

Structures



Systems Programming

Structures

- Structures
- Typedef
- Declarations
- Using Structures with Functions

10.1 Introduction

• Structures

- A collection of related variables (aggregates) under one name.
 - Can contain variables of different data types.
- Commonly used to define records to be stored in files.
- When combined with pointers, structures can create linked lists, stacks, queues, and trees.

© 2007 Pearson Ed -All rights reserved.

Structures

Example 1:

```
struct player
```

```
{
```

```
    char *name;
```

```
    int num;
```

```
    char *team;
```

```
    char *pos;
```

```
} ; /* Don't forget this semicolon! */
```

Typedef Example

Example 2:

```
struct card
{
    const char *face;
    const char *suit;
};
typedef struct card Card;
```

`struct` introduces the definition for structure `card`.

- `card` is the structure name and is used to declare variables of the structure type.
- `card` contains two members of type `char *`
 - These members are `face` and `suit`.

© 2007 Pearson Ed -All rights reserved.

typedef

Another way::

```
typedef struct  
{  
    const char *face;  
    const char *suit;  
} Card;
```

...

```
Card deck[52];
```

© 2007 Pearson Ed -All rights reserved.

10.6 typedef

Example:

```
typedef struct Card *CardPtr;
```

or

```
Card *Cardptr;
```

- Defines a new type name `CardPtr` as a synonym for type `struct Card *`
- `typedef` does not create a new data type.
 - It only creates an alias.
- Capitalize the first letter of `typedef` names to emphasize that they are synonyms for other type names.

© 2007 Pearson Ed -All rights reserved.

10.2 Structure Definitions

- **struct** information
 - A **struct** cannot contain an instance of itself.
 - It can contain a member that is a pointer to the same structure type.
 - A structure definition does not reserve space in memory. Rather a **struct** creates a new data type used to define structure variables.
- Definitions
 - Defined like other variables:
`card oneCard, deck[52], *cPtr;`
 - Can use a comma separated list:

```
struct card {  
    char *face;  
    char *suit;  
} oneCard, deck[ 52 ], *cPtr;
```

© 2007 Pearson Ed -All rights reserved.

10.2 Structure Definitions

- Valid Operations
 - Assigning a structure to a structure of the same type.
 - Taking the address (&) of a structure
 - Accessing the members of a structure.
 - Using the `sizeof` operator to determine the size of a structure.

10.3 Initializing Structures

- **Initializer lists**

- **Example:**

- ```
card oneCard = { "Three", "Hearts" };
```

- **Assignment statements**

- **Example:**

- ```
card threeHearts = oneCard;
```

- **Could also define and initialize threeHearts as follows:**

- ```
card threeHearts;
```

- ```
threeHearts.face = "Three";
```

- ```
threeHearts.suit = "Hearts";
```

# 10.4 Accessing Members of Structures

- Accessing structure members

- The dot operator (.) {the structure member operator} is used to access a structure member via the structure variable name.

```
card myCard;
printf("%s", myCard.suit);
```

- The arrow operator (->) {the structure pointer operator} accesses a structure member via a pointer to the structure.

```
card *myCardPtr = &myCard;
printf("%s", myCardPtr->suit);
```

- myCardPtr->suit is equivalent to  
( \*myCardPtr ).suit

© 2007 Pearson Ed -All rights reserved.

# Structure member and pointer operators

```
1 /* Fig. 10.2: fig10_02.c
2 Using the structure member and
3 structure pointer operators */
4 #include <stdio.h>
5
6 /* card structure definition */
7 struct card {
8 char *face; /* define pointer face */
9 char *suit; /* define pointer suit */
10 }; /* end structure card */
11
12 int main(void)
13 {
14 struct card aCard; /* define one struct card variable */
15 struct card *cardPtr; /* define a pointer to a struct card */
16
17 /* place strings into aCard */
18 aCard.face = "Ace";
19 aCard.suit = "Spades";
```

Structure definition

Structure definition must end with semicolon

Dot operator accesses members of a structure

© 2007 Pearson Ed -All rights reserved.

# Structure member and pointer operators

```
20
21 cardPtr = &aCard; /* assign address of aCard to cardPtr */
22
23 printf("%s%s\n%s%s\n%s%s\n", aCard.face, " of ", aCard.suit,
24 cardPtr->face, " of ", cardPtr->suit,
25 (*cardPtr).face, " of ", (*cardPtr).suit);
26
27 return 0; /* indicates successful termination */
28
29 } /* end main */
```

Ace of Spades  
Ace of Spades  
Ace of Spades

Arrow operator accesses members  
of a structure pointer

© 2007 Pearson Ed -All rights reserved.

# 10.5 Using Structures with Functions

- Passing structures to functions
  - The entire structure can be passed.
  - Individual members of the structure can be passed.
  - For both cases, they are passed by value.
- To pass a structure by-reference
  - Pass the address of the structure variable.
- To pass arrays by-value
  - Create a structure with the array as a member and then pass the structure.

# A Structure Example

```
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 {
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
```

Each **card** has a face and a suit

**Card** is now an alias for  
**struct card**

© 2007 Pearson Ed -All rights reserved.

# A Structure Example

```
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
39 return 0; /* indicates successful termination */
40
41 } /* end main */
42
43 /* place strings into Card structures */
44 void fillDeck(Card * const wDeck, const char * wFace[],
45 const char * wSuit[])
46 {
47 int i; /* counter */
48
49 /* loop through wDeck */
50 for (i = 0; i <= 51; i++) {
51 wDeck[i].face = wFace[i % 13];
52 wDeck[i].suit = wSuit[i / 13];
53 } /* end for */
54
55 } /* end function fillDeck */
56
```

Constant pointer to  
modifiable array of **Cards**

Fills the deck by giving each  
**Card** a face and suit

© 2007 Pearson Ed -All rights reserved.



# A Structure Example

```
57 /* shuffle cards */
58 void shuffle(Card * const wDeck)
59 {
60 int i; /* counter */
61 int j; /* variable to hold random value between 0 - 51 */
62 Card temp; /* define temporary structure for swapping Cards */
63
64 /* Loop through wDeck randomly swapping Cards */
65 for (i = 0; i <= 51; i++) {
66 j = rand() % 52;
67 temp = wDeck[i];
68 wDeck[i] = wDeck[j];
69 wDeck[j] = temp;
70 } /* end for */
71
72 } /* end function shuffle */
73
74 /* deal cards */
75 void deal (const Card * const wDeck)
76 {
77 int i; /* counter */
78
79 /* Loop through wDeck */
80 for (i = 0; i <= 51; i++) {
81 printf("%5s of %-8s%c", wDeck[i].face, wDeck[i].suit,
82 (i + 1) % 2 ? '\t' : '\n');
83 } /* end for */
84
85 } /* end function deal */
```

Each card is swapped with another, random card, shuffling the deck

© 2007 Pearson Ed -All rights reserved.

# A Structure Example

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

# Summary

- Definition of structures in C
- Syntax details for declaring structs
- Initializing structs
- Typedef
- Structure member (.) and pointer -> operators
- Passing structures to functions
- A Structure Example