مبانی برنامه نویسی
به زبان سی

۱۴، ۱۶ و ۱۸ دی ۱۳۹۹

جلسه های ۲۶، ۲۷ و ۲۸

ملکی مجد

مباحث این هفته:

- ادامه استراکت
  - مثال بُر زدن کارت
- Enumeration

- آشنایی با زبان برنامه نویسی C++
  - مثال ساده جمع دو عدد
- کتابخانه های استاندارد C++
- تابع های inline
- پارامتر References
- Default Arguments
- عملگر ::
- Function Overloading
- مقدمه ای بر مفهوم شی و UML

10.6 typedef

- The keyword `typedef` provides a mechanism for creating synonyms (or aliases) for previously defined data types.
  - For example, the statement
    - `typedef struct card Card;`
    - defines the new type name `Card` as a synonym for type `struct card`.
  - For example, the following definition
    - ```
      typedef struct {
         char *face;
         char *suit;
      } Card;
      ```

  creates the structure type `Card` without the need for a separate `typedef` statement.
      ```

10.7 Example: High-Performance Card Shuffling and Dealing Simulation

- The program in Fig. 10.3 is based on the card shuffling and dealing simulation discussed in Chapter 7.
- The program represents the deck of cards as an array of structures.

- The declaration
  - `Card deck[ 52 ];`

  declares an array of 52 `Card` structures (i.e., variables of type `struct card`).

```
1  /* Fig. 10.3: fig10_03.c
2     The card shuffling and dealing program using structures */
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <time.h>
6
7  /* card structure definition */
8  struct card {
9     const char *face; /* define pointer face */
10    const char *suit; /* define pointer suit */
11 }; /* end structure card */
12
13 typedef struct card Card; /* new type name for struct card */
14
15 /* prototypes */
16 void fillDeck( Card * const wDeck, const char * wFace[],
17    const char * wSuit[] );
18 void shuffle( Card * const wDeck );
19 void deal( const Card * const wDeck );
20
21 int main( void )
22 {
```

initializes the `Card` array in order with Ace through King of each suit

Fig. 10.3 | High-performance card shuffling and dealing simulation. (Part 1 of 4.)

```
23      Card deck[ 52 ]; /* define array of Cards */
24
25      /* initialize array of pointers */
26      const char *face[] = { "Ace", "Deuce", "Three", "Four", "Five",
27         "Six", "Seven", "Eight", "Nine", "Ten",
28         "Jack", "Queen", "King"};
29
30      /* initialize array of pointers */
31      const char *suit[] = { "Hearts", "Diamonds", "Clubs", "Spades"};
32
33      srand( time( NULL ) ); /* randomize */
34
35      fillDeck( deck, face, suit ); /* load the deck with Cards */
36      shuffle( deck ); /* put Cards in random order */
37      deal( deck ); /* deal all 52 Cards */
38      return 0; /* indicates successful termination */
39   } /* end main */
40
41   /* place strings into Card structures */
42   void fillDeck( Card * const wDeck, const char * wFace[],
43      const char * wSuit[] )
44   {
45      int i; /* counter */
46
```

Fig. 10.3 | High-performance card shuffling and dealing simulation. (Part 2 of 4.)

```
47      /* loop through wDeck */
48      for ( i = 0; i <= 51; i++ ) {
49         wDeck[ i ].face = wFace[ i % 13 ];
50         wDeck[ i ].suit = wSuit[ i / 13 ];
51      } /* end for */
52   } /* end function fillDeck */
53
54   /* shuffle cards */
55   void shuffle( Card * const wDeck )
56   {
57      int i; /* counter */
58      int j; /* variable to hold random value between 0 - 51 */
59      Card temp; /* define temporary structure for swapping Cards */
60
61      /* loop through wDeck randomly swapping Cards */
62      for ( i = 0; i <= 51; i++ ) {
63         j = rand() % 52;
64         temp = wDeck[ i ];
65         wDeck[ i ] = wDeck[ j ];
66         wDeck[ j ] = temp;
67      } /* end for */
68   } /* end function shuffle */
69
```

This algorithm cannot suffer from indefinite postponement like the shuffling algorithm presented in Chapter 7.

Fig. 10.3 | High-performance card shuffling and dealing simulation. (Part 3 of 4.)

```
70  /* deal cards */
71  void deal( const Card * const wDeck )
72  {
73     int i; /* counter */
74
75     /* loop through wDeck */
76     for ( i = 0; i <= 51; i++ ) {
77        printf( "%5s of %-8s%s", wDeck[ i ].face, wDeck[ i ].suit,
78           ( i + 1 ) % 4 ? "  " : "\n" );
79     } /* end for */
80  } /* end function deal */
```

Fig. 10.3 | High-performance card shuffling and dealing simulation. (Part 4 of 4.)

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

Fig. 10.4 | Output for the high-performance card shuffling and dealing simulation.

10.10 Enumeration Constants

- C provides one final user-defined type called an enumeration.
- An enumeration, introduced by the keyword **`enum`**, is a set of integer enumeration constants represented by identifiers.
  - Values in an `enum` start with `0`, unless specified otherwise, and are incremented by `1`.

- For example, the enumeration
  - ```
    enum months {
        JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP,
        OCT, NOV, DEC };
    ```

    creates a new type, `enum months`, in which the identifiers are set to the integers `0` to `11`, respectively.
    ```