Esercizi

8.5 Scrivete un programma che prenda in input dalla tastiera un carattere e lo controlli con ognuna delle funzioni incluse nella libreria per la gestione dei caratteri. Il programma dovrà visualizzare il valore restituito da ogni funzione.

8.6 Scrivete un programma che prenda in input una riga di testo con la funzione `gets` e la immagazzini nel vettore di caratteri `s[100]`. Inviate in output la riga prima in lettere maiuscole e poi in minuscole.

8.7 Scrivete un programma che prenda in input 4 stringhe che rappresentino degli interi, le converta in interi, sommi i valori ottenuti e visualizzi il loro totale.

8.8 Scrivete un programma che prenda in input 4 stringhe che rappresentino dei valori in virgola mobile, le converta in `double`, sommi i valori ottenuti e visualizzi il loro totale.

8.9 Scrivete un programma che utilizzi la funzione `strcmp` per confrontare due stringhe immesse dall'utente. Il programma dovrà stabilire se la prima stringa è minore, uguale o maggiore della seconda.

8.10 Scrivete un programma che utilizzi la funzione `strncmp` per confrontare due stringhe immesse dall'utente. Il programma dovrà ricevere in input il numero di caratteri da confrontare e dovrà stabilire se la prima stringa è minore, uguale o maggiore della seconda.

8.11 Scrivete un programma che utilizzi la generazione di numeri casuali per creare delle frasi. Il programma dovrà utilizzare quattro vettori di puntatori a `char` chiamati `article`, `noun`, `verb` e `preposition` e dovrà creare una frase selezionando una parola a caso da ognuno dei vettori nel seguente ordine: `article`, `noun`, `verb`, `preposition`, `article` e `noun`. Man mano che selezionate le singole parole queste dovranno essere concatenate a quelle precedenti, in un vettore che sia sufficientemente grande per contenere la frase intera. Le parole dovranno essere separate da spazi. Quando la frase finale sarà inviata in output, questa dovrà incominciare con una lettera maiuscola e terminare con un punto. Il programma dovrà generare 20 frasi.

I vettori dovranno essere riempiti come segue: `article` dovrà contenere gli articoli `"the"`, `"a"`, `"one"`, `"some"` e `"any"`; `noun` dovrà contenere i nomi `"boy"`, `"girl"`, `"dog"`, `"town"` e `"car"`; `verb` dovrà contenere i verbi `"drove"`, `"jumped"`, `"ran"`, `"walked"` e `"skipped"`; `preposition` dovrà contenere le preposizioni `"to"`, `"from"`, `"over"`, `"under"` e `"on"`.

Una volta che il suddetto programma sarà stato scritto e sarà funzionante, modificatelo in modo che scriva una breve storia formata da una serie di queste frasi. (Vi piace l'idea di uno scrittore casuale?).

8.12 *(Limerick)* Una limerick è una poesia umoristica di cinque versi in cui il primo e il secondo verso fanno rima con il quinto, mentre il terzo fa rima con il quarto. Usando tecniche simili a quelle sviluppate nell'Esercizio 8.10, scrivete un programma C che produca una serie di limerick a caso. Raffinare questo programma in modo che generi delle buone limerick sarà un compito impegnativo, ma il risultato varrà bene la fatica fatta!

8.13 Scrivete un programma che codifichi delle frasi della lingua inglese in pig Latin (Latino del maiale). Il pig Latin è una forma di linguaggio codificato usato spesso per divertimento. Esistono molte versioni dei metodi utilizzati per formare delle frasi in pig Latin. Per semplicità, utilizzate il seguente algoritmo:

Per formare una frase in pig Latin formulatene una in lingua inglese e suddividetela in parole con la funzione `strtok`. Per tradurre ogni parola inglese nella corrispondente in pig Latin, spostate la prima lettera della parola inglese in coda alla stessa e aggiungete le lettere `"ay"`. In questo modo, la parola `"jump"` diventerà `"umpjay"`, `"the"` si trasformerà in `"hetay"` e `"computer"` diventerà `"omputercay"`. Gli spazi tra le parole rimarranno tali. Supponete quanto segue: la frase in inglese consisterà di parole separate da spazi, non ci saranno segni di punteggiatura e tutte le parole saranno formate da due o più lettere. La funzione `printLatinWord` dovrà visualizzare ogni parola. Suggerimento: ogni volta che una chiamata di `strtok` avrà trovato un token, passate il puntatore ottenuto alla funzione `printLatinWord` e visualizzate la parola in pig Latin.

8.14 Scrivete un programma che prenda in input un numero telefonico in una stringa del formato `(555) 555-5555`. Il programma dovrà utilizzare la funzione `strtok` per estrarre il token del prefisso, quello delle prime tre e delle ultime quattro cifre del numero telefonico. Le sette cifre del numero telefonico dovranno essere concatenate in una stringa. Il programma dovrà convertire il prefisso in un

`int` e la stringa del numero telefonico in un `long`. Dovranno essere visualizzate entrambe le informazioni: prefisso e numero telefonico.

8.15 Scrivete un programma che prenda in input una riga di testo, la suddivida in token con la funzione `strtok` e li invii in output in ordine inverso.

8.16 Scrivete un programma che prenda in input dalla tastiera una riga di testo e una stringa di ricerca. Utilizzando la funzione `strstr`, individuate la prima occorrenza della stringa di ricerca nella riga di testo e assegnate la locazione ottenuta alla variabile `searchPtr` di tipo `char *`. Nel caso che la stringa di ricerca sia stata ritrovata, cominciando da questa, visualizzate la riga di testo. Utilizzate quindi nuovamente `strstr` per individuare nella riga di testo la successiva occorrenza della stringa di ricerca. Nel caso che sia stata trovata una seconda occorrenza della stringa di ricerca, visualizzate la riga di testo cominciando dalla seconda occorrenza. Suggerimento: la seconda invocazione di `strstr` dovrà contenere `searchPtr + 1` come suo primo argomento.

8.17 Scrivete un programma, basato sull'Esercizio 8.16, che prenda in input diverse righe di testo e una stringa di ricerca e utilizzi la funzione `strstr` per determinare il totale delle occorrenze della stringa nelle righe di testo. Visualizzate il risultato.

8.18 Scrivete un programma che prenda in input diverse righe di testo e un carattere da ricercare e utilizzi la funzione `strchr`, per determinare il totale delle occorrenze del carattere nelle righe di testo.

8.19 Scrivete un programma, basato su quello dell'Esercizio 8.18, che prenda in input diverse righe di testo e utilizzi la funzione `strchr` per determinare il totale delle occorrenze nelle righe di testo di ogni carattere incluso nell'alfabeto. Le lettere maiuscole e quelle minuscole dovranno essere contate insieme. Immagazzinate i totali di ogni lettera in un vettore e, una volta che saranno stati determinati, visualizzate i suddetti valori in un formato tabulare.

8.20 Scrivete un programma che prenda in input diverse righe di testo e utilizzi la funzione `strtok` per contare il numero totale delle parole. Supponete che queste siano separate da spazi o da caratteri newline.

8.21 Utilizzate le funzioni di confronto delle stringhe discusse nella Sezione 8.6 e le tecniche di ordinamento dei vettori sviluppate nel Capitolo 6, per scrivere un programma che disponga in ordine alfabetico un elenco di stringhe. Utilizzate i nomi di 10 o 15 città della vostra zona come dati per il programma.

8.22 La tabella nell'Appendice D mostra le rappresentazioni in codice numerico dei caratteri dell'insieme ASCII. Studiate questa tabella e quindi stabilite se ognuna delle seguenti affermazioni sia vera o falsa.

a) La lettera "`A`" precede la "`B`".
b) La cifra "`9`" precede lo "`0`".
c) I simboli comunemente utilizzati per l'addizione, la sottrazione, la moltiplicazione e la divisione precedono tutte le cifre.
d) Le cifre precedono le lettere.
e) Nel caso che un programma di ordinamento ordinasse delle stringhe in modo ascendente, allora quel programma sistemerebbe il simbolo della parentesi tonda chiusa prima di quello della parentesi tonda aperta.

8.23 Scrivete un programma che legga una serie di stringhe e visualizzi solo quelle che iniziano con la lettera "`b`".

8.24 Scrivete un programma che legga una serie di stringhe e visualizzi solo quelle che terminano con le lettere "`ED`".

8.25 Scrivete un programma che prenda in input un codice ASCII e visualizzi il carattere corrispondente. Modificate questo programma, in modo che generi tutti i possibili codici di tre cifre compresi nell'intervallo da 000 a 255 e tenti di visualizzare il carattere corrispondente. Che cosa succederà quando questo programma sarà eseguito?

8.26 Usando come guida la tabella dei caratteri ASCII dell'Appendice D, scrivete le vostre versioni delle funzioni per la gestione dei caratteri presentate nella Figura 8.1.

8.27 Scrivete le vostre versioni delle funzioni per la conversione delle stringhe in numeri presentate nella Figura 8.5.

8.28 Scrivete due versioni di ognuna delle funzioni per la copia e la concatenazione delle stringhe presentate nella Figura 8.17. La prima versione dovrà utilizzare gli indici di vettore, mentre la seconda dovrà utilizzare i puntatori e la relativa aritmetica.

8.29 Scrivete le vostre versioni delle funzioni **`getchar`**, **`gets`**, **`putchar`** e **`puts`** descritte nella Figura 8.12.

8.30 Scrivete due versioni per ognuna delle funzioni per la comparazione delle stringhe presentate nella Figura 8.20. La prima versione dovrà utilizzare gli indici di vettore, mentre la seconda versione dovrà utilizzare i puntatori e la relativa aritmetica.

8.31 Scrivete le vostre versioni delle funzioni di ricerca nelle stringhe presentate nella Figura 8.22.

8.32 Scrivete le vostre versioni delle funzioni per la manipolazione dei blocchi di memoria presentate nella Figura 8.30.

8.33 Scrivete due versioni della funzione **`strlen`** presentata nella Figura 8.36. La prima versione dovrà utilizzare gli indici di vettore, mentre la seconda dovrà utilizzare i puntatori e la relativa aritmetica.

Sezione speciale: esercizi di manipolazione avanzata delle stringhe

Gli esercizi precedenti sono stati incentrati sul testo e progettati per verificare la vostra comprensione dei concetti fondamentali della manipolazione delle stringhe. Questa sezione includerà una collezione di problemi di livello intermedio e avanzato. Il lettore dovrebbe trovare questi problemi impegnativi, ma anche divertenti. La difficoltà dei problemi varierà considerevolmente. Alcuni richiederanno un'ora o due per la scrittura del programma e per l'implementazione. Altri saranno utili per attività di laboratorio che potrebbero richiedere due o tre settimane per lo studio e per l'implementazione. Altri ancora sono progetti che vi impegneranno per un intero trimestre.

8.34 *(Analisi del testo)* La disponibilità dei computer, con le loro capacità di manipolazione delle stringhe, ha prodotto alcuni approcci piuttosto interessanti per analizzare gli scritti dei grandi autori. Molta attenzione è stata concentrata sul sospetto che William Shakespeare non sia mai esistito. Alcuni studiosi ritengono che ci siano valide dimostrazioni secondo le quali Christopher Marlowe avrebbe scritto in realtà i capolavori attribuiti a Shakespeare. I ricercatori hanno utilizzato i computer per trovare delle somiglianze negli scritti di questi due autori. Questo esercizio esaminerà tre metodi per analizzare i testi con il computer.

a) Scrivete un programma che legga diverse righe di testo e visualizzi una tabella indicante il numero di occorrenze nel testo per ogni lettera dell'alfabeto. Per esempio, la frase

```
To be, or not to be: that is the question:
```

contiene una "a", due "b", nessuna "c", ecc.

b) Scrivete un programma che legga diverse righe di testo e visualizzi una tabella indicante il numero di parole formate da una sola lettera, da due, da tre, ecc che appaiono nel testo. Per esempio, la frase

```
Whether 'tis nobler in the mind to suffer
```

contiene

Lunghezza della parola	Occorrenze
1	0
2	2
3	2
4	2 (inclusa 'tis)
5	0
6	2
7	1

c) Scrivete un programma che legga diverse righe di testo e visualizzi una tabella indicante il numero di occorrenze nel testo di ogni parola diversa. La prima versione del vostro programma dovrà includere le parole nella tabella sistemandole nell'ordine in cui queste compaiono nel testo. In seguito, provate una visualizzazione più interessante (e utile) in cui le parole siano ordinate alfabeticamente. Per esempio, le righe

```
To be, or not to be: that is the question:
Whether 'tis nobler in the mind to suffer
```

contengono la parola "to" tre volte, "be" due volte, "or" una, ecc.

8.35 *(Elaborazione dei testi)* L'approfondimento dedicato alla manipolazione delle stringhe in questo libro è da attribuire in gran parte all'emozionante sviluppo dell'elaborazione dei testi negli anni recenti. Una funzione importante per i sistemi di elaborazione testi è la *giustificazione tipografica*: l'allineamento delle parole al margine sinistro e a quello destro della pagina. La giustificazione tipografica genera un aspetto professionale del documento, dando l'impressione che sia stato impostato in tipografia piuttosto che preparato su una macchina per scrivere. Sui computer la giustificazione tipografica può essere realizzata inserendo uno o più caratteri di spazio tra ognuna delle parole presenti in una riga, in modo che quella più a destra sia allineata con il margine destro.

Scrivete un programma che legga diverse righe di testo e le visualizzi con una giustificazione tipografica. Supponete che il testo debba essere stampato su un foglio largo 8 pollici e mezzo (215,9 millimetri) e che, a sinistra e a destra della pagina stampata, debba essere lasciato un margine di un pollice (25,4 millimetri). Supponete anche che il computer stampi 10 caratteri per pollice. Di conseguenza, il vostro programma dovrà stampare 6 pollici e mezzo di testo (165,1 millimetri), ovverosia 65 caratteri per riga.

8.36 *(Visualizzare le date in vari formati)* Nella corrispondenza commerciale le date sono visualizzate di solito in molti formati. Due dei formati più comuni sono:

```
21/07/55 e 21 luglio 1955
```

Scrivete un programma che legga una data nel primo formato e la visualizzi nel secondo.

8.37 *(Protezione degli assegni)* Molto spesso i computer sono impiegati in sistemi per la compilazione degli assegni, come le applicazioni per la gestione degli stipendi e per la contabilità del debito. Circolano molte strane storie secondo le quali sarebbero stati stampati (erroneamente) degli assegni per la paga settimanale con cifre che superino il milione di dollari. Quelle strane cifre sono stampate dai

sistemi computerizzati per la compilazione degli assegni solo a causa di errori umani e/o di deterioramenti della macchina. I progettisti dei sistemi, naturalmente, fanno tutto il possibile per implementare nei propri sistemi dei controlli che prevengano l'emissione di assegni sbagliati.

Un altro grave problema è l'alterazione intenzionale dell'importo di un assegno da parte di qualcuno che intenda incassarlo in modo fraudolento. La maggior parte dei sistemi computerizzati per la compilazione degli assegni impiegano una tecnica chiamata *protezione degli assegni*, proprio per prevenire l'alterazione dei loro importi.

Gli assegni progettati per la compilazione da parte dei computer contengono un numero fisso di spazi in cui il computer può stampare l'importo. Supponete che l'assegno di uno stipendio contenga otto spazi bianchi nei quali si suppone che il computer debba stampare l'ammontare di una paga settimanale. Nel caso che l'importo sia sostanzioso, allora saranno riempiti tutti gli otto spazi. Per esempio:

```
1,230.60 (ammontare dell'assegno in dollari)
- - - - -
12345678 (numeri di posizione)
```

Invece, nel caso che la cifra sia inferiore ai $1000 allora molti degli spazi sarebbero lasciati in bianco. Per esempio,

```
   99.87
- - - - -
12345678
```

contiene tre spazi bianchi. Sarebbe sicuramente più facile alterare l'importo dell'assegno, qualora questo fosse stato stampato con degli spazi bianchi. Per prevenire l'alterazione di un assegno, molti sistemi per la loro compilazione inseriscono degli *asterischi iniziali* in modo da proteggere l'importo, come nell'esempio che segue:

```
***99.87
- - - - -
12345678
```

Scrivete un programma che prenda in input l'importo in dollari che dovrà essere stampato su un assegno e lo visualizzi nel formato protetto dagli asterischi iniziali, qualora siano necessari. Supponete che per stampare l'importo siano disponibili nove spazi.

8.38 *(Scrivere in lettere l'importo di un assegno)* Continuando la discussione dell'esempio precedente, ribadiamo quanto sia importante progettare dei sistemi per la compilazione degli assegni che prevengano l'alterazione del loro importo. Un metodo di sicurezza comune richiede che l'importo dell'assegno sia scritto in cifre e "dichiarato" anche in lettere. Qualcuno è in grado di alterare l'importo numerico di un assegno, ma è estremamente difficile cambiare l'importo espresso in lettere.

Molti sistemi computerizzati per la compilazione degli assegni non stampano l'importo in lettere. Probabilmente, la causa principale di questa omissione è che molti dei linguaggi di alto livello utilizzati nelle applicazioni commerciali non contengono delle caratteristiche adeguate per la manipolazione delle stringhe. Un'altra ragione è che la logica per scrivere in lettere gli importi degli assegni è alquanto complicata.

Scrivete un programma C che prenda in input l'ammontare numerico di un assegno e scriva il suo equivalente in lettere. Per esempio, l'importo 112.43 dovrebbe essere scritto come

```
ONE HUNDRED TWELVE and 43/100
```

8.39 *(Il codice Morse)* Il più famoso schema di codifica è probabilmente il codice Morse, sviluppato da Samuel Morse nel 1832 per il sistema telegrafico. Il codice Morse assegna una serie di punti e di linee a ogni lettera dell'alfabeto, a ogni numero e a pochi caratteri speciali (come il punto, la virgola, i

due punti e il punto e virgola). Nei sistemi acustici, il punto è rappresentato da un suono breve e la linea è rappresentata da uno lungo. Con i sistemi basati su segnali luminosi o sull'utilizzo di bandierine, sono utilizzate altre rappresentazioni di punti e linee.

La separazione tra le parole è indicata da uno spazio o, più semplicemente, dall'assenza di un punto o di una linea. Nei sistemi acustici, uno spazio è indicato da un breve periodo durante il quale non è trasmesso alcun suono. La versione internazionale del codice Morse è mostrata nella Figura 8.39.

Scrivete un programma che legga una frase in lingua italiana e la codifichi in codice Morse. Scrivete anche un programma che legga una frase in codice Morse e la converta nell'equivalente in lingua italiana. Utilizzate uno spazio tra le lettere del codice Morse e tre fra le parole.

Carattere	Codice	Carattere	Codice
A	.-	T	-
B	-...	U	..-
C	-.-.	V	...-
D	-..	W	.--
E	.	X	-..-
F	..-.	Y	-.--
G	--.	Z	--..
H	....		
I	..	**Numeri**	
J	.---	1	.----
K	-.-	2	..---
L	.-..	3	...--
M	--	4	....-
N	-.	5	.....
O	---	6	-....
P	.--.	7	--...
Q	--.-	8	---..
R	.-.	9	----.
S	...	0	-----

Figura 8.39 Le lettere dell'alfabeto espresse nel codice Morse internazionale.

8.40 *(Programma per la conversione metrica)* Scrivete un programma che assista l'utente con le conversioni metriche. Il vostro programma dovrà consentire all'utente di specificare con delle stringhe i nomi delle unità di misura (ovverosia, centimetri, litri, grammi, ecc. per il sistema metrico decimale e pollici, quarti di gallone, libbre, ecc. per il sistema anglosassone) e dovrà rispondere a semplici domande come:

```
"A quanti pollici corrispondono 2 metri?"
"A quanti litri corrispondono 10 quarti di gallone?"
```

Il vostro programma dovrà riconoscere le conversioni non valide. Per esempio, la domanda

```
"A quanti piedi corrispondono 5 chilogrammi?"
```

non ha senso perché i `"piedi"` sono un'unità di misura della lunghezza mentre il `"chilogrammo"` è una unità di misura del peso.

8.41 *(Lettere di sollecito)* Molte aziende commerciali spendono una gran quantità di tempo e di denaro per il recupero del credito. Il *sollecito* è appunto l'invio a un debitore di ripetute e insistenti richieste, nel tentativo di incassare un credito.

Spesso i computer sono utilizzati per generare automaticamente le lettere di sollecito, con un grado crescente di severità man mano che il credito invecchia. La teoria alla base di ciò è che un credito diventa più difficile da incassare man mano che invecchia e, di conseguenza, le lettere di sollecito devono diventare più minacciose.

Scrivete un programma C che contenga i testi di cinque lettere di sollecito con severità crescente. Il vostro programma dovrà accettare come input:

1. Il nome del debitore.
2. L'indirizzo del debitore.
3. Il conto del debitore.
4. La cifra dovuta.
5. L'età della cifra dovuta (ovverosia, un ritardo di un mese, di due mesi, ecc.).

Utilizzate l'età della cifra dovuta per selezionare uno dei cinque testi del messaggio e stampate la lettera di sollecito, inserendo in modo appropriato le altre informazioni fornite dall'utente.

Un progetto impegnativo per la manipolazione delle stringhe

8.42 *(Un generatore di cruciverba)* Molte persone hanno risolto almeno una volta un cruciverba, ma poche hanno tentato di generarne uno. Generare un cruciverba è un problema difficile. Lo proponiamo in questo contesto come un progetto per la manipolazione delle stringhe che richieda un ragionamento e uno sforzo sostanziosi. Il programmatore dovrà risolvere molti problemi anche per ottenere il più semplice programma per la generazione dei cruciverba. Per esempio, in che modo si potrà rappresentare la griglia di un cruciverba in un computer? È preferibile utilizzare una serie di stringhe o delle matrici? Il programmatore avrà bisogno di una fonte di parole (ovverosia, un dizionario computerizzato) cui il programma possa fare riferimento in modo diretto. In quale forma dovrebbero essere immagazzinate queste parole, per facilitare la complessa manipolazione richiesta dal programma? Il lettore molto ambizioso vorrà generare anche la porzione delle "definizioni" per il cruciverba, in cui siano stampati i brevi suggerimenti forniti al solutore per ogni parola "orizzontale" e "verticale". Anche stampare semplicemente una versione vuota del cruciverba non è un problema semplice.