Hacettepe University
Computer Engineering Department

Name and Surname : Berke Keser
Identity Number : Y21521778

Course : BBM203 Software Laboratory I
Experiment : Assignment 1
Subject : Array / Two-Dimensional Array
Due Date : 19.11.2020
Advisors : R.A. Feyza Nur KILIÇASLAN

e-mail : b21521778@cs.hacettepe.edu.tr

Main Program : Main.cpp

2 - Software Design Notes

2.1 Problem Definition

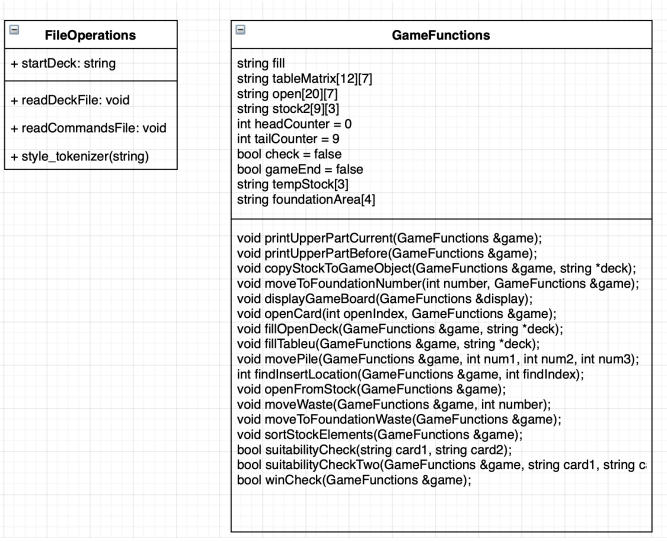
In the given assignment, we are expected to develop the classic Solitaire, also known as Klondike Solitaire. The game will be played with commands that read from the text file. In the same way, output will be printed out to text file. These files are should be taken as a program argument. In game board, we have 4 main area which are stock, waste, foundation area and tableau area. At the beginning of game, all these areas are blank. In commands file, we have different operations. While we are reading this commands line from commands.txt, we should fill these areas accordingly given operations. Additionally, we must obey the game rules for correct output file.

2.2 Solution

2.2.1

After analyzing game board, I decided two split game board into two part. First part is containing the upper part of game board that contains stock, waste and foundation area. Second part is containing tableau area. All these areas must contain cards in some point at the game and also these areas should be change to corresponding commands. For this reason, I conclude to hold tableau area and stock in two-dimensional array, for the rests which are waste and foundation area I decided to use one dimensional array. At the beginning of the game, we don't have any cards in the table board. Cards should be read from deck.txt file and they will deal according to game rules. After I checked deck.txt, I realized last card in the file should be first card in the tableau area. For this implementation, after reading deck file, I stored these cards in another array in reverse order. I think, the most important part of the problem is figure out what is happening in the stock area. For more detail, we have 24 cards in stock, we have to open from cards from stock and also need to shuffle cards when all of them are opened. Besides that, at some point of the game we have to print out card from the previous step.

2.2.2

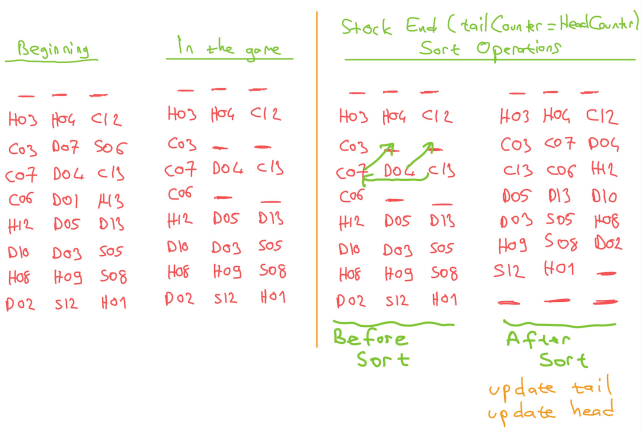


I have two main classes. There is no hierarchy between them. FileOperations class manage file operations. Its read data from given input text file. Game functions is manage all of the game process.

2.2.3

I use one-dimensional array as a data structure in the foundation area and temporary stock variable. In the foundation area, we need to display four cards that accordingly starts with H, D, S and C letter. For this purpose, I implemented foundationArea array containing 4 different cards. In the foundation area, if all cards will be H13, D13, S13 and C13 game will be end. For that reason, we don't need to know cards from the previous steps, so I decided to go with one dimensional array. If corresponding cards occur in foundation area game will be end. While we executing game commands, if we have an error we don't move card to foundation Area. Additionally, I used array for a stock variable that holds 3 cards from the previous steps. If all cards moved from the waste area, I displayed previous cards from the temporary stock variable.

I used two-dimensional array for the stock and the. Firstly, I would like to explain my implementation in the tableau area. In tableau area, we need to display cards to user. At the beginning of the game we have max 6 closed card and 1 opened card in the last column. According the game rules, if we have card in the same column with correct order 1 to 13 these cards have to move to foundation area. For this reason, this column may take 13 more cards. Totally we may have 19 cards at most for one column. Also, we have 7 columns in tableau area, so I used an array 20 by 7. Besides that, I also stored closed cards in the same logic. When I am opening closed card in the tableau area, I accessed to copy of the first two-dimensional implementation and opened matching card. Secondly, I would like to discuss about stock implementation. We need to store 24 cards in the stock area. I create a two-dimensional array that 9 by 3. I reserve first row for corresponding matching strings which is "___ ____ ___". This part shown at the beginning of the game. Other dimensions of the stock contain triple card with the matching order. Also, I have tail and head counter. If commands equal to "open from stock" we are opening 3 cards from the stock and display in the waste area. At some point of the game, these cards finished and need to reorder according the game flow. For that reason, I used these two counters. When head and tail counter equal to each other, I understand all cards from the stocks area opened and we need to sort of remainders. This sorting operation is for shifting elements. I tried to demonstrate in the given example below.



After sorting completed, we have fresh stock area with reordered cards. This two dimensional stock area following and updating according the given commands. At some point of the game we need to display previous stock element. This is happened when we moved all 3 cards from the waste area which is stock array in our implementation. For the checking this kind of situation I checked current head all cards, if all of them is equal to "___" I use current head tail – 1 to access previous opened cards.