Monopoly

Steve Fox

CSC-5

Project 1

40715

Introduction

Monopoly is game that many of us played as we were growing up. Who could generate the largest banks and acquire the most property while the other players are reduced to nothing. Coding this game provided many inquiries into the c++ language do to the logic necessary for the program to function properly. This version is a two player game of monopoly that utilizes a random number generator to simulate actual dice roles. These dice rolls accumulate and dictate what function is called that corresponds to the current dice roll. If the total dice roll goes over 39 then it is set to zero and $200 is passed into the players bank to simulate passing go. In this version all properties are established except community chess, jail, and go to jail which will be available in the next version. Chance is also dictated by a random number generator which utilizes a switch statement to execute what card is being drawn which parallels what community chest will be.

Furthermore, to keep track of the properties that each player purchases a two dimensional array was established for each player. Both players array are initiated with ? and when a property is purchased than a b is passed to the array indicating the property was purchased. The same is true for house. However, in this version you must own all three properties of a corresponding color in order to purchase houses but you will not have a cost subtracted from your bank. In the next version, appropriate fees will be established for the purchasing of house. Until then, getting three properties of corresponding colors is the goal and this made even harder because in this version you cannot sell property. In the next version you will be able to do so. The goal of Monopoly in this version in not to let your bank go negative or your defeated. And because you cannot sell property, you must purchase with caution because any small fee could put your balance in negative.

The game is ran by a do while loop that stops when a bank goes negative. Every valid dice roll has a property that corresponds to the number. The property names and their rates are dictated by files that are imported into the program called “Propert.txt.” and “rent.txt.” By sending these files to elements in an array it made it a lot easier to duplicate functions that had similar algorithims. Furthermore, having four parallel arrays made working with the data so much easier. So enjoy this timeless game and good luck.

Required Structures

(Note there are numerous times when these structures are utilized so this will be an example of each)

|  |  |  |
| --- | --- | --- |
| Structure | Line | Purpose |
| For loop | 96-115 | Initialize Arrays with ? |
| While loop | 88 | Write file to Array |
| Do while loop | 120-343 | Controls players turns and whether someone has lost |
| Switch Statement | 2710-2901 | Used in Chance function  To draw a card |
| If/else if/else | 261-343 | Dictates fuction call of player 1’s dice roll |
| Nested if statements | 3626-3982 | Menu and check to see if player owns required properties to purchase houses |
| Array | 70 | Holds properties names imported from files |
| 2D Array | 71 | Holds the properties that player1 owns |
| Files | 87&95 | Holds property names and  rents |

Functions

|  |  |  |
| --- | --- | --- |
| Functions | Line | Purpose |
| int rollDie(string [], char [][7],int&,int&, int,int \*); | 347 | Rolling dice |
| void purp1(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 380 | Mediterranean Ave Buy and rent costs |
| void purp2(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 483 | Baltic Ave Buy and rent costs |
| void teal1(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 586 | Oriental Ave Buy and rent costs |
| void teal2(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 690 | Vermont Ave Buy and rent costs |
| void teal3(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 797 | Connecticut Ave  Buy and rent cost |
| void pink1(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 900 | St Charles Place  Buy and rent cost |
| void pink2(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 1005 | States Ave  Buy and rent cost |
| void pink3(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 1110 | Virgina Ave  Buy and rent cost |
| void orange1(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 1214 | St James Place  Buy and rent cost |
| void orange2(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 1321 | Tennessee Ave  Buy and rent cost |
| void orange3(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 1426 | New York Ave  Buy and rent cost |
| void red1(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 1530 | Kentucky Ave  Buy and rent cost |
| void red2(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 1636 | Indiana Ave  Buy and rent cost |
| void red3(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 1742 | Illinois Ave  Buy and rent cost |
| void yellow1(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 1849 | Atlantic Ave  Buy and rent cost |
| void yellow2(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 1954 | Ventnor Ave  Buy and rent cost |
| void yellow3(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 2058 | Marvin Gardens  Ave  Buy and rent cost |
| void green1(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 2165 | Pacific Ave  Buy and rent cost |
| void green2(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 2271 | North Carolina Ave  Buy and rent cost |
| void green3(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 2376 | Pennsylvania Ave  Buy and rent cost |
| void blue1(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 2480 | Park Place  Buy and rent cost |
| void blue2(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 2585 | Board walk  Buy and rent cost |
| void start(); | 2612 | Starting point |
| int chance(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool,int &,int &,int); | 2689 | Chance cards |
| int comChest(string[], char[][7],char[][7], int&,int&,int); | 2901 | Community Chest Cards(Not working) |
| int incTax(string[],int&,int&,bool); | 3439 | Income tax charges |
| int luxTax(string[], int&,int&,bool); | 3424 | Luxury tax charges |
| int jail(string[], char[][7],char[][7], int&,int&,int); | 3416 | Jail  (Not Working) |
| int goToJail(string[], char[][7],char[][7], int&,int&,int); | 3420 | Go To Jail,  Not working |
| void rr1(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 2905 | Reading Railroad  Buy and rent cost |
| void rr2(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 3030 | Pennsylvania railroad  Buy and rent cost |
| void rr3(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 3157 | B&O Railroad  Buy and rent cost |
| void rr4(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 3284 | Shortline Railroad  Buy and rent cost |
| void elecCom(string[], char[][COL],char[][COL], int&,int&,bool,int); | 3584 | Electric Company  Buy and rent cost |
| void watWork(string[], char[][COL], char[][COL],int&,int&,bool,int); | 3485 | Water works Ave  Buy and rent cost |
| int frePark(string[]); | 3410 | Free Parking |
| int menu(string[], int[][COL],char[][COL],char[][COL], int&,int&, bool); | 3617 | For placing houses on properties |

PseudoCode

Bool turn=false;

Player 1 RollDie();

Player 1 dice roll total=sum of dice;

Function call() based on based on player 1 Dice Total;

Bool buy = false

Buy=property;

Choice= yes or no;

If yes bool switch to true;

Character b placed in array;

Collect rent if opponent lands on it

Option for houses

If dice total is greater than 39. Minus 39 from total and gain $200

Bool turn=true

Player 2 RollDie();

Player 2 RollDie();

Player 2 dice roll total=sum of dice;

Function call() based on based on player 2 Dice Total;

Bool buy = false

Buy=property;

Choice= yes or no;

If yes bool switch to true;

Place character b in array;

Collect rent if opponent lands on it

Option for houses

If dice total is greater than 39. Minus 39 from total and gain $200

If(bank1<0)

Player 1 losses

Else (bank2<0)

Player 2 losses

Flowchart

Note a copy of this flowchart is in the project 1 folder

