Structures
Structures

- Structures
- Typedef
- Declarations
- Using Structures with Functions
- Structure Example
Structures

- A collection of related variables aggregated 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.
Example 1:

```c
struct player
{
    char *name;
    int num;
    char *team;
    char *pos;
};

/* Don't forget the semicolon! */
```

structure tag

structure members
Example 1:
struct player
{
    char *name;
    int num;
    char *team;
    char *pos;
} player1, player2;
Example 2:

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

The new type `Card` is an alias for type `struct 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.
Another way to declare this!!

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

...  
Card deck[52];
10.6 typedef

Example:

typedef struct Card *CardPtr;

or

Card *Cardptr;

- Defines a new type name CardPtr as an alias 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.
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 self-referential structure**).
  - 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;
    ```
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 \texttt{sizeof} operator to determine the size of a structure.
10.3 Initializing Structures

- **Initializer lists**
  - Example:
    ```c
    struct card oneCard = { "Three", "Hearts" }; 
    ```

- **Assignment statements**
  - Example:
    ```c
    struct card threeHearts = oneCard; 
    ```
  - Could also define and initialize `threeHearts` as follows:
    ```c
    struct 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.
    ```c
    card myCard;
    printf( "%s", myCard.suit );
    ```
  - The arrow operator (\(\rightarrow\)) \{the structure pointer operator\} accesses a structure member via a pointer to the structure.
    ```c
    card *myCardPtr = &myCard;
    printf( "%s", myCardPtr->suit );
    ```
  - `myCardPtr->suit` is equivalent to \(( \ast myCardPtr ).suit\)
/* Fig. 10.2: fig10_02.c  
  Using the structure member and  
  structure pointer operators */ 
#include <stdio.h>

/* card structure definition */
struct card {
  char *face; /* define pointer face */
  char *suit; /* define pointer suit */
}; /* end structure card */

int main( void )
{
  struct card aCard; /* define one struct card variable */
  struct card *cardPtr; /* define a pointer to a struct card */

  /* place strings into aCard */
  aCard.face = "Ace";
  aCard.suit = "Spades";
}
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,  
                  cardPtr->face, " of ", cardPtr->suit,  
                  (*cardPtr).face, " of ", (*cardPtr).suit );  
24  
25    return 0; /* indicates successful termination */  
26  
27 } /* end main */  

Ace of Spades  
Ace of Spades  
Ace of Spades  
Ace of Spades  

Arrow operator accesses members of a structure pointer
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.
/* Fig. 10.3: fig10_03.c  
  The card shuffling and dealing program using structures */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/* card structure definition */
struct card {
    const char *face; /* define pointer face */
    const char *suit; /* define pointer suit */
}; /* end structure card */

typedef struct card Card; /* new type name for struct card */

/* prototypes */
void fillDeck( Card * const wDeck, const char * wFace[],
          const char * wSuit[] );
void shuffle( Card * const wDeck );
void deal( const Card * const wDeck );

int main( void )
{
    Card deck[ 52 ]; /* define array of Cards */
    /* initialize array of pointers */
    const char *face[] = { "Ace", "Deuce", "Three", "Four", "Five",
        "Six", "Seven", "Eight", "Nine", "Ten",
        "Jack", "Queen", "King"};
/* initialize array of pointers */
const char *suit[] = { "Hearts", "Diamonds", "Clubs", "Spades"};

srand( time( NULL ) ); /* randomize */

fillDeck( deck, face, suit ); /* load the deck with Cards */
shuffle( deck ); /* put Cards in random order */
deal( deck ); /* deal all 52 Cards */

return 0; /* indicates successful termination */

/* place strings into Card structures */

void fillDeck( Card * const wDeck, const char * wFace[],
               const char * wSuit[] )
{
    int i; /* counter */

    /* loop through wDeck */
    for ( i = 0; i <= 51; i++ ) {
        wDeck[ i ].face = wFace[ i % 13 ];
        wDeck[ i ].suit = wSuit[ i / 13 ];
    } /* end for */

} /* end function fillDeck */
/* shuffle cards */
void shuffle( Card * const wDeck )
{
    int i;     /* counter */
    int j;    /* variable to hold random value between 0 - 51 */
    Card temp; /* define temporary structure for swapping Cards */

    /* loop through wDeck randomly swapping Cards */
    for ( i = 0; i <= 51; i++ ) {
        j = rand() % 52;
        temp = wDeck[ i ];
        wDeck[ i ] = wDeck[ j ];
        wDeck[ j ] = temp;
    } /* end for */
}
/* end function shuffle */

/* deal cards */
void deal( const Card * const wDeck )
{
    int i;     /* counter */

    /* loop through wDeck */
    for ( i = 0; i <= 51; i++ ) {
        printf( "%5s of %-8s%c", wDeck[ i ].face, wDeck[ i ].suit,
            ( i + 1 ) % 2 ? 't' : '\n' );
    } /* end for */
} /* end function deal */
A Structure Example

<table>
<thead>
<tr>
<th>Four of Clubs</th>
<th>Three of Hearts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Three of Diamonds</td>
<td>Three of Spades</td>
</tr>
<tr>
<td>Four of Diamonds</td>
<td>Ace of Diamonds</td>
</tr>
<tr>
<td>Nine of Hearts</td>
<td>Ten of Clubs</td>
</tr>
<tr>
<td>Three of Clubs</td>
<td>Four of Hearts</td>
</tr>
<tr>
<td>Eight of Clubs</td>
<td>Nine of Diamonds</td>
</tr>
<tr>
<td>Deuce of Clubs</td>
<td>Queen of Clubs</td>
</tr>
<tr>
<td>Seven of Clubs</td>
<td>Jack of Spades</td>
</tr>
<tr>
<td>Ace of Clubs</td>
<td>Five of Diamonds</td>
</tr>
<tr>
<td>Ace of Spades</td>
<td>Five of Clubs</td>
</tr>
<tr>
<td>Seven of Diamonds</td>
<td>Six of Spades</td>
</tr>
<tr>
<td>Eight of Spades</td>
<td>Queen of Hearts</td>
</tr>
<tr>
<td>Five of Spades</td>
<td>Deuce of Diamonds</td>
</tr>
<tr>
<td>Queen of Spades</td>
<td>Six of Hearts</td>
</tr>
<tr>
<td>Queen of Diamonds</td>
<td>Seven of Hearts</td>
</tr>
<tr>
<td>Jack of Diamonds</td>
<td>Nine of Spades</td>
</tr>
<tr>
<td>Eight of Hearts</td>
<td>Five of Hearts</td>
</tr>
<tr>
<td>King of Spades</td>
<td>Six of Clubs</td>
</tr>
<tr>
<td>Eight of Diamonds</td>
<td>Ten of Spades</td>
</tr>
<tr>
<td>Ace of Hearts</td>
<td>King of Hearts</td>
</tr>
<tr>
<td>Four of Spades</td>
<td>Jack of Hearts</td>
</tr>
<tr>
<td>Deuce of Hearts</td>
<td>Jack of Clubs</td>
</tr>
<tr>
<td>Deuce of Spades</td>
<td>Ten of Diamonds</td>
</tr>
<tr>
<td>Seven of Spades</td>
<td>Nine of Clubs</td>
</tr>
<tr>
<td>King of Clubs</td>
<td>Six of Diamonds</td>
</tr>
<tr>
<td>Ten of Hearts</td>
<td>King of Diamonds</td>
</tr>
</tbody>
</table>
Review of Structure

- 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