Mid Term Exam Review

This document is compiled to review some main concepts for the coming exam. Exam will include all contents from Chapter 5 to Chapter 8.

Exercise 1 (Functions Review):

Reference: 5.11 Example: A Game of Chance

One of the most popular games of chance is a dice game known as "craps," which is played in casinos and back alleys throughout the world. The rules of the game are straightforward:

A player rolls two dice. Each die has six faces. These faces contain 1, 2, 3, 4, 5, and 6 spots. After the dice have come to rest, the sum of the spots on the two upward faces is calculated. If the sum is 7 or 11 on the first throw, the player wins. If the sum is 2, 3, or 12 on the first throw (called "craps"), the player loses (i.e., the "house" wins). If the sum is 4, 5, 6, 8, 9, or 10 on the first throw, then that sum becomes the player's "point." To win, you must continue rolling the dice until you "make your point." The player loses by rolling a 7 before making the point.

Create a simulation of this game in a c program.

Exercise 2 (Arrays and Pointer Review):

Case Study: Card Shuffling and Dealing Simulation

Use random number generation to develop a card shuffling and dealing simulation program. You can use a 2 dimensional array as shown in Fig 7.23 The top-down step wise refinement is given.

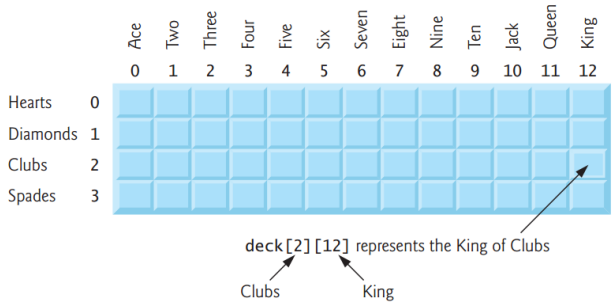

Fig. 7.23 | Double-subscripted array representation of a deck of cards.

This simulated deck of cards may be shuffled as follows. First the arraydeck is cleared to zeros. Then, a row(0–3) and a column(0–12) are each chosen at random. The number 1 is inserted in array element deck[row][column]to indicate that this card will be the first one dealt from the shuffled deck. This process continues with the numbers 2, 3, …, 52 being randomly inserted in the deckarray to indicate which cards are to be placed second, third, …, and fifty-second in the shuffled deck. As the deckarray begins to fill with card numbers, it's possible that a card will be

selected again—i.e., deck[row] [column]will be nonzero when it's selected. This selection is simply ignored and other rows and columns are repeatedly chosen at random until an unselected card is found. Eventually, the numbers 1 through 52 will occupy the 52 slots of the deckarray. At this point, the deck of cards is fully shuffled.

This shuffling algorithm can execute indefinitely if cards that have already been shuffled are repeatedly selected at random. This phenomenon is known as indefinite postponement. In this chapter's exercises, we discuss a better shuffling algorithm that eliminates the possibility of indefinite postponement.

To deal the first card, we search the array for deck[row][column]equal to 1. This is accomplished with nested for statements that vary row from 0 to 3 and column from 0 to 12. What card does that element of the array correspond to? The suit array has been preloaded with the four suits, so to get the suit, we print the character string suit[row]. Similarly, to get the face value of the card, we print the character string face[column]. We also print the character string " of ". Printing this information in the proper order enables us to print each card in the form "King of Clubs", "Ace of Diamonds" and so on. Let's proceed with the top-down, stepwise refinement process. The top is simply

Shuffle and deal 52 cards

Our first refinement yields:

Initialize the suit array
Initialize the face array
Initialize the deck array
Shuffle the deck
Deal 52 cards

"Shuffle the deck" may be expanded as follows:

For each of the 52 cards

Place card number in randomly selected unoccupied slot of deck

"Deal 52 cards" may be expanded as follows:

For each of the 52 cards

Find card number in deck array and print face and suit of card

Incorporating these expansions yields our complete second refinement:

Initialize the suit array
Initialize the face array
Initialize the deck array
For each of the 52 cards
Place card number in randomly selected unoccupied slot of deck
For each of the 52 cards
Find card number in deck array andprint face and suit of card

"Place card number in randomly selected unoccupied slot of deck" may be expanded as:

Choose slot of deck randomly
While chosen slot of deck has been previously chosen
Choose slot of deck randomly
Place card number in chosen slot of deck

"Find card number in deck array and print face and suit of card" may be expanded as:

For each slot of the deck array
If slot contains card number
Print the face and suit of the card
Incorporating these expansions yields our third refinement:
Initialize the suit array
Initialize the face array
Initialize the deck array
For each of the 52 cards
Choose slot of deck randomly
While slot of deck has been previously chosen
Choose slot of deck randomly
Place card number in chosen slot of deck
For each of the 52 cards
For each slot of deck array
If slot contains desired card number
Print the face and suit of the card

Exercise 3 (Characters and Strings Review):

(Tokenizing Telephone Numbers) Write a program that inputs a telephone number as a string in the form (555) 555-5555. The program should use function strtok to extract the area code as a token, the first three digits of the phone number as a token and the last four digits of the phone number as a token. The seven digits of the phone number should be concatenated into one string. The program should convert the area-code string to int and convert the phone-number string to long. Both the area code and the phone number should be printed.

Exercise 4 (Characters and Strings Review):

(Removing a Particular Word From a Given Line of Text) Write a program that inputs a line of text and a given word. The program should use string library functions strcmp and strcpy to remove all occurrences of the given word from the input line of text. The program should also count the number of words in the given line of text before and after removing the given word using the strtok function.

Exercise 5 (Characters and Strings Review):

(Printing Dates in Various Formats) Dates are commonly printed in several different for-mats in business correspondence. Two of the more common formats are 07/21/2003 and July 21, 2003. Write a program that reads a date in the first format and prints it in the second format.