Mastermind

CSC-7 42486

Branden Hitt

Date: 04/27/2018

Table of Contents

Introduction

Title: Mastermind

This is a small implementation of the game Mastermind, using 4 digits (1-6) that user must guess within 10 turns. The user may also choose whether or not to allow duplicates in the answer key they are attempting to guess.

After each guess the program displays the current guess along with the number of close or exact guesses. They are also given the previous guesses. The program continues in this fashion until either the player runs out of guesses or they guess the numbers correctly.

Close guess: The user guessed a correct digit, just in the wrong place.

Exact guess: The user guessed a correct digit in the correct place.

Ex:

Enter in a guess now XXXX (digits 1-6):

3456

6 5 4 3

3 4 5 6

Close numbers: 4

Exact numbers: 0

Summary and Future Improvements

Project size: 170 lines

The program was fairly simple to write, although I ran into some trouble accounting for duplicates. The idea going forward would be to come up with some sort of AI that could play the game and guess the correct key within a small number of guesses. It may also be interesting to display the game in a better UI that is more appealing.

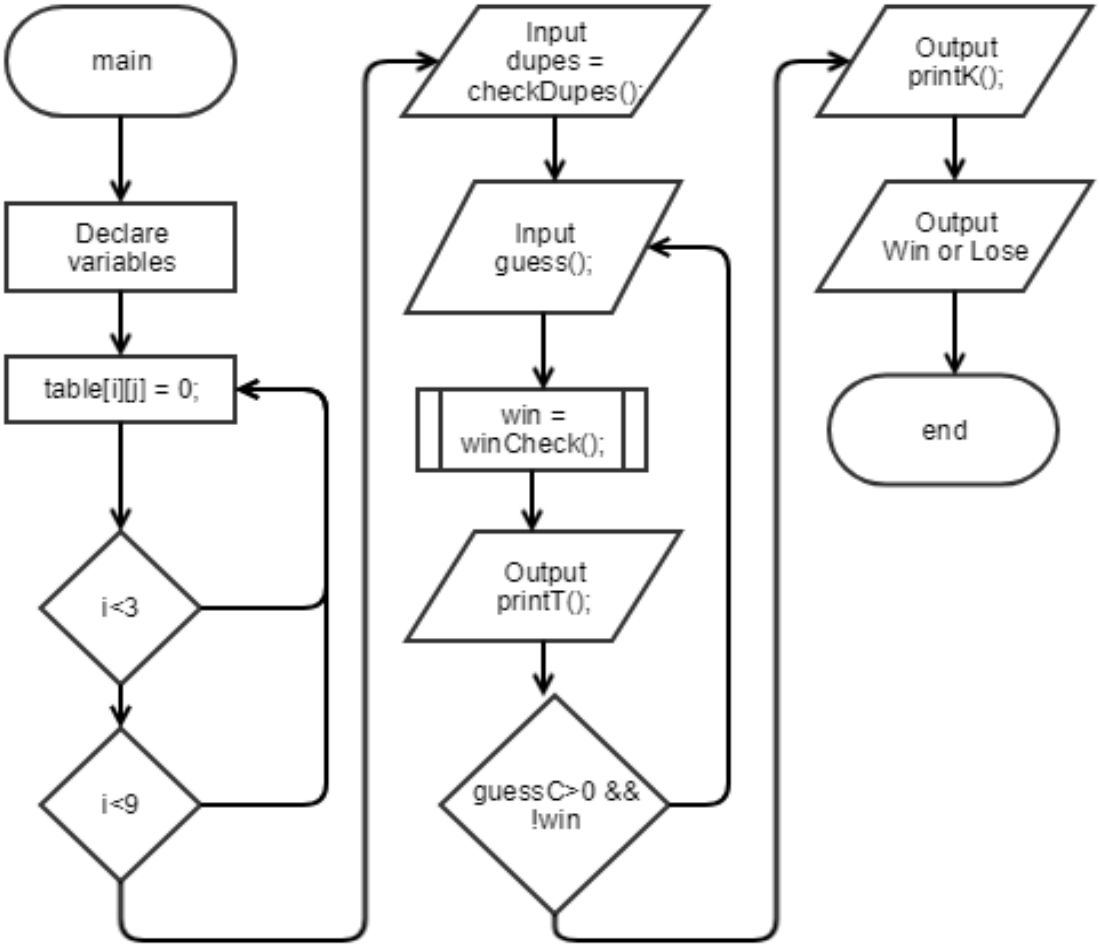
Simple Program Function Overview

```
main{
        srand(); <- set random seed
        dupes(); <- check if the player wants duplicates
        createG(); <- create answer key
        do{
                guess(); <- prompt for player guess
                winCheck(); <- check to see if player won the game
                printT(); <- display past guesses/ current guess/ close and exact count
        }while(); <- loop until guesses run out or game is won
        printK(); <- display the answer key
}
```

Pseudocode

```
/*
 * MASTERMIND
 */

/*
 * File:   main.cpp
 * Author: Branden Hitt
 *
 * Created on April 4, 2018, 7:43 PM
 */

/*
 *
 */
//Function Prototypes
            //print table
            //prompt for duplicates
            //create Game
            //print key
            //guess row
            //validate guess
            //check for game win

int main(int argc, char** argv) {
    //declare variables

    //set random seed

    //create 2d array, answer array, current guess array

    //total guesses allowed

    //initialize display table to zeros

    //check for duplicates or not

    //create answer key

    //run through guesses

       //prompt for guess

       //check for win

       //print response

    //printT(table);

    //display win or loss
}
void printT(int t[][4],int rows, int close, int exact){
    //print table
}
```

```
bool checkDupes(){
   //prompt for duplicates with validation
}
void createG(int a[], int range, bool dupes){
       //create answer key without duplicates within range

       //or create answer key with dupes

}
void printK(int k[]){
   //print out key
}
void guess(int g[], int &gC, int t[][4], int range, bool dupes){

   //output number of guesses remaining


   //prompt for guess


   //read in guess and validate

}

bool winCheck(int cG[], int key[], int &close, int &exact, bool dupes){

   //temp count for dupes


   //set close and exact back to zero


   //loop through to find close numbers

   //count total number for close guesses

   //loop through to find exact


}

bool validate(string g, bool dupes){

   //regex


   //if not alowing duplicated

     //make sure input doesnt have duplicates

   //check for four numbers


   //return true if passing validation

}
```


main
Declare variables
table[i][j] = 0;
i<3
i<9
Input dupes = checkDupes();
Input guess();
win = winCheck();
Output printT();
guessC>0 && !win
Output printK();
Output Win or Lose
end

Program Code

```
/*
 * MASTERMIND
 */

/*
 * File:   main.cpp
 * Author: Branden Hitt
 *
 * Created on April 4, 2018, 7:43 PM
 */



#include <iostream> //io
#include <cstdlib>  //rand
#include <regex>    //regular expression

using namespace std;

/*
 *
 */
//Function Prototypes
void printT(int [][4], int, int ,int);          //print table
bool checkDupes();                              //prompt for duplicates
void createG(int [], int, bool);                //create Game
void printK(int []);                           //print key
void guess(int [], int &, int [][4], int, bool);//guess row
bool validate(string, bool);                    //validate guess
bool winCheck(int [], int [],int &, int &, bool);//check for game win

int main(int argc, char** argv) {
    //declare variables
```

```
const int XDIM=4;               //table dimensions (digits to guess)
const int YDIM=10;              //table dimensions (amount of possible guesses)
bool win = false;               //win sentinel value
bool dupes = false;             //duplicates allowed or not
int close=0, exact=0;           //temp value holders
//set random seed
 srand(static_cast<unsigned int>(time(0)));
//create 2d array, answer array, current guess array
int table[YDIM][XDIM] = {};
int answers[4] = {};
int cGuess[4] = {0,0,0,0};
int range = 6;  //from 1 to __
//total guesses allowed
int guessC = 10;
//initialize display table to zeros
for(int i=9;i>=0;i--){
    for(int j=3;j>=0;j--){
        table[i][j]=0;
    }
}
//check for duplicates or not
dupes = checkDupes();
//create answer key
createG(answers,range,dupes);
//run through guesses
do{
   //prompt for guess
   guess(cGuess, guessC, table, range, dupes);
   //check for win
   win = winCheck(cGuess, answers, close, exact, dupes);
   //print response
   printT(table,guessC,close,exact);
}while(guessC > 0 && !win);
```

```
    //print
    //printT(table);
    printK(answers);
    if(win) cout<<"Congrats! You have won!";
    else cout<<"You have run out of guesses. You lose.";
    return 0;
}
void printT(int t[][4],int rows, int close, int exact){
    for(int i=0;i<10-rows;i++){
        for(int j=0;j<4;j++){
            cout<<t[i][j]<<" ";
        }
        cout<<endl;
    }
    cout<<"Close numbers: "<<close<<endl;
    cout<<"Exact numbers: "<<exact<<endl;
    cout<<"********************"<<endl;
}
bool checkDupes(){
    string temp;
    int temp2;
    do{
        cout<<"Would you like to play with duplicates?"<<endl;
        cout<<"Enter 1 for yes, or 2 for no"<<endl;
        cin>>temp;
        temp2 = temp[0] - '0';
    }while(temp2!=1 && temp2!=2);
    if(temp2 == 1) return true;
    else return false;
}
void createG(int a[], int range, bool dupes){
    if(!dupes){
        //create answer key without duplicates within range
```

```
        a[0] = rand()%range +1;
        do{
            a[1] = rand()%range +1;
        }while(a[1] == a[0]);
        do{
            a[2] = rand()%range +1;
        }while(a[2] == a[1] || a[2] == a[0]);
        do{
            a[3] = rand()%range +1;
        }while(a[3] == a[2] || a[3] == a[1] || a[3] == a[0]);
    }else{
        //create answer key with dupes
        a[0] = rand()%range +1;
        a[1] = rand()%range +1;
        a[2] = rand()%range +1;
        a[3] = rand()%range +1;
    }
}
void printK(int k[]){
    cout<<"Key : ";
    for(int i=0;i<4;i++){
        cout<<k[i]<<" ";
    }
    cout<<endl;
}
void guess(int g[], int &gC, int t[][4], int range, bool dupes){
    string guess;
    cout<<"You have "<<gC<<" guesses remaining."<<endl;
    do{
        cout<<"Enter in a guess now XXXX (digits 1-"<<range<<"): "<<endl;
        cin>>guess;
    }while(validate(guess,dupes));
    for(int i=0;i<4;i++){
```

```
        g[i] = guess[i] - '0';
        t[10-gC][i] = guess[i] - '0';
    }
    gC--;
}
bool winCheck(int cG[], int key[], int &close, int &exact, bool dupes){
    //temp count for dupes
    int closeA[] = {0,0,0,0};
    //set close and exact back to zero
    close=0, exact=0;
    //loop through to find close numbers
    for(int i=0;i<4;i++){
        for(int j=0;j<4;j++){
            if(key[i] == cG[j]){
                closeA[i]++;
            }
        }
    }
    //count total number for close guesses
    for(int i=0;i<4;i++){
        if(closeA[i]>0) close++;
    }
    //loop through to find exact
    for(int i=0;i<4;i++){
        if(cG[i] == key[i]) exact++;
    }
    if(exact == 4)return true;
    return false;
}
bool validate(string g, bool dupes){
    regex myRegex("[1-9][1-9][1-9][1-9]");
    if(!dupes){
        if(g[0] == g[1] || g[0] == g[2] || g[0] == g[3]) return true; //check for duplicate numbers
```

```
        if(g[1] == g[2] || g[1] == g[3]) return true;
        if(g[2] == g[3]) return true;
    }
    if(regex_match(g,myRegex))return false; //check for four numbers
    return true;
}
```