili strutturate che sono aggregazioni di elementi di natura diversa. Dichiareremo e utilizzeremo il tipo `struct` e sapremo lavorare con combinazioni di strutture e array.

Il **Caso di studio III Gestione anagrafica** affronta il problema della gestione di un'anagrafica in memoria centrale utilizzando la struttura dati più consona a questo scopo: un array di `struct`, ossia una struttura fatta di elementi omogenei, le persone, ciascuno dei quali è un'aggregazione di elementi di natura diversa: nome, indirizzo, età. L'argomento è anche propedeutico al trattamento dei file.

Il **Capitolo 17 File** tratta la memorizzazione permanente dei dati su memoria di massa attraverso i file. Archivieremo e manipoleremo i dati su memoria persistente; apriremo, chiuderemo, leggeremo, scriveremo e posizioneremo il puntatore all'interno degli archivi. Lavoreremo con strutture e file. Approfondiremo gli aspetti legati a standard input e standard output. Conosceremo e utilizzeremo le funzioni di basso livello e i permessi di accesso dei file.

Senza i file è poco praticabile lavorare con quantità di dati che non siano minimali, infatti li dovremmo immettere ogni volta che facciamo girare il programma. A questo punto il lettore può riprendere tutti i programmi visti a partire dagli array e trasformarli in modo che lavorino su file. Un esercizio interessante, che gli permetterà di sperimentare nuove funzioni risparmiando il tempo d'immissione di corpose sequenze di dati.

Il **Caso di studio IV Gestione anagrafica con i file** affronta ancora il problema della gestione di un'anagrafica, questa volta non più appoggiata su array di struct ma memorizzata su file.

Il **Capitolo 18 Approfondimento sui tipi e trattamento dei bit** riprende e completa il trattamento dei tipi dati fondamentali introdotti soprattutto nel Capitolo 6. Acquisiremo una visione d'insieme dei tipi fondamentali disponibili (`char`, `int`, `enum`, `float`, `double`), dei qualificatori (`short`, `long`, `unsigned`) e dell'uso di variabili e costanti. Tratteremo i dati attraverso gli operatori bit a bit, come AND, OR, shift e complemento, molto importanti rispetto alla programmazione di basso livello per la gestione di dispositivi hardware per cui il C è spesso utilizzato. Interpreteremo e opereremo conversioni di tipo implicite ed esplicite (*cast*).

Il **Capitolo 19 I/O formattato e funzioni ad argomenti variabili** tratta compiutamente le funzioni di output e input formattato e introduce la costruzione di funzioni che accettano liste di parametri variabili per numero e tipo. Acquisiremo una conoscenza completa delle funzioni di libreria `printf`, `scanf` e loro simili e ci sorprenderemo della flessibilità delle funzioni C come del `main`.

Il **Capitolo 20 Tipi Derivati e classi di memoria** riprende e completa i tipi derivati e tratta le classi di memoria. Acquisiremo una visione d'insieme dei tipi derivati semplici `void`, array, funzioni, puntatori, strutture e unioni. Definiremo e utilizzeremo unioni e campi. Sapremo lavorare con i tipi derivati composti come combinazione di strutture, array, puntatori e funzioni. Creeremo alias ai tipi dati. Divideremo il programma in moduli memorizzati su differenti file. Approfondiremo gli aspetti legati alla visibilità dei nomi. Apprenderemo la classificazione delle variabili e a utilizzare `extern`, `static`, `register`.

Nel **Capitolo 21 Strutture dati** sono presentate le strutture dati lineari e alcune loro implementazioni in linguaggio C. Approfondiremo ulteriormente l'uso di funzioni, puntatori, strutture e passaggio di parametri. Prenderemo maggiore confidenza con allocazione e deallocazione dinamica di memoria. Conosceremo e lavoreremo con le liste lineari e saremo in grado di implementare strutture astratte, quali pile e code, mediante array e liste. Verrà ulteriormente messa in pratica la scomposizione funzionale in sottoprogrammi e la ricerca di soluzioni ricorsive.

Con il **Caso di studio V Gestione di una sequenza ordinata con una lista lineare** acquisiremo dimestichezza con le strutture dati riprendendo la gestione di una sequenza e realizzandola tramite una lista lineare.

Il **Capitolo 22 Alberi e grafi** prosegue la trattazione iniziata nel capitolo precedente, passando alle strutture dati non lineari e ad alcune loro implementazioni in linguaggio C. Definiremo e lavoreremo con strutture dati non lineari. Conosceremo e implementeremo alberi binari e realizzeremo visite in ordine simmetrico, anticipato e differito. Sapremo implementare grafi e combinare array e liste.

Il **Capitolo 23 Preprocessore** presenta le possibilità offerte dal precompilatore. Acquisiremo una visione d'insieme delle direttive del precompilatore, riprenderemo approfondendole le direttive già introdotte. Conosceremo e utilizzeremo la compilazione condizionale.

Il **Capitolo 24 Semantiche e correttezza dei programmi** mette a fuoco i concetti di correttezza semantica e affidabilità di un programma. Presenta i programmi in esecuzione come sequenza di stati e la semantica semplice delle istruzioni evidenziandone le proprietà. Rilancia la "grande sfida" per ottenere un "compilatore verificatore" che non solo traduca i programmi da linguaggio ad alto livello a linguaggio macchina, ma ne verifichi anche la correttezza semantica. Comprenderemo e applicheremo in linguaggio C i metodi delle asserzioni e delle specifiche.

Nel **Caso di studio VI Progetto di gestione aziendale della fatturazione** svilupperemo un programma per la gestione aziendale della fatturazione ai clienti. Lavoreremo con dati relativi al cliente come partita IVA e ragione sociale, agli articoli come codice e descrizione e alle fatture come numero progressivo e univoco, data di emissione. I dati saranno memorizzati su file e si offrirà al responsabile amministrativo un menu interattivo di lavoro.

Il **Capitolo 25 Programmare uno Web server** affronta la realizzazione di pagine Web dinamiche attraverso la programmazione su Internet lato Server. Vedremo come sia possibile richiedere e ricevere pagine Web, immagini, file audio e simili, e soprattutto innescare dal lato server delle vere e proprie applicazioni che accettino valori in ingresso, svolgano computazioni e restituiscano in risposta nuove pagine Web.

Nel **Capitolo 26 Programmazione orientata agli oggetti** sono introdotti i concetti astratti della programmazione orientata agli oggetti attraverso esemplificazioni pratiche e l'ausilio di uno pseudo-linguaggio: uno Objective-C semplificato. Comprenderemo il significato di generalizzazione e specializzazione, classe e oggetto, istanza, ereditarietà, superclasse e classe derivata, polimorfismo e biding dinamico.

Il **Capitolo 27 Linguaggio Objective-C** introduce il linguaggio Ojective-C. Apprenderemo a scrivere i primi programmi e a compilarli, a usare le librerie predefinite, a produrre un output formattato a istanziare un oggetto, a usare i tipi booleani e gli oggetti generici. Faremo pratica nell'uso dei messaggi che costituiscono la base del linguaggio; manipoleremo le stringhe.

Nel **Capitolo 28 Programmare con la libreria base** l'apprendimento dell'Objective-C procede per esemplificazioni che inizialmente usufruiscono di alcune tra le più importanti classi di base che il linguaggio mette a disposizione. Lavoreremo con stringhe, array, dizionari, facendo pratica di programmazione ad oggetti.

Il **Capitolo 29 Costruire classi** mostra come costruire proprie gerarchie di classi e presenta le diverse opzioni disponibili per la gestione della memoria. Apprenderemo a definire l'interfaccia della classe, a ereditare dalla classe radice di tutte le classi, a gestire la visibilità delle variabili d'istanza, a implementare i metodi, a costruire oggetti "in fabbrica", a gestire la memoria rendendo libera quella non più utilizzata. Soprattutto costruiremo e useremo le sottoclassi sperimentando l'ereditarietà.

Se nei capitoli precedenti si sono evidenziati i vantaggi ottenuti con il livello di astrazione permesso dall'ereditarietà, nel **Capitolo 30 Protocolli, Categorie, Introspezione ed Eccezioni** si presentano anche le difficoltà e si introducono alcune opzioni costituite dai meccanismi di comunicazione tra classi attraverso i protocolli formali e informali del linguaggio. Implementeremo i protocolli formali, le classi deleganti e delegate, e quelli informali realizzando la comunicazione fra le classi mediante le categorie.

L'**Appendice A Sintassi** riporta la sintassi formale dell'ultima versione dello standard, in modo che il lettore durante lo studio possa far riferimento a un quadro completo e rigoroso del linguaggio. L'appendice è arricchita da alcune note sulle novità della sintassi del C e dalle parole chiavi del C++.

L'**Appendice B Librerie standard** e l'**Appendice C Operatori** forniscono quadri riassuntivi sul linguaggio di rapida consultazione durante lo studio e il lavoro di programmazione.

L'**Appendice D Rappresentazione dell'informazione** aiuta alla comprensione del trattamento della memoria, dei tipi dati, delle operazioni matematiche e non; è dunque utile alla piena comprensione o all'approfondimento di alcuni temi.

L'**Appendice E Codice ASCII** riporta il codice, in modo da averlo a portata di mano nelle prove di programmazione, assieme ad alcune considerazioni sulle tipologie di caratteri rappresentati e alle codifiche Unicode UTF.

L'**Appendice F Embedded SQL** e l'**Appendice G Pagine HTML** trattano argomenti correlati a quelli trattati.

Sul sito web del libro

Sul sito **www.ateneonline.it/guidi** è disponibile la documentazione digitale relativa a questo libro.

Vi si possono scaricare: **alcuni capitoli e appendici**, **i listati dei programmi di tutto il testo**, **le soluzioni di molti degli esercizi proposti** e altro materiale didattico di supporto per professori e studenti.

Ringraziamenti

Nel cammino che ci ha portato dalla prima edizione del libro all'attuale, sono stati tanti e generosi i contributi che abbiamo ricevuto. Un riconoscimento particolare all'amico matematico professor Nazario Renzoni, per i suoi consigli sulle macchine di Turing e gli algoritmi ricorsivi della torre di Hanoi e delle nove regine, e al professor Maurizio Vincini dell'Università degli studi di Modena e Reggio Emilia autore dell'ultimo più ampio caso di studio, un progetto software parte integrante del testo denso di contenuti solidi che hanno un riflesso immediato nella realtà pratica del lavoro dello sviluppatore.

Ringraziamo inoltre tutti i docenti che hanno partecipato alle review delle diverse edizioni. Ci sono state preziose tanto le loro riflessioni generali sull'insieme del lavoro quanto le loro osservazioni specifiche e concrete sulle singole parti.

Alcuni studenti ci hanno scritto in questi anni permettendoci di correggere errori e imprecisioni, dandoci alcune perspicaci indicazioni. A tutti va il nostro più sentito ringraziamento, a noi ogni responsabilità per le decisioni prese. Al lettore auguriamo di cuore buono studio e buon lavoro. Chi lo desidera può inviarci commenti e consigli ai seguenti indirizzi di posta elettronica: **andreag@softpi.com** e **abel@mathema.com**.

Andrea Guidi
Alessandro Bellini