Lab 10: Deck of Cards

CSci 515 Spring 2015

<2015-04-07 Tue>

Dates:

Due: In Lab, Wednesday April 8, by 4:05 pm (lab end time)

Objectives

- Get practice in declaring structs and enums user defined types.
- Become familiar with defining arrays of structs.
- Learn how to pass arrays of structs to functions and process the structs.

Description

In this lab we will create a basic data structure to represent an array of standard playing cards. A deck of playing cards contains 52 cards. There are 4 different suits in a standard deck of cards (HEARTS, DIAMONDS, CLUBS, SPADES). For each suite there is one card each of 13 possible face values (ACE, DEUCE, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING). You will define enumerated types to represent the possible suits and face values that cards can have. Then you need to define a `struct` to represent a single playing card. Each individual playing card has a single suit, and is of some particular face value.

Once you have defined your user defined types to represent a card, we will simulate the creation of a deck of cards. You will write a function that initializes an array of 52 Card structures to a standard deck of cards.

Perform the following tasks:

1. Define enumerated types called Suit and Face. Define each `enum` using the list of discrete possible values shown above.

2. Declare a `struct` called Card. Using your Suit and Face enumerated type, define the Card type to represent a single card.

3. Write a function called `initDeckOfCards`. This function will expect an array of Card structures as input. It will assume that the array has been initialized to hold exactly 52 cards, thus you do not need to pass in the size of this array to the function. The function should initialize the array of 52 cards correctly. (e.g. one of each possible Face/Suit combination in the deck).

4. In your main function, show an example of declaring an array of 52 Card structures. Demonstrate calling your init function to initialize your array of cards.

5. If you have time, and for extra credit, in main display the 52 resulting cards from your deck. Or even better, create a function called `displayDeckOfCards` that takes an array of cards and displays them.

NOTE: Now that our programs have more functions than just the `main()` function, the use of the function headers becomes meaningful and required. Make sure that all of your functions have function headers preceding them that document the purpose of the functions, and the input parameters and return value of the function.

Lab Submission

An eCollege dropbox has been created for this lab. You should upload your version of the lab by the end of lab time to the eCollege dropbox named `Lab 10 Deck of Cards`. Work submitted by the end of lab will be considered, but after the lab ends you may no longer submit work, so make sure you submit your best effort by the lab end time in order to receive credit.

Requirements and Grading Rubrics

Program Execution, Output and Functional Requirements

1. Your program must compile, run and produce some sort of output to be graded. 0 if not satisfied.

2. 33+ pts. You must declare the enumerated and struct types as specified.

3. 33+ pts. Your program must have the required initialization named function, that accepts the required input parameters and return the required values (if any). The function must initialize the array of cards correctly.

4. 33+ pts. You should demonstrate creating an array of your Card structures, and calling your initialization function in `main`.

5. 5+ extra credit pts. Display the resulting deck of cards on standard output. Use a function that takes the array of cards for the full 5 points, and displays strings for the face/suit (not integer values).

Program Style

Your programs must conform to the style and formatting guidelines given for this course. The following is a list of the guidelines that are required for the lab to be submitted this week.

1. The file header for the file with your name and program information and the function header for your main function must be present, and filled out correctly.

2. A function header must be present for all functions you define. You must document the purpose, input parameters and return values of all functions. Your function headers must be formatted exactly as shown in the style guidelines for the class.

3. You must indent your code correctly and have no embedded tabs in your source code. (Don't forget about the Visual Studio Format Selection command).

4. You must not have any statements that are hacks in order to keep your terminal from closing when your program exits (e.g. no calls to system()).

5. You must have a single space before and after each binary operator.

6. You must have a single blank line after the end of your declaration of variables at the top of a function, before the first code statement.

7. You must have a single blank space after , and ; operators used as a separator in lists of variables, parameters or other control structures.

8. You must have opening `{` and closing `}` for control statement blocks on their own line, indented correctly for the level of the control statement block.

9. All control statement blocks (if, for, while, etc.) must have `{ }` enclosing them, even when they are not strictly necessary (when there is only 1 statement in the block).

10. You should attempt to use meaningful variable and function names in your program, for program clarity. Of course, when required, you must name functions, parameters and variables as specified in the assignments. Variable and function names must conform to correct `camelCaseNameingConvention` .

Failure to conform to any of these formatting and programming practice guidelines for this lab will result in at least 1/3 of the points (33) for the assignment being removed for each guideline that is not followed (up to 3 before getting a 0 for the assignment). Failure to follow other class/textbook programming guidelines may result in a loss of points, especially for those programming practices given in our Deitel textbook that have been in our required reading so far.