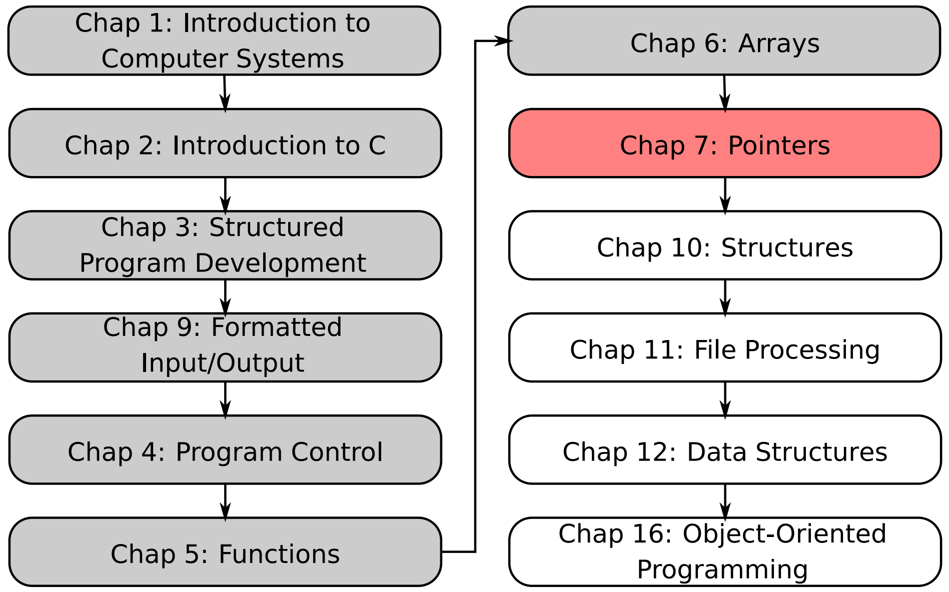
Computer Programming 143 – Lecture 22
Pointers V

Electrical and Electronic Engineering Department
University of Stellenbosch

Corné van Daalen
Thinus Booysen
Willie Smit
Willem Jordaan

Module Overview


Chap 1: Introduction to Computer Systems
Chap 2: Introduction to C
Chap 3: Structured Program Development
Chap 9: Formatted Input/Output
Chap 4: Program Control
Chap 5: Functions
Chap 6: Arrays
Chap 7: Pointers
Chap 10: Structures
Chap 11: File Processing
Chap 12: Data Structures
Chap 16: Object-Oriented Programming

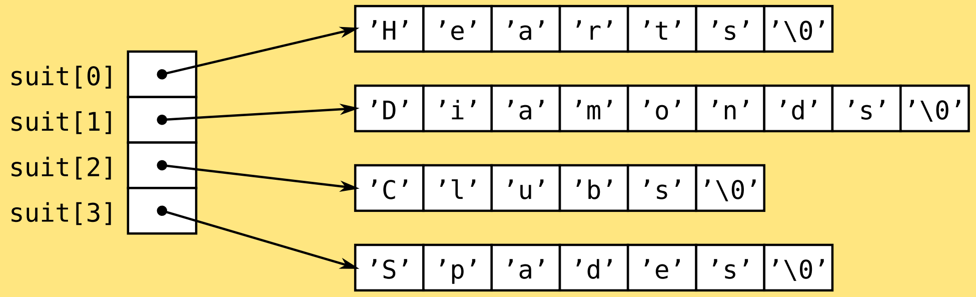
Lecture Overview

7.10 Arrays of Pointers

Array of strings

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

- Declares an array of 4 pointers to type char
- Initialises 4 strings and assigns the addresses of the first characters to the 4 pointers


suit[0]
suit[1]
suit[2]
suit[3]
'H' 'e' 'a' 'r' 't' 's' '\0'
'D' 'i' 'a' 'm' 'o' 'n' 'd' 's' '\0'
'C' 'l' 'u' 'b' 's' '\0'
'S' 'p' 'a' 'd' 'e' 's' '\0'

7.11 Card Shuffling and Dealing I

Problem statement

Design and implement an algorithm that shuffles and deals a 52-card deck

7.11 Card Shuffling and Dealing II

Deck representation

```
int deck[ 4 ][ 13 ];
```

		Ace	Two	Three	Four	Five	Six	Seven	Eight	Nine	Ten	Jack	Queen	King
		0	1	2	3	4	5	6	7	8	9	10	11	12
Hearts	0													
Diamonds	1													
Clubs	2													
Spades	3													

deck[2][12] represents the King of Clubs

The value stored in deck[2][12], represents the card's unique position in the deck

7.11 Card Shuffling and Dealing III

Empty Deck representation

		Ace	Two	Three	Four	Five	Six	Seven	Eight	Nine	Ten	Jack	Queen	King
		0	1	2	3	4	5	6	7	8	9	10	11	12
Hearts	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Diamonds	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Clubs	2	0	0	0	0	0	0	0	0	0	0	0	0	0
Spades	3	0	0	0	0	0	0	0	0	0	0	0	0	0

7.11 Card Shuffling and Dealing IV

Ordered Deck representation

		Ace	Two	Three	Four	Five	Six	Seven	Eight	Nine	Ten	Jack	Queen	King
		0	1	2	3	4	5	6	7	8	9	10	11	12
Hearts	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Diamonds	1	14	15	16	17	18	19	20	21	22	23	24	25	26
Clubs	2	27	28	29	30	31	32	33	34	35	36	37	38	39
Spades	3	40	41	42	43	44	45	46	47	48	49	55	51	52

7.11 Card Shuffling and Dealing V

Shuffled Deck representation

		Ace	Two	Three	Four	Five	Six	Seven	Eight	Nine	Ten	Jack	Queen	King
		0	1	2	3	4	5	6	7	8	9	10	11	12
Hearts	0	26	51	16	44	36	15	50	46	3	38	52	40	10
Diamonds	1	11	19	34	27	1	20	47	21	25	4	32	18	31
Clubs	2	6	35	8	9	37	2	49	29	22	13	5	42	33
Spades	3	12	7	23	45	24	41	28	14	39	30	48	43	17

7.11 Card Shuffling and Dealing VI

Top-level pseudocode

```
Shuffle and deal 52 cards
```

First refinement

```
Initialise the constant array of suit names
Initialise the constant array of face names
Initialise the deck array of card positions
Shuffle the deck
Deal 52 cards
```

7.11 Card Shuffling and Dealing VII

Second refinement

```
Initialise the constant array of suit names
Initialise the constant array of face names
Initialise the deck array of card positions

For each of the 52 ordinal positions in the empty deck
    Place ordinal position number in randomly selected unoccupied slot of deck

For each of the 52 ordinal positions in the shuffled deck
    Find ordinal position number in deck array and print face and suit of card
```

7.11 Card Shuffling and Dealing VIII

Third refinement

```
Initialise the suit array
Initialise the face array
Initialise the deck array

For each of the 52 ordinal positions in the empty deck
    Do
        Choose slot of deck randomly
    While slot of deck is not empty
    Place card number in chosen slot of deck

For each of the 52 ordinal positions in the shuffled deck
    For each slot of deck array
        If slot contains desired card number
            Print the face and suit of the card
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
// prototypes
void shuffle( int wDeck[][ 13 ] );
void deal( const int wDeck[][ 13 ], const char *wFace[],
           const char *wSuit[] );

int main( void )
{
   // initialise suit array
   const char *suit[ 4 ] = { "Hearts", "Diamonds", "Clubs", "Spades" };
   // initialise face array
   const char *face[ 13 ] = { "Ace", "Deuce", "Three", "Four", "Five",
      "Six", "Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King" };
   // initialise deck array
   int deck[ 4 ][ 13 ] = {{ 0 }};
   srand( time( 0 ) ); // seed random-number generator
   shuffle( deck ); // shuffle the deck
   deal( deck, face, suit ); // deal the deck
   return 0; // indicates successful termination
} // end main
```

```c
// shuffle cards in deck
void shuffle( int wDeck[][ 13 ] )
{
   int row; // row number
   int column; // column number
   int card; // counter

   // for each of the 52 ordinal positions, choose slot of deck randomly
   for ( card = 1; card <= 52; card++ ) {
      // choose new random location until unoccupied slot found
      do {
         row = rand() % 4;
         column = rand() % 13;
      } while ( wDeck[ row ][ column ] != 0 ); // end do...while
      // place card number in chosen slot of deck
      wDeck[ row ][ column ] = card;
   } // end for
} // end function shuffle
```

```c
// deal cards in deck
void deal( const int wDeck[][ 13 ], const char *wFace[],
const char *wSuit[] )
{
   int card, row, column; // card, row & column counters
   // deal each of the 52 ordinal positions
   for ( card = 1; card <= 52; card++ ) {
      for ( row = 0; row <= 3; row++ ) { // loop rows
         for ( column = 0; column <= 12; column++ ) { // loop columns
            // if slot contains current card, display card
            if ( wDeck[ row ][ column ] == card ) {
               if(card % 2 == 0){ // 2-column format
                 printf("%5s of %-8s%c",wFace[column],wSuit[row],'\n');
               }
               else {
                 printf("%5s of %-8s%c",wFace[column],wSuit[row],'\t');
               } // end else
            } // end if
         } // end for
      } // end for
   } // end for
} // end function deal
```

Perspective

Today

Pointers V

- Arrays of pointers
- Design example: card shuffling and dealing

Next lecture

- Structures

Homework

1. Study Sections 7.10-7.11 in Deitel & Deitel
2. Do Exercises 7.16