Capitolo 2

Il vostro primo programma

Ora che conoscete la struttura di questo libro, è tempo di capire come funziona la programmazione per il Mac e i dispositivi iOS. A questo scopo:

- installerete gli strumenti per sviluppatori messi a disposizione da Apple;
- creerete un semplice progetto usando questi strumenti;
- vedrete come si utilizzano questi strumenti per assicurarvi che il vostro progetto funzioni.

Al termine di questo capitolo avrete scritto il vostro primo programma per il Mac.

In questo capitolo

- **Installare gli strumenti per sviluppatori di Apple**
- **Primi passi con Xcode**
- **Da dove si comincia per scrivere il codice?**
- **Come si esegue il programma?**
- **Ma che cos'è un programma?**
- **Non fermatevi mai**

Installare gli strumenti per sviluppatori di Apple

Per scrivere applicazioni destinate a OS X (Mac) o iOS (iPhone e compagni) utilizzerete gli strumenti per sviluppatori di Apple. L'applicazione principale si chiama Xcode ed è disponibile soltanto sul Mac (non per Windows o Linux), perciò per lavorare con questo libro vi serve un Mac. Inoltre, questo libro si basa su Xcode 5, che è compatibile con OS X 10.8 (Mountain Lion) e versioni successive. Potete scaricare gratuitamente l'ultima versione di Xcode dal Mac App Store. Suggeriamo di trascinare l'icona di Xcode sul Dock, dato che la utilizzerete molto spesso.

Primi passi con Xcode

Xcode è l'ambiente di sviluppo integrato (IDE) di Apple; mette a disposizione tutto ciò che vi serve per scrivere, compilare ed eseguire nuove applicazioni.

Una nota sulla terminologia: chiamiamo *programma* qualsiasi cosa che sia eseguibile su un computer. Alcuni programmi dispongono di un'interfaccia utente grafica, e in questo caso li chiamiamo *applicazioni*.

Alcuni programmi sono privi di interfaccia grafica e vengono eseguiti per giorni in background; li chiamiamo *demoni*. Il termine incute paura, ma non è il caso di averne. Probabilmente avrete una sessantina di demoni in esecuzione sul vostro Mac, in questo momento. Rimangono in attesa, aspettando il momento in cui possono rendersi utili. Per esempio, uno dei demoni in esecuzione sul vostro sistema si chiama `pboard`. Quando eseguite un'operazione di copia e incolla, il demone `pboard` mantiene in memoria i dati che state copiando.

Alcuni programmi non hanno interfaccia grafica e vengono eseguiti per breve tempo in una finestra di Terminale: li chiamiamo *strumenti della riga di comando*. In questo libro scriverete prevalentemente questi tipi di programmi, in modo da focalizzarvi sui concetti essenziali della programmazione senza distrarvi creando e gestendo un'interfaccia utente.

Nel seguito creerete un semplice strumento della riga di comando utilizzando Xcode, così potrete vedere come funziona il tutto.

Quando scrivete un programma, create e modificate una serie di file. Xcode tiene traccia di tali file in un *progetto*. Avviate Xcode, aprite il menu *File*, selezionate *New* e poi *Project*.

Per aiutarvi, Xcode mette a disposizione un certo numero di template o modelli di progetto. Potete scegliere un template corrispondente al tipo di programma che avete l'intenzione di scrivere. Nella colonna di sinistra selezionate *Application* nella sezione *OS X*, quindi selezionate *Command Line Tool* tra le opzioni che appaiono sulla destra (Figura 2.1).

Fate clic sul pulsante *Next*.

Assegnate al vostro nuovo progetto il nome *AGoodStart*. Per gli esercizi di questo libro il nome dell'organizzazione e l'identificatore dell'azienda non contano, ma dovete comunque inserire qualcosa per continuare. Inserite *Big Nerd Ranch* e *com.bignerdranch*. Dal menu a comparsa *Type*, selezionate *C* (Figura 2.2).

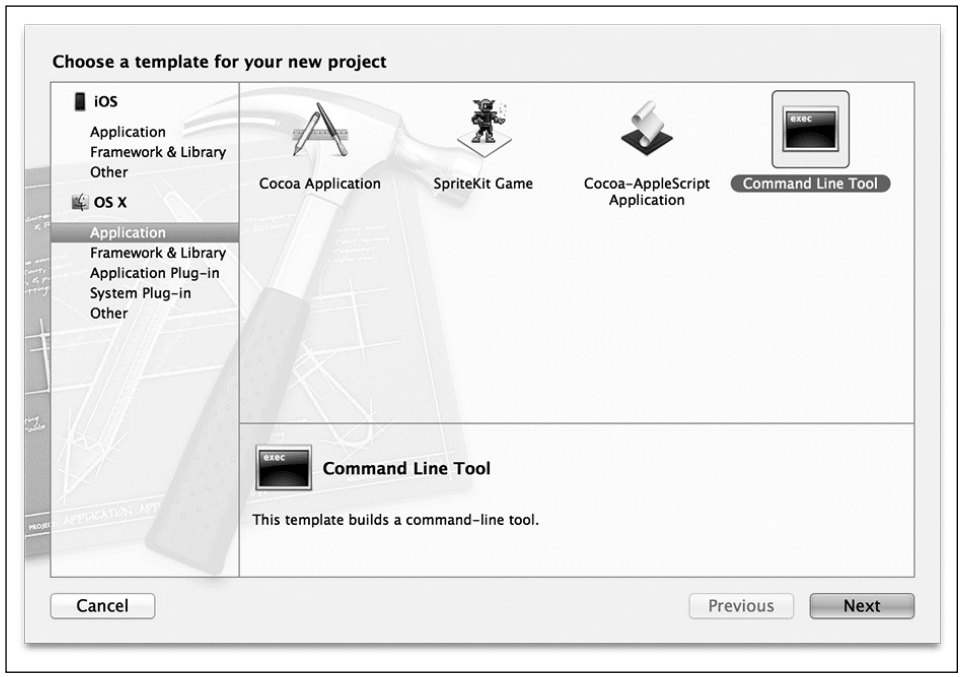

Figura 2.1 Scelta di un template.

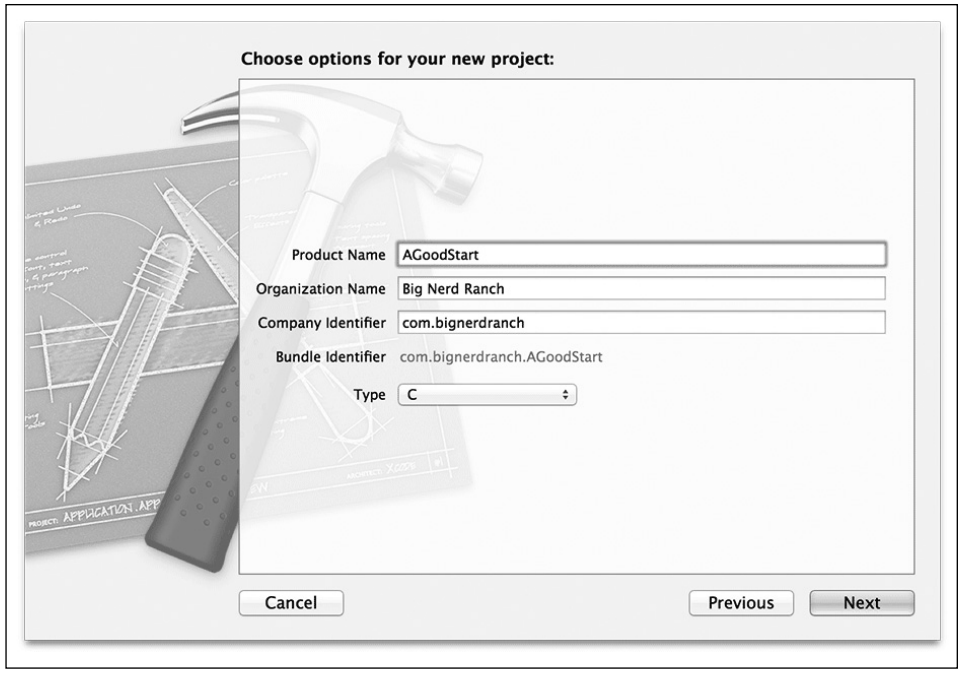

Figura 2.2 Scelta di opzioni per il progetto.

Fate clic sul pulsante *Next*.
Nella finestra che segue scegliete la cartella in cui volete che sia creata la directory del progetto (se non sapete che cosa fare, accettate la posizione predefinita proposta da *Xcode*). Non vi servirà un repository per il controllo delle versioni, perciò disattivate la casella etichettata *Create git repository*. Infine, fate clic sul pulsante *Create*.
Anche nei prossimi capitoli creerete questo stesso tipo di progetto, perciò più avanti ci limiteremo a indicare: "Create un nuovo strumento della riga di comando in C denominato *nome-del-programma*" per specificare la procedura qui descritta.
Perché abbiamo indicare di creare progetti in C? Perché Objective-C è basato proprio sul linguaggio di programmazione C, e dovete comprendere alcune parti del linguaggio C prima di poter entrare nei dettagli di Objective-C.

Da dove si comincia per scrivere il codice?

Dopo aver creato il progetto, vedrete una finestra di benvenuto con numerose informazioni su *AGoodStart* (Figura 2.3).

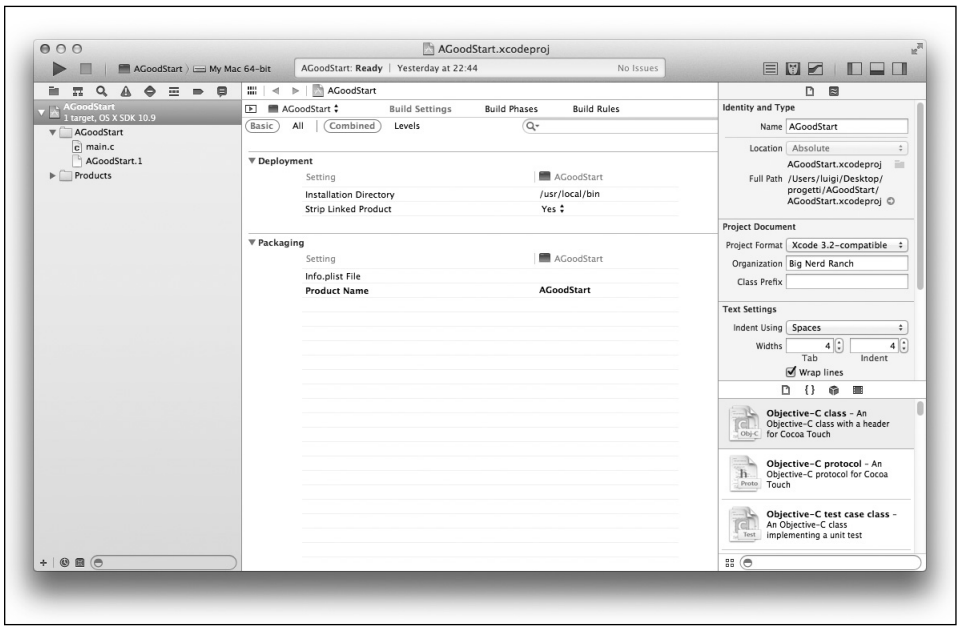

Figura 2.3 La prima visualizzazione del progetto AGoodStart.

Questa finestra contiene più dettagli di quanti servano, perciò cerchiamo di renderla più semplice. Per prima cosa, nell'angolo superiore destro dello schermo trovate tre pulsanti:  .

Con questi pulsanti potete visualizzare e nascondere diverse parti della finestra. Per ora la parte più a destra non vi servirà, perciò fate clic sul pulsante più a destra per nasconderla.
Ora avete a disposizione due aree: quella di navigazione sulla sinistra e quella dell'editor sulla destra.

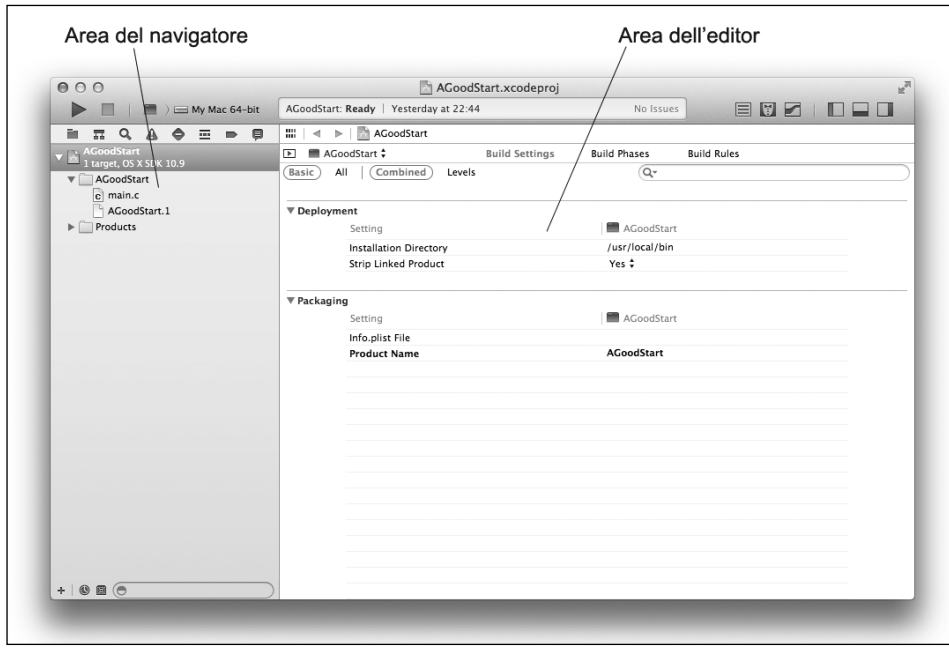

Figura 2.4 Le aree di navigazione e dell'editor in Xcode.

L'area di navigazione mostra il navigatore corrente. Esistono diversi navigatori, ognuno dei quali fornisce un modo diverso per esaminare il contesto del progetto. Ora state osservando il *navigatore del progetto*, che elenca i file che costituiscono il progetto.
Nel navigatore del progetto, trovate un file denominato `main.c` e fate clic su di esso (se non lo trovate, fate clic sul triangolino accanto alla cartella *AGoodStart* per mostrare il contenuto di quest'ultima).
Quando selezionate `main.c` nel navigatore del progetto, l'area dell'editor cambia per visualizzare il contenuto del file (Figura 2.5).

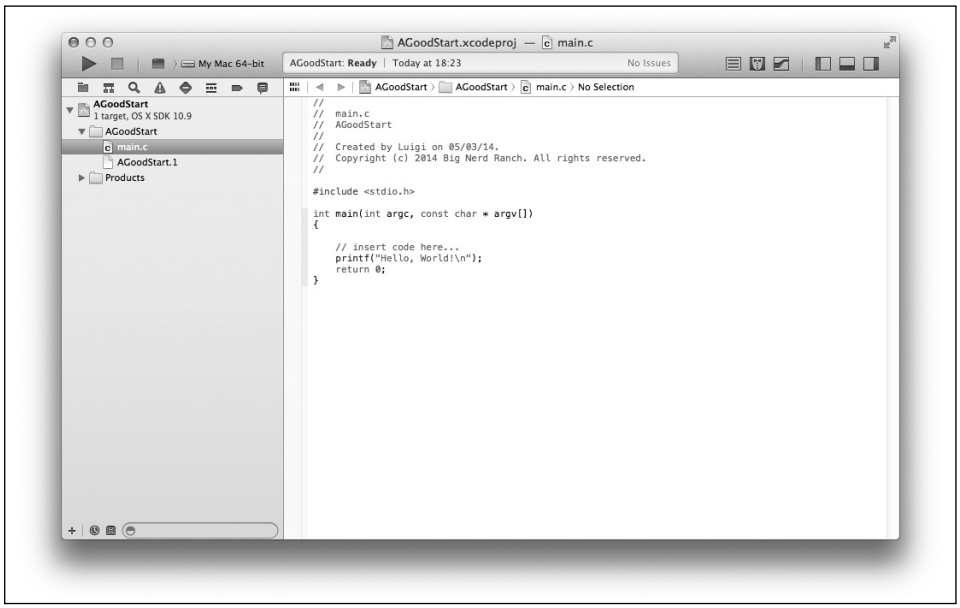

Figura 2.5 Selezione di main.c nel navigatore del progetto.

Il file `main.c` contiene una *funzione* denominata `main`. Una funzione è un elenco di istruzioni che il computer deve eseguire, e le è sempre assegnato un nome. In un programma C o Objective-C, `main` è il nome della funzione che viene richiamata all'avvio di un programma.

```
#include <stdio.h>

int main(int argc, const char * argv[]) {

    // inserire codice qui...
    printf("Hello, World!\n");
    return 0;
}
```

Questa funzione contiene i due tipi di informazioni che si inseriscono in un programma: codice e commenti.

- Il codice è l'insieme di istruzioni che indicano al computer che cosa fare.
- I commenti sono ignorati dal computer, ma servono ai programmatori per documentare il codice scritto. Più è difficile il problema di programmazione che si cerca di risolvere, più sono utili i commenti per documentare il modo in cui il problema viene risolto. L'importanza di tale documentazione risulta evidente quando si torna a riesaminare il programma mesi dopo, si osserva il codice che si è dimenticato di commentare e si pensa: "Sono certo che questa soluzione sia brillante, ma non ricordo proprio come funzioni".

In C e Objective-C esistono due modi per distinguere i commenti dal codice.

- Se inserite `//` in una riga di codice, tutto quanto segue la doppia barra fino alla fine della riga è considerato un commento. Potete vedere un esempio nel codice precedente, con il commento "Inserire codice qui...".
- Se volete inserire commenti più lunghi, potete utilizzare `/*` e `*/` per contrassegnare l'inizio e la fine di commenti che si estendono su più righe.

Queste regole per la definizione di commenti fanno parte della *sintassi* del linguaggio C. La sintassi è l'insieme di regole che governano il modo in cui si deve scrivere il codice in un determinato linguaggio di programmazione. Si tratta di regole estremamente specifiche che vanno seguite, altrimenti il programma non funziona.
Mentre la sintassi relativa ai commenti è piuttosto semplice, quella del codice può assumere vari livelli di difficoltà in base a ciò che il codice fa e a come lo fa. In ogni caso, ogni *istruzione* deve terminare con un punto e virgola (vedrete altri esempi di codice tra breve). Se dimenticate un punto e virgola commettete un errore di sintassi e il programma non funzionerà.
Fortunatamente Xcode è in grado di avvisare quando rileva questi tipi di errori. In effetti, una delle prime sfide che i programmatori devono affrontare è quella di interpretare gli avvertimenti di Xcode quando qualcosa va storto e poi di correggere gli errori. Nel libro vedrete alcuni dei messaggi visualizzati da Xcode per gli errori di sintassi più comuni.
Ora apportate qualche modifica a `main.c`. Per prima cosa dovete creare un po' di spazio. Trovate le parentesi graffe ({ e }) che contrassegnano l'inizio e la fine della funzione `main`, e cancellate tutto ciò che si trova racchiuso tra di esse.
Ora sostituite il contenuto della funzione `main` con il codice riportato di seguito. In pratica aggiungete un commento, due istruzioni di codice e un altro commento. Non preoccupatevi se non capite ciò che digitate, per ora vogliamo semplicemente cominciare a programmare qualcosa. Avete l'intero libro a disposizione per capire di che cosa si tratta.

```
#include <stdio.h>

int main (int argc, const char * argv[])
{
    // Stampa l'inizio del racconto
    printf("Erano i tempi migliori.\n");
    printf("Erano i tempi peggiori.\n");
    /* Va bene davvero?
    Forse bisognerebbe riscriverlo. */

    return 0;
}
```

Notate che il nuovo codice da digitare è evidenziato in grassetto. Il codice non in grassetto è quello già presente nel file. Utilizzeremo questa convenzione di scrittura in tutto il libro.

Mentre digitate, potreste notare che Xcode cerca di venirvi in aiuto con dei suggerimenti. Questa funzionalità si chiama *completamento del codice* ed è molto utile. Ora ignoratela e concentratevi sul digitare voi stessi il codice, ma proseguendo nella lettura del libro, potrete iniziare a sperimentare tale funzionalità per vedere come può aiutarvi a scrivere il codice in maniera più comoda e anche più precisa.
(Potete vedere e impostare le diverse opzioni per il completamento del codice nelle preferenze di Xcode. Selezionate *Xcode* → *Preferences* e aprite le preferenze relative alla modifica del testo).
Xcode utilizza diversi colori per facilitare l'individuazione dei commenti e delle diverse parti del codice. Per esempio, i commenti sono sempre riportati in verte. Quando acquisirete un po' di familiarità con Xcode, inizierete a notare per istinto quando i colori non sono giusti. Spesso questo indica che avete commesso un errore di sintassi. E più presto vi accorgete di aver commesso un errore, più facile sarà trovarlo e correggerlo.

Come si esegue il programma?

Ora eseguite il programma e osservatene il funzionamento. Si procede in due passaggi. Xcode *genera* il programma e poi lo *esegue*. Durante la fase di generazione del programma, Xcode prepara il codice all'esecuzione, eseguendo tra l'altro una serie di controlli per verificare la presenza di errori sintattici e di altro tipo.
Nell'angolo superiore sinistro della finestra del progetto trovate il pulsante che assomiglia a quello di riproduzione in iTunes o su un lettore di DVD. Se posizionate il puntatore sopra tale pulsante, vedrete apparire un suggerimento che indica "Build and then run the current scheme", cioè "Genera e poi esegue lo schema corrente". Fate clic su quel pulsante.
Se tutto va bene, sarete ricompensati con:

Altrimenti vedrete:

E in quest'ultimo caso che cosa dovete fare? Confrontate con attenzione il codice da voi inserito con quello riportato nel libro. Cercate errori di battitura e punti e virgola mancanti. Xcode evidenzierà le righe che considera problematiche. Dopo che avete trovato e corretto l'errore, fate clic nuovamente sul pulsante di esecuzione, ed eventualmente ripetete tutta la procedura finché ottenete un'esecuzione corretta. (Se vi capita di non riuscire a generare codice senza errori, con gli esempi di questo libro o altro codice che scriverete in futuro, non dovete abbattervi. Gli errori, commessi e corretti, vi aiutano a comprendere ciò che fate. In effetti, le prime volte è meglio sbagliare piuttosto che fare tutto giusto per un colpo di fortuna.)
Quando il programma è generato con successo, nella parte inferiore della finestra appare una nuova area (Figura 2.6). La metà destra di tale area è la *console* e mostra l'output del codice in esecuzione:

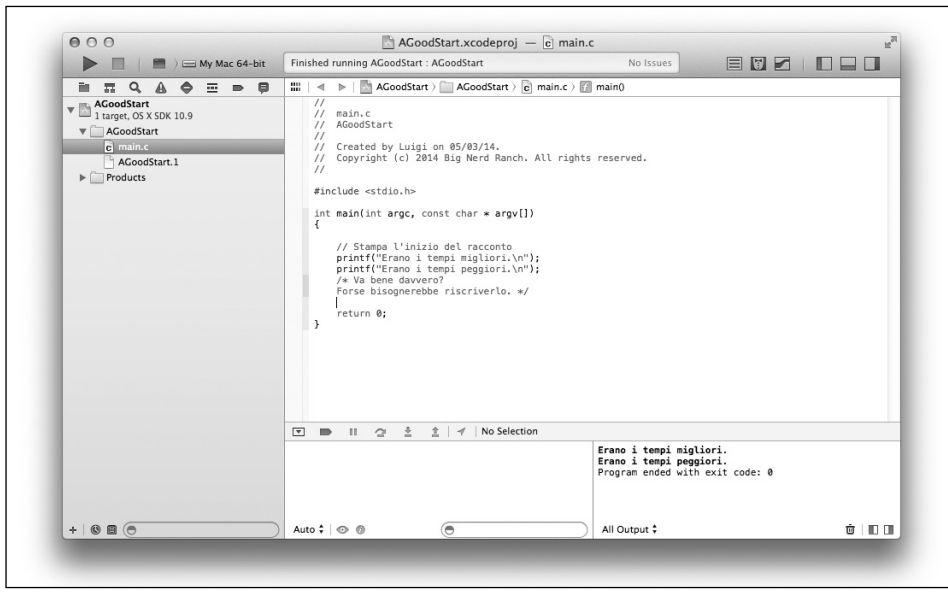

Figura 2.6 Output visualizzato nella console in basso a destra.

Ma che cos'è un programma?

Ora che avete scritto ed eseguito il vostro primo programma, cerchiamo di capire meglio come funziona. Un programma è un insieme di funzioni. E una funzione è un elenco di operazioni che il processore deve eseguire.
Ogni funzione ha un nome, e la funzione che avete appena scritto si chiama `main`.
Quando i programmatori parlano di funzioni, solitamente inseriscono una coppia di parentesi vuote, quindi la funzione `main` è indicata come `main()`.
Nel programma precedente c'è un'altra funzione: `printf()`. Non l'avete scritta voi, ma l'avete usata.

Per un programmatore, scrivere una funzione è un po' come scrivere una ricetta. Come una funzione, una ricetta ha un nome e una serie di istruzioni (Figura 2.7). La differenza è che le ricette si eseguono in cucina, mentre le funzioni sono eseguite dal computer.

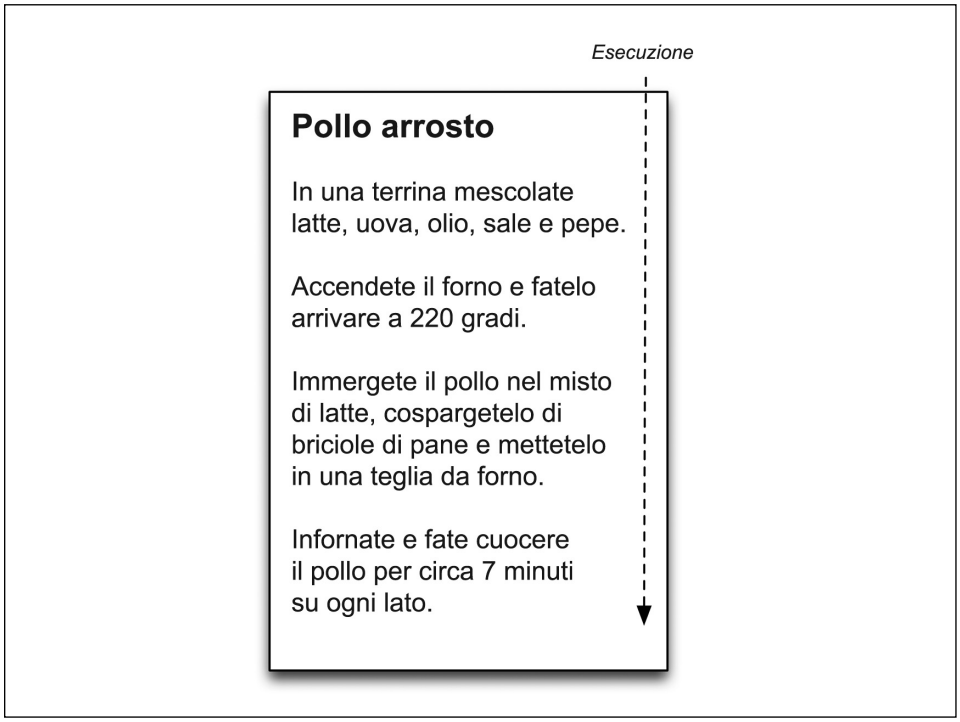

Figura 2.7 Una ricetta per cucinare il pollo arrosto.

Queste istruzioni di cucina sono scritte in italiano. Nella prima parte di questo libro le funzioni sono scritte in linguaggio C. Tuttavia, il processore del computer ha bisogno di istruzioni in codice macchina. Come si ottengono?

Quando scrivete un programma in C (che è relativamente facile da leggere per voi), il *compilatore* traduce le sue funzioni in codice macchina (facile da capire ed efficiente per il processore). Il compilatore è anch'esso un programma eseguito da Xcode quando premete il pulsante di esecuzione. Compilare un programma significa generarlo, e utilizzeremo i due termini come sinonimi.

Quando eseguite un programma, le funzioni compilate vengono copiate dal disco fisso nella memoria del computer, e la funzione denominata `main` viene eseguita dal processore. La funzione `main` solitamente richiama altre funzioni.

Per esempio, la funzione `main` dell'esempio richiama la funzione `printf`. Esamineremo meglio il funzionamento delle funzioni nel Capitolo 5.

Non fermatevi mai

Per giungere a questo punto, probabilmente avete dovuto affrontare diversi inconvenienti: problemi di installazione, errori di battitura, molti termini nuovi da apprendere. E forse non riuscite nemmeno a comprendere il senso di ciò che avete fatto finora. Ma è del tutto normale.

Otto, il figlio di Aaron l'autore di questo libro, ha sei anni, e più volte al giorno gli capita di essere confuso. Cerca continuamente di assorbire conoscenze che non rientrano nella sua attuale struttura mentale. Questi inconvenienti si verificano tanto spesso che ormai non se ne preoccupa nemmeno molto. Non si ferma mai a pensare: "Perché è così difficile? Dovrei buttare questo libro nel cestino?".

Quando diventiamo adulti, ci capita meno spesso di trovarci confusi – non perché sappiamo tutto, ma perché teniamo a tenerci distanti dalle cose che ci confondono. Per esempio, leggere un libro di storia può essere piacevole perché ne otteniamo conoscenze che possiamo gestire con la nostra struttura mentale. Si tratta di un tipo di apprendimento facile.

Imparare una nuova lingua è un esempio di apprendimento difficile. Sapete che milioni di persone parlano quella lingua senza alcun problema, ma a voi sembra incredibilmente strana e difficile. E quando le persone la parlano rivolgendovi a voi, spesso rimanete sconcertati.

Anche apprendere a programmare un computer è difficile. Di tanto in tanto vi sentirete confusi, soprattutto all'inizio. Ma è giusto. Anzi, in un certo senso è bello, è un po' come tornare a quando avevate sei anni. Continuate a leggere questo libro: vi promettiamo che la confusione cesserà prima di arrivare all'ultima pagina.