# Esercizi di programmazione per il corso di Algoritmi*

## 1 Lezione 4

### Esercizi sull'esempio di laboratorio

**Esercizio 1** Si ammetta che un giocatore possa non avere mosse a disposizione, che di conseguenza passi il turno e che questo sia rappresentato nel file di testo da due trattini consecutivi (`--`). L'algoritmo deve ora verificare che la sospensione sia corretta, cioè che davvero non ci siano mosse possibili. Lo farà scorrendo tutte le caselle libere della scacchiera e, per ciascuna, verificando che tutte le direzioni sono non lecite per il giocatore. Si tratta quindi di quattro cicli annidati, due per le caselle e due per le direzioni. Se si trova una direzione lecita, si deve stampare un messaggio di errore e uscire[1].

**Esercizio 2** Si aggiunga ora anche una verifica finale che dopo aver eseguito tutte le mosse del file entrambi i giocatori siano privi di mosse disponibili. Ovviamente, la soluzione dell'esercizio precedente è particolarmente utile allo scopo.

**Esercizio 3** Per approfondire e chiarire il concetto di struttura dati astratta, si estraggano dal codice realizzato in precedenza le dichiarazioni di tipo, le costanti simboliche e le funzioni dedicate a gestire la scacchiera, in modo da costruire una libreria che consenta a qualsiasi programma di gestire una scacchiera senza conoscere nulla della sua effettiva implementazione[2].

**Esercizio 4** Si generi una procedura che per ciascun giocatore valuta la mossa migliore da eseguire in base al numero di pedine catturate e la esegue. In caso di parità si aggiunga un criterio di scelta qualsiasi (per esempio, l'ordine lessicografico). Si noti che questo criterio di scelta è scorretto, dato che in genere è più convienente avere poche pedine impossibili da catturare che tante soggette a cattura. Si possono utilizzare criteri più lungimiranti preferendo gli angoli alle caselle laterali, e queste a quelle interne.

*tratti o ispirati dal testo di K.N. King

[1] Una soluzione per l'esercizio è disponibile nel file `othello6.c`.

[2] Una soluzione per l'esercizio è disponibile nel file `othello7.c`, con la libreria `scacchiera.c` e l'intestazione `scacchiera.h`.

# Esercizi sui vettori (statici)

**Esercizio 1** Dato un numero intero `n`, si conti e si stampi a video il numero di volte che nella sua rappresentazione decimale compare ciascuna cifra da 0 a 9.

Suggerimento: Occorre un ciclo che generi le cifre calcolando i resti di successive divisioni del numero per 10 e un vettore di interi indicizzato con i numeri da 0 a 9 per contenere il numero di occorrenze di ciascuna cifra.

**Esercizio 2** Si aggiunga al codice precedente la costruzione di un vettore di interi, che contenga le sole cifre ripetute più volte nel numero dato.

Suggerimento: Questo vettore ha un numero di elementi variabile e non noto a priori. Tale numero è sicuramente $\leq 10$. Si definisca allora un vettore di 10 elementi, si usino solo i primi elementi per conservare le cifre e si usi una variabile intera per esprimere la lunghezza della porzione di vettore effettivamente usata.

**Esercizio 3** Si scriva un programma `MATRICI.C` che costruisca due matrici quadrate di ordine 10 assegnando al generico elemento $(i, j)$, rispettivamente, i valori $i + j$ e $i - j$. Quindi, il programma calcoli una terza matrice, prodotto delle prime due e due vettori che contengono, rispettivamente, la media aritmetica degli elementi di ciascuna riga e la media aritmetica degli elementi di ciascuna colonna. Si noti che ciò comporta una conversione da intero a reale.

**Esercizio 4** Si scriva un codice che calcola e conserva in un vettore i primi `N` numeri di Fibonacci (con $N \leq 40$ per evitare l'*overflow*), per poi stamparli a video. Si stampi a video anche la sequenza dei rapporti fra due numeri consecutivi (attenzione alla conversione da intero a reale e a non eccedere i limiti del vettore).

**Esercizio 5** Si scriva un codice che dichiara una scacchiera come matrice bidimensionale quadrata di caratteri composta da 8 righe e 8 colonne, e poi inizializza e stampa le scacchiere relative al gioco della dama e degli scacchi (codificando opportunamente le caselle vuote e quelle occupate dai vari pezzi).

# Esercizi sui record

**Esercizio 1** Si scriva un programma che definisce il tipo `rational` per rappresentare i numeri razionali, costituito da due campi `num` e `den` che conterranno il numeratore e il denominatore del numero stesso. Si definiscano funzioni che ricevono due numeri razionali e restituiscono la loro somma, differenza, prodotto, rapporto.

Nota: Questo esercizio si può complicare, chiedendo che, al termine delle operazioni, `num` e `den` siano semplificati sino a diventare primi fra loro. Per farlo, occorre calcolarne il massimo comun divisore, per esempio con l'algoritmo di Euclide.

**Esercizio 2** Si scriva un programma che definisce il tipo `complex` per rappresentare i numeri complessi, costituito da due campi `r` e `i` che conterranno la parte reale e immaginaria del numero stesso. Si definiscano funzioni che ricevono due numeri complessi e restituiscono la loro somma, differenza, prodotto.

**Esercizio 3** Si scriva un programma che definisce il tipo `data` per rappresentare le date, intese come terne giorno-mese-anno. Per il mese, si definisca un tipo enumerativo *ad hoc*. Si definiscano:

- una funzione che riceve una data e restituisce la posizione del giorno lungo l'anno (da 1 a 366)
- una funzione che riceve due date e restituisce $-1$ se la prima data precede la seconda, 0 se coincidono, $+1$ se la prima data segue la seconda

**Esercizio 4** Si scriva un programma che definisce il tipo `time` contenente tre campi `ora`, `minuto` e `secondo`. Si definisca una funzione che riceve un valore `long secondi_totali`, lo interpreta come il numero di secondi trascorsi dalla mezzanotte e lo traduce in una struttura di tipo `time`.

**Esercizio 5** Si scriva un programma che definisce il tipo `colore` contenente tre campi `R`, `G`, e `B`, corrispondenti all'intensità luminosa nel campo del rosso, del verde e del blu (valori compresi fra 0 e 255). Si definisca una funzione `Schiarisce()` che riceve un colore e restituisce un colore più chiaro, dividendo per 0.7 tutti i valori e arrotondandoli all'intero più vicino compreso fra 0 e 255.

**Esercizio 6** Si scriva un programma che definisce il tipo `point` contenente due campi interi `x` e `y` per rappresentare un punto su un piano e il tipo `rectangle` che rappresenta un rettangolo attraverso due campi `nordovest` e `sudest` che rappresentano i due punti estremi in alto a sinistra e in basso a destra. Si definiscano funzioni che:

- dati due punti, restituisca il rettangolo da loro individuato
- dato un rettangolo, ne restituisca l'area
- dato un rettangolo e un punto, restituisca un valore `boolean` (tipo enumerativo da definire) che indichi se il punto sta nel rettangolo (bordi compresi) o fuori

Come cambierebbero tali funzioni se i punti avessero coordinate reali?

**Esercizio 7** Si definiscano due tipi enumerativi: `pezzo`, con valori `RE, REGINA, TORRE, ALFIERE, CAVALLO, PEDONE, VUOTO` e `colore`, con valori `BIANCO, NERO, VUOTO`. Si definisca poi una struttura `casella`, che rappresenta una casella del gioco degli scacchi, componendo i due tipi precedenti nonché la riga e la colonna. Infine, si definisca un tipo `scacchiera` come matrice quadrata di caselle. Si scriva un programma con una funzione che inizializza le posizioni dei pezzi su una scacchiera e un'altra funzione che stampa a video la scacchiera con le posizioni correnti.