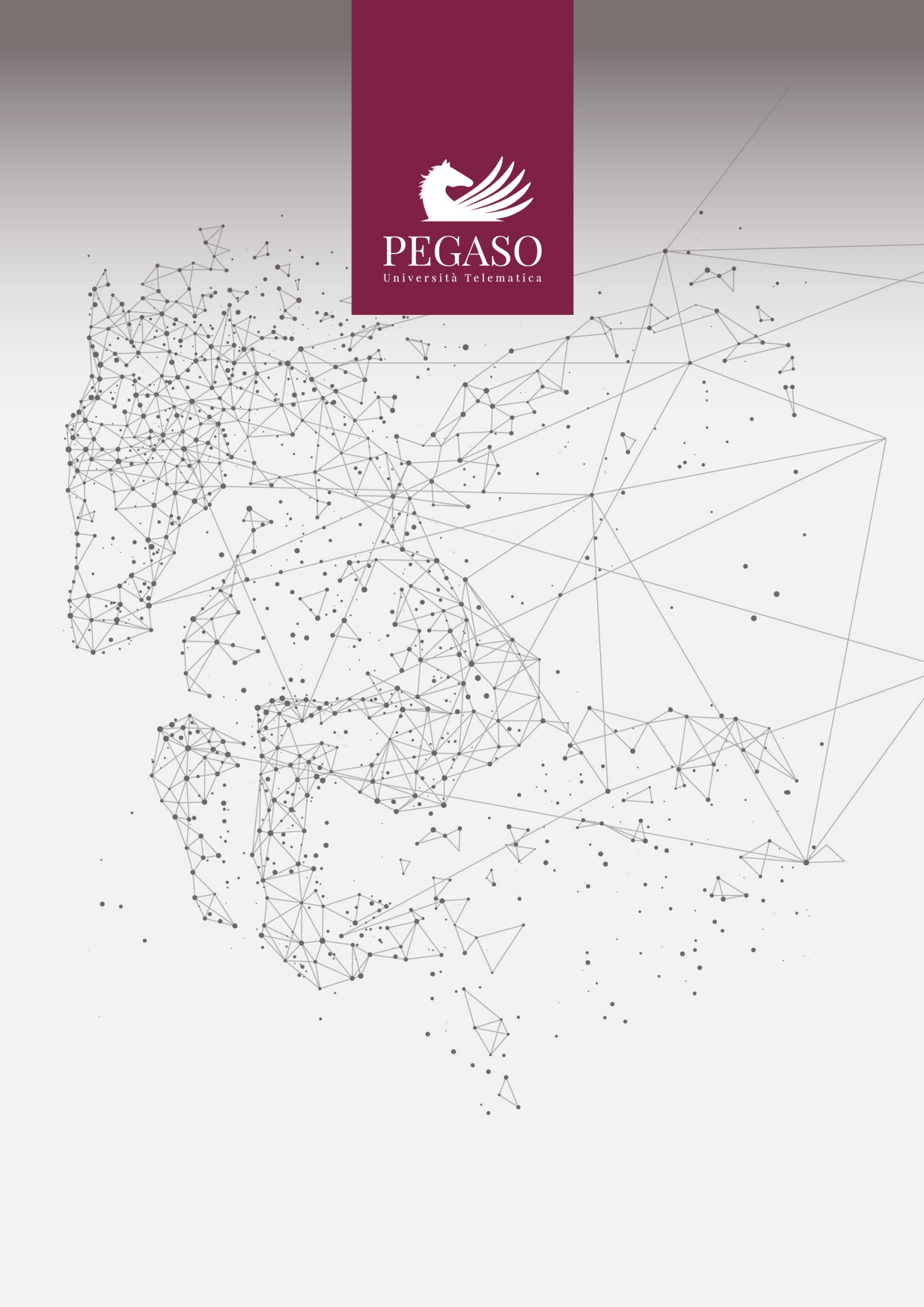

PEGASO
Università Telematica

Indice

1. Stringhe di qualsiasi lunghezza

Gli array possono contenere puntatori.

Un uso comune di un array di puntatori è quello di realizzare un array di stringhe.

Ogni elemento nell'array è una stringa, ma in C una stringa è essenzialmente un puntatore al suo primo carattere.

Pertanto, ogni elemento in un array di stringhe è in realtà un puntatore al primo carattere di una stringa.

Considerate la definizione dell'array di stringhe `suit`, che potrebbe essere utile per rappresentare un mazzo di carte da gioco.

```
const char *suit[4] = {"Hearts", "Diamonds", "Clubs", "Spades"};
```

La parte `suit[4]` della definizione indica un array di 4 elementi.

La parte `char *` della dichiarazione indica che ogni elemento dell'array `suit` è del tipo "puntatore a char".

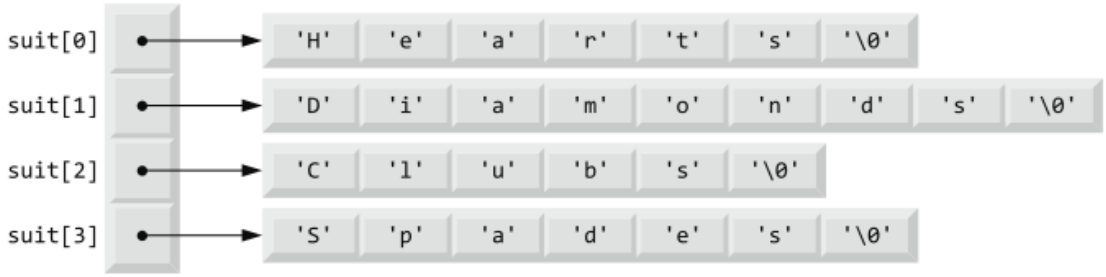
Il qualificatore const indica che le stringhe puntate da ogni elemento non saranno modificate.

I quattro valori da inserire nell'array sono "Hearts", "Diamonds", "Clubs" e "Spades".

Ognuno è memorizzato in memoria come una stringa di caratteri terminata da null, che è di un carattere più lunga del numero dei caratteri tra le virgolette.

Le quattro stringhe sono lunghe, rispettivamente, 7, 9, 6 e 7 caratteri.

Sebbene sembri che queste stringhe vengano inserite nell'array `suit`, di fatto solo i puntatori sono memorizzati nell'array.



Ogni puntatore punta al primo carattere della stringa corrispondente.

Così, anche se l'array `suit` è fisso come dimensione, fornisce l'accesso a stringhe di caratteri di qualsiasi lunghezza.

Questa flessibilità è un esempio delle potenti capacità di strutturazione dei dati del linguaggio C.

I semi delle carte da gioco avrebbero potuto essere rappresentati con un array bidimensionale, nel quale ogni riga avrebbe rappresentato un seme e ogni colonna una lettera del nome di un seme.

Una tale struttura di dati avrebbe un numero fisso di colonne per riga e quel numero dovrebbe essere grande tanto quanto la stringa più grande.

Potrebbe dunque andare sprecata una memoria considerevole quando si memorizza un grande numero di stringhe di cui la maggior parte è più breve della stringa più lunga.

Nel prossimo paragrafo useremo array di stringhe per rappresentare un mazzo di carte.

2. Caso pratico: mescolare le carte e simularne la distribuzione

In questo paragrafo usiamo la generazione di numeri casuali per sviluppare un programma che mescola le carte e ne simula la distribuzione.

Questo programma può quindi essere usato per implementare programmi per specifici giochi di carte.

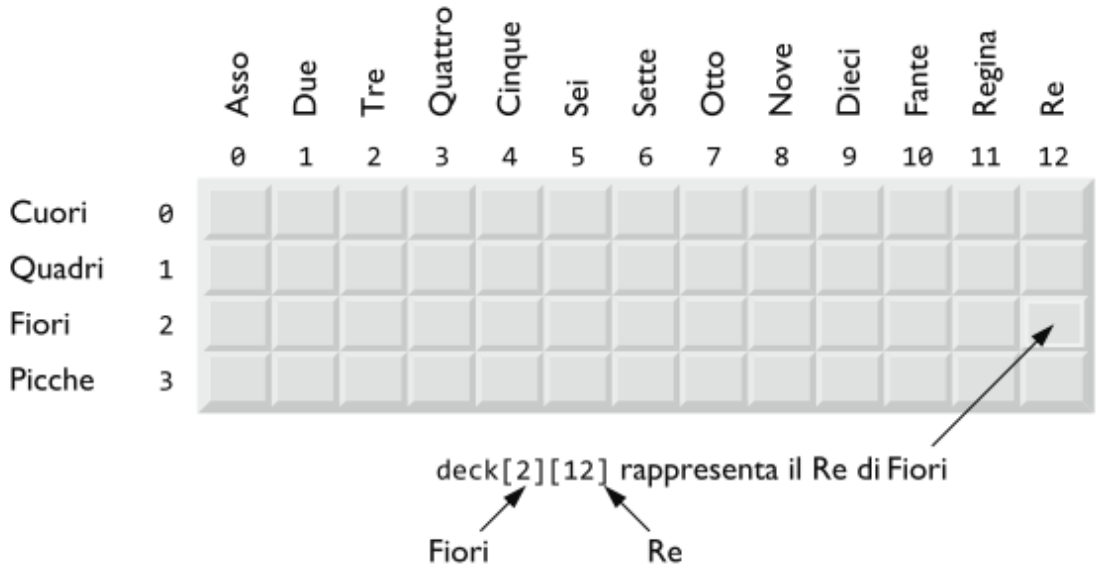
Per evidenziare alcuni sottili problemi di prestazioni, abbiamo intenzionalmente usato algoritmi subottimi per mescolare e distribuire le carte.

Usando l'approccio top-down con affinamento graduale, svilupperemo un programma che mescola un mazzo di 52 carte da gioco e poi le distribuisce.

L'approccio top-down è particolarmente utile per affrontare problemi grandi e complessi.

Rappresentare un mazzo di carte come array bidimensionale

Usiamo l'array `deck` bidimensionale 4 per 13 per rappresentare il mazzo di carte da gioco.



Le righe corrispondono ai semi: la riga 0 corrisponde ai cuori, la riga 1 ai quadri, la riga 2 ai fiori e la riga 3 alle picche.

Le colonne corrispondono ai valori relativi alle figure delle carte: le colonne da 0 a 9 corrispondono, rispettivamente, alle figure dall'asso al dieci e le colonne da 10 a 12 corrispondono al fante, alla regina e al re.

Caricheremo l'array di stringhe `suit` con le stringhe di caratteri che rappresentano i quattro semi, e l'array di stringhe face con stringhe che rappresentano i tredici valori delle figure delle carte.

Mescolare l'array bidimensionale

Questo mazzo di carte simulato può essere mescolato come segue.

Dapprima l'array viene azzerato.

Quindi, vengono scelte a caso una riga (row: 0–3) e una colonna (column: 0–12).

Viene inserito il numero 1 nell'elemento dell'array `deck[row][column]` per indicare che questa carta sarà la prima del mazzo mescolato a essere distribuita.

Questo processo continua con i numeri 2, 3, ..., 52 che vengono inseriti a caso nell'array deck per indicare quali carte sono da porre come seconda, terza, ... e cinquantaduesima nel mazzo mescolato.

Quando l'array `deck` comincia a essere pieno di numeri di carte, è possibile che una carta

venga selezionata di nuovo, cioè, quando viene selezionato, `deck[row][column]` risulta già diverso da zero.

Questa selezione viene semplicemente ignorata e si scelgono ripetutamente a caso altre righe e colonne, finché non viene trovata una carta non ancora selezionata.

Alla fine, i numeri da 1 a 52 occupano i 52 spazi dell'array `deck`.

A questo punto il mazzo di carte è completamente mescolato.

Possibilità di posposizione indefinita

Questo algoritmo per mescolare le carte può venire eseguito indefinitamente se le carte che sono già state mescolate vengono ripetutamente selezionate a caso.

Questo fenomeno è noto come posposizione indefinita.

Prestazioni

A volte un algoritmo che viene individuato in un modo "naturale" può presentare sottili problemi di prestazioni, come la posposizione indefinita.

Cercate algoritmi che la evitino.

Distribuire le carte dall'array bidimensionale

Per distribuire la prima carta cerchiamo nell'array l'elemento `deck[row][column]` uguale a 1.

Ciò si realizza con istruzioni for annidate che fanno variare `row` da 0 a 3 e `column` da 0 a 12.

A quale carta corrisponde quell'elemento dell'array?

L'array `suit` è stato precaricato con i quattro semi, per cui, per ottenere il seme, stampiamo la stringa di caratteri `suit[row]`.

In modo simile, per ottenere il valore relativo alla figura della carta, stampiamo la stringa di caratteri face`[column]`.

Stampiamo anche la stringa di caratteri " of ".

La stampa di queste informazioni nell'ordine corretto ci permette di stampare ogni carta nella forma "King of Clubs", "Ace of Diamonds" e così via.

Sviluppare la logica del programma con il processo top-down di affinamento graduale

Procediamo con il processo top-down di affinamento graduale.

Il top è semplicemente

Mescola e distribuisci le 52 carte

Il nostro primo affinamento produce:

Inizializza l'array dei semi

Inizializza l'array delle figure

Inizializza l'array del mazzo

Mescola il mazzo

Distribuisci le 52 carte

"Mescola il mazzo" può essere espanso come segue:

Per ognuna delle 52 carte

Inserisci il numero d'ordine della carta in un elemento non occupato

e selezionato a caso dell'array del mazzo

"Distribuisci le 52 carte" può essere espanso come segue:

Per ognuna delle 52 carte

Trova il numero d'ordine della carta nell'array del mazzo

e stampa la figura e il seme della carta

Accorpando queste espansioni si ottiene il nostro secondo affinamento completo:

Inizializza l'array dei semi

Inizializza l'array delle figure

Inizializza l'array del mazzo

Per ognuna delle 52 carte

Inserisci il numero d'ordine della carta in un elemento non occupato

e selezionato a caso dell'array del mazzo

Per ognuna delle 52 carte

Trova il numero d'ordine della carta nell'array del mazzo

e stampa la figura e il seme della carta

"Inserisci il numero d'ordine della carta in un elemento non occupato e selezionato a caso dell'array del mazzo" può essere espanso come:

Scegli a caso un elemento dell'array del mazzo

Finché l'elemento scelto dell'array del mazzo risulta già precedentemente scelto

Scegli a caso un elemento dell'array del mazzo

Inserisci il numero d'ordine della carta nell'elemento scelto dell'array del mazzo

"Trova il numero d'ordine della carta nell'array del mazzo e stampa la figura e il seme della carta" può essere espanso come:

Per ogni elemento dell'array del mazzo

Se l'elemento contiene il numero d'ordine della carta

Stampa la figura e il seme della carta

Accorpando queste espansioni si ottiene il nostro terzo affinamento:

Inizializza l'array dei semi

Inizializza l'array delle figure

Inizializza l'array del mazzo

Per ognuna delle 52 carte

Scegli a caso un elemento dell'array del mazzo

Finché l'elemento scelto dell'array del mazzo risulta già precedentemente scelto

Scegli a caso un elemento dell'array del mazzo

Inserisci il numero d'ordine della carta nell'elemento scelto dell'array del mazzo

Per ognuna delle 52 carte

Per ogni elemento dell'array del mazzo

Se l'elemento contiene il numero d'ordine della carta

Stampa la figura e il seme della carta

Questo completa il processo di affinamento.

Questo programma risulta più efficiente se le porzioni dell'algoritmo che mescolano e distribuiscono le carte sono combinate in modo che ogni carta venga distribuita non appena è inserita nel mazzo.

Abbiamo scelto di programmare queste operazioni separatamente, perché normalmente le carte vengono distribuite dopo, e non durante, il loro mescolamento.

Implementare il programma che mescola e distribuisce le carte

Il programma che mescola e distribuisce le carte è mostrato di seguito con un esempio di esecuzione.

Si usa lo specificatore di conversione `%s` nelle chiamate a `printf` per stampare stringhe di caratteri.

L'argomento corrispondente nella chiamata a `printf` deve essere un puntatore a `char` (o a un array `char`).

La specificazione di formato `"%5s of %-8s"` (riga 68) stampa una stringa di caratteri allineata a destra in un campo di cinque caratteri seguita da " of " e da una stringa di caratteri allineata a sinistra in un campo di otto caratteri.

Il segno meno in `%-8s` significa allineamento a sinistra.

C'è un punto di debolezza nell'algoritmo per distribuire le carte.

Una volta che viene localizzata la carta cercata, le due istruzioni `for` interne continuano ancora la ricerca nei restanti elementi del mazzo.

```
1  // Programma per
2  // Mescolare e distribuire le carte.
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <time.h>
6
7  #define SUITS 4
8  #define FACES 13
9  #define CARDS 52
10
11 // prototipi
12 void shuffle(unsigned int wDeck[][FACES]);
13 void deal(unsigned int wDeck[][FACES], const char *wFace[],
14    const char *wSuit[]);
15
16 int main(void)
17 {
18    // inizializza l'array deck
19    unsigned int deck[SUITS][FACES] = {0};
20
21    srand(time(NULL)); // seme per i numeri casuali
22    shuffle(deck); // mescola il mazzo
23
24    // inizializza l'array suit
25    const char *suit[SUITS] =
26       {"Hearts", "Diamonds", "Clubs", "Spades"};
27
28    // inizializza l'array face
29    const char *face[FACES] =
```

```
30          {"Ace", "Deuce", "Three", "Four",
31           "Five", "Six", "Seven", "Eight",
32           "Nine", "Ten", "Jack", "Queen", "King"};
33
34       deal(deck, face, suit); // distribuisci il mazzo
35    }
36
37    // mescola le carte nel mazzo
38    void shuffle(unsigned int wDeck[][FACES])
39    {
40       // per ogni carta, scegli a caso un elemento dell'array
41       for (size_t card = 1; card <= CARDS; ++card) {
42          size_t row; // numero della riga
43          size_t column; // numero della colonna
44
45          //  scegli una locazione a caso non occupata
46          do {
47             row = rand() % SUITS;
48             column = rand() % FACES;
49          } while(wDeck[row][column] != 0);
50
51          // inserisci numero d'ordine carta nell'elemento scelto
52          wDeck[row][column] = card;
53       }
54    }
55
56    // distribuisci le carte nel mazzo
57    void deal(unsigned int wDeck[][FACES], const char *wFace[],
58       const char *wSuit[])
59    {
60       // distribuisci ognuna delle carte
61       for (size_t card = 1; card <= CARDS; ++card) {
```

```
62          // effettua un ciclo lungo le righe di wDeck
63          for (size_t row = 0; row < SUITS; ++row) {
64             //  effettua un ciclo lungo le colonne di wDeck
65             for (size_t column = 0; column < FACES; ++column) {
66                //  se l'elemento contiene la carta corrente
67                if (wDeck[row][column] == card) {
68                   printf("%5s of %-8s%c", wFace[column],
                        wSuit[row],
69                         card % 2 == 0 ? '\n' : '\t'); // 2 colonne
70                }
71             }
72          }
73       }
74    }
```

```
Nine of Hearts          Five of Clubs
Queen of Spades         Three of Spades
Queen of Hearts           Ace of Clubs
 King of Hearts           Six of Spades
 Jack of Diamonds        Five of Spades
Seven of Hearts          King of Clubs
Three of Clubs          Eight of Hearts
Three of Diamonds        Four of Diamonds
Queen of Diamonds        Five of Diamonds
Six of Diamonds         Five of Hearts
Ace of Spades            Six of Hearts
Nine of Diamonds        Queen of Clubs
Eight of Spades           Nine of Clubs
Deuce of Clubs             Six of Clubs
Deuce of Spades           Jack of Clubs
Four of Clubs           Eight of Clubs
Four of Spades          Seven of Spades
Seven of Diamonds        Seven of Clubs
```

```
King of Spades           Ten of Diamonds
Jack of Hearts           Ace of Hearts
Jack of Spades           Ten of Clubs
Eight of Diamonds       Deuce of Diamonds
Ace of Diamonds        Nine of Spades
Four of Hearts         Deuce of Hearts
King of Diamonds         Ten of Spades
Three of Hearts           Ten of Hearts
```

Riferimenti bibliografici

- Paul Deitel, Harvey Deitel, "Il linguaggio C – Fondamenti e tecniche di programmazione", Libro edito da Pearson Italia. Include anche utili esercizi di autovalutazione.