War Game Documentation

The first design decision to be made was that of the suit array, which I decided to use a string (char array), rather than an int array with numbers.

Structs used to store information on each players hand and player number.

Printf and scanf is used to receive input from the user as they choose game options (new game, load game etc….).

if and else’s were used to determine whether the player count had been correctly chosen , setting valid to true, when the number entered was between 2 and 10 but not anything else.

If statement was used to determine the roundcount was or wasnt = 13 then enter the a new round as game is not complete before 13 rounds and when 13 rounds are complete it displays game is over and the final results of the game.

Do while is used to display users hand(which is done by a function where the users hand array is sent through and displayed) and for allowing them to choose a card. A const char pointer is used which is set to 99. And the do while continues to repeat unless a valid card is chosen as the choose card function returns a const char of 99 if its not valid causing the do while to repeat or else it returns the cardChosen by the user if it is found in the array using the strstr() scanning the array to find if it exists within the array. strcpy() is then used to remove it from the array(done using nested for loops to overwrite and replace array after the point which it has been matched.) once it has been matched to disallow the user from using the same card for the next round and then returns the card chosen to allow moving on from the do while.

Rand() method used to determine the random number between 13 to chose the random number in the array for the “computer” players to chose from. srand(time(NULL)); As well as #include <time.h> had to be implemented to allow this to work as it uses the time to generate the random numbers

The other “computer” players also use the chooseCard function the users card uses to determine if it is playable. Sending through their randomly generated number rather than the one the user selects.

All of the cards played by each player including the user are then stored into an array. The displayRound function is then used to output all the cards played so the user can see what cards the opposition have played by sending the array of cardsPlayed to the function along with the amount of players s the for loop knows how many times it has to run based on the players.

The highscores are then added up for the round and added to a highscores array, these are totalled up through a calculateHighscores function which is used to to run through the cards played array and determine the biggest card played, as well as adding the score of all the cards player. This is done using a switch statement as all the cards played are stored via string(in a char array) and cannot be compared as ints, so a new score array is added and based on the cardsUsed array card played it adds the score value to the array (e.g. if ‘J’ encounter in cards used array, 11 is added to score array and so on.) aswell as adding it to the roundScore (total score of all cards played). The scoreArray is then compared to the highestCard(which to begin with is 0), when a bigger number is encountered(which is guaranteed as smallest number is 2) , it is then set as the highest card and the players number ID is also saved as the value of k(used on for loop). The player with the highest card is displayed along with their id. Aswell as the total points accumulated for the round based on the cardsplayed. The roundScore and player ID are then returned to the main via a struct, which are then added to the high scores array using the highestscore id as the position in array and the roundScore is stored in the array as their score, this is then updated every round added to their score in the array to total their overall score and not overwriting it each round.

Round ends and options are then displayed to continue to next round, save, output game status or exit without saving.

Continue to next round repeats all of the above adding to totals and decrementing hands cards etc…

Save game saves the players hand, roundCount and card count to one file and the player points and playerCount to another file both using fprintf. This is done in the SaveGame function.

Output game, uses the function outputGameStatus to output the current status of the game showing the current scores of each player.

Exit game without saving allows the user to create a new game, load an existing game or exit application.

Load game uses function loadGame to read in the data stores in the savefile from the txt files and store them in variables which are stored in a struct sending them back to main via return of the struct.