Project 2

Mastermind

Josh Drentlaw

2/8/2018

CSC-5-40652 OL

Introduction

The game I chose for my project is Mastermind. I remember playing it at a very young age with my dad, and having a lot of fun. I felt like it would provide a good challenge, while still being doable in the amount of time, and with the constraints we had. I have implemented a pretty complete version of the game with 2 human players, but the computer player is still not fully functional. It can make a code pretty well, but it can't give correct hints or guess different combinations.

Rules of the Game

The game is pretty simple: There are 2 players, a code maker, and a code breaker. The code maker uses colored pegs to create a sequence of 4 colors (different or the same) that is hidden from the code breaker. The code breaker has 12 attempts to guess the correct sequence, but after each guess, the code maker gives hints to the breaker about which colors they got right, and if they're in the correct position. The code maker has 6 colors to choose from: red, orange, yellow, green, blue, and purple. This allows for 1296 different patterns (6^4=1296). Points are awarded to the maker for each incorrect guess the breaker makes. If the breaker doesn't succeed after the 12th guess, the maker gets an extra point for stumping the breaker. The players chose an even number of matches, and switch off between maker and breaker roles. After the agreed number of matches is completed, the player with the most points is the winner.

Inputs and Outputs

The first thing that is output is a summary of the rules for the game. The first couple inputs ask for the number of human players, and for the number of matches. Both of these inputs are simple int's, and both are validated. Human players should be input as 1 or 2, and the number of matches should be an even, positive number. A match begins with the code maker opening the code-doc.txt file and entering a 4 character code like so: royg. They then enter 1 to move the program to the guessing phase, and behind the scenes, the code is read in from the text file. This was the best way I could currently think of to hide the code from the code breaker. Originally I had the console output several end lines, to shoot the code out of sight as quickly as possible, but it felt a little hack. The score is also displayed during the code entering phase, which will display 0-0 for the initial run. During the guessing, the output follows this pattern:

Guess i: ← A for loop, loops through the 12 guesses, and i from that loop is used here

royb ← This is the guess made by the code breaker

xxx ← Hints left by the code maker. x or o are used.

I initially had the hint system wrong, and trying to correct it took up all my time. I could not get the program to run as expected in time, and if there is more than one color in the code, the hints will not display properly. Originally I had the hints correspond directly to the position of the color in the guess, but this is not actually how it works.

The x's and o's don't correspond to the position of the colors in the guess, they just appear x's first followed by o's.

Finally, the game outputs that the code has been guessed if the code breaker was successful, or if the number of attempts has been reached. In either case, it outputs the number of points the code maker earned during the current match. After all the matches have been played, it outputs the final score and the winner.

Pseudo Code

```
// HEADER

main function
        Output rules
        Input # of humans, 1 or 2
        Input # matches
        If it's the humans turn run validateCode()
        If it's the computers turn, run computerCode()
        While matches > 0
                If it's the humans turn they make a guess
                If it's the computers turn, run computerCode() to make a random guess
                Did The Breaker guess correctly?
                        End game and start new match
                Maker gives hints
                Matches--
        Output winner



readCode function
        Open code-doc.txt
        Read code and assign it to code
        Return code



computerCode function
        Seed random with time
```

4 times:
Generate random number
Use as index in colors[]
Add to code
Return code

validateCode function
Save code in code-doc.txt
readCode() returns code
Validate code
Return code

getHints function
Push code to check
Loop thru each letter in guess
Loop thru each letter in the check
Is this color in the check?
Is it in the same position?
Add x to right
Mark found
If not
Add o to almost
Mark found
Return right+almost

computerGuess function
I wanted to make something interesting where the computer logs the guesses it made and tries new guesses based on old guesses and how good the hints were from that guess, but I wasn't even able to begin. I figured I'd be able to fulfill my array and/or vector requirements here.

getWinner function
If player 1 has more points
Return player 1
If player 2 has more points
Return player 2
Else
Return tie