Project 2

Project Title: Additionally Fun

Name: Kevin Vo
Course: CSC-5
Date: June 2, 2012

Introduction

Title: Additionally Fun

Additionally Fun is a competitively enjoyable game that operates by providing a series of simple mathematical problems, for younger children who are starting out with addition. It provides them a competitive environment, by allow the child to challenge one of their friends to a math dual. Both will be given identical math problems to deliver a fair chance, so that the occurrence of one player having more difficult problems than the other will be eliminated. Then once the game is over it will display the answer of both players on a chart separated by the correct answer in the middle. On top of that custom usernames will be assigned to each player so they know whose answers are which; along with how many problems they got incorrect.

Summary

Project Size: 344 lines

Number of Functions: 6

Many aspects from this project derived from chapter 6 and 7, with a small fraction from chapter 8 from the Gaddis book as required. Many of the ideas came when I was reviewing my previous work, Project 1: Business & Pleasure. Including the realization that with the help of arrays (both 1 and 2 dimensions), the opportunity of storing more information has become much simpler. Therefore the idea of creating a multiplayer game with the same caliber has been implemented. Instead of playing by oneself, a new system has been introduced when another person is able to step into the arena.

Description

Alternately the decision is made of centering this game for child rather than their adult counterparts as the previous project did. Project 2 will offer identical math problems to improve on their memorization skills as well. It also has a multifunctional uses such as a math quiz students within a class room of lower grade.

Pseudo Code

```
Initialize
Displays the rules and game description.
Declares and stores all of the predefined answers to an 1D array.
Iterates another 1D array twice to store both usernames as string.
Loops the addition problems twice for player 1 and 2.
If the iteration is on 0 (first loop) then player 1's username will display.
    If it is the first round, the first math problem will display along with the current and possible score.
    Stores the inputted data.
    Calculates the stored inputted data to provide a score.
        If player 1 inputted data match then score increases by 50.
Else if inner loop iterates a second time then the second math problem will display.
Repeats last step four more times for player 1.
If outer first iteration is done player 2's username will display.
    The same previous math problems will be redisplayed for player 2 along with the current and possible
    score.
    Stores the inputted data.
    Calculates the stored inputted data to provide a score.
        If player 2 inputted data match then score increases by 50.
    Repeats last step four more times for player 2.
Displays the title and borders of the score screen.
Displays the answers of both players separated by the correct answers column.
Displays the result of how many answers were wronged by player 1.
Displays the result of how many answers were wronged by player 2.
Linear Search finds and sorts out the answer containing array to find which problem has the answer 15 from the
beginning until the value 15 is found.
```

If there is no such value found in the array a passage confirming that displays.

Else displays the number of the problem with 15 as the answer in the array.

Starts again.

Program

```
/*
 * File:   main.cpp
 * Author: Kevin Vo
 * Assignment: Project #2 for CSC-5 (43969)
 * Description: 2 player math game that keeps scores and declares the victor.
 * Created on May 23, 2012, 4:16 PM
 */

#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
// EndGameStatDisplayer is an example of overloading functions
void EndGameStatDisplayer(int, int, string[]);
void EndGameStatDisplayer(int[][2], int[]);

int answerCorrectorPlayer1(int[], int[][2]);
int answerCorrectorPlayer2(int[], int[][2]);
int pointMeterPlayer1(int);
int healthMeterPlayer2(int);

//nextPossibleScore...(); demonstrates passing data by value by only changing
//the value within the scope of the function
void nextPossibleScorePlayer1(int);
void nextPossibleScorePlayer2(int);

//rulesDisplay function shows usage of default arguments
void rulesDisplay(const int one = 1, const int two = 2, const int three = 3);
```

```
//shows Searching and Storing arrays with linear search

int findingFifteen(const int[], int, int);



//Connstant global variable for array searcher & sorter

const int sizeCorrectAnswers = 5;

int main(int argc, char** argv) {

   const int SIZE = 11;

   int userInput[5][2];

   int randTopNum[SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10},

      randBotNum[SIZE] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10},

         answerAdd[5];

   string arrName[2];

   int player1_num_wrong, player2_num_wrong, pointsPlayer1, pointsPlayer2;

   int correctAnswers[sizeCorrectAnswers] = {9, 5, 9, 15, 6};

   int similarAnswers;



   char restart;//Choice to restart

   do{

   //Rules Display

    rulesDisplay();



     //answers for the math problems

     answerAdd[0] = randTopNum[0] + randBotNum[9];

     answerAdd[1] = randTopNum[2] + randBotNum[3];

     answerAdd[2] = randTopNum[4] + randBotNum[5];

     answerAdd[3] = randTopNum[8] + randBotNum[7];

     answerAdd[4] = randTopNum[5] + randBotNum[1];



     for (int user = 0; user <= 1; user++){

            cout<<"Please enter username for player # "<<(user + 1)<<": ";

            cin>>arrName[user];

    }



     for (int player = 0; player <= 1; player++){
```

```
if (player == 0){
    cout<<"Player: "<<arrName[player]<<endl;
    for (int x = 0; x <= 4; x++){
    //Displays the math problem

        if (x == 0){
            player1_num_wrong = answerCorrectorPlayer1(answerAdd,
                    userInput);
            pointsPlayer1 = pointMeterPlayer1(player1_num_wrong);

            cout<<randTopNum[0]<<" + "<<randBotNum[9]<<" = ";
            cin>>userInput[0][0];

            //Damage and Health System
            player1_num_wrong = answerCorrectorPlayer1(answerAdd,
                    userInput);
            pointsPlayer1 = pointMeterPlayer1(player1_num_wrong);



            //Display for points
            cout<<"                                          "<<
                  "Player 1's Points: "<<(pointsPlayer1 + 50)<<endl;

            //Displays the next possible score if answer is correct
            nextPossibleScorePlayer1(pointsPlayer1);

        }
        else if (x == 1){
            cout<<randTopNum[2]<<" + "<<randBotNum[3]<<" = ";
            cin>>userInput[1][0];

            //Damage and Health System
            player1_num_wrong = answerCorrectorPlayer1(answerAdd,
                    userInput);
            pointsPlayer1 = pointMeterPlayer1(player1_num_wrong);
    //Display for points
    cout<<"                                   "<<
        "Player 1's Points: "<<(pointsPlayer1 + 50)<<endl;

    //Displays the next possible score if answer is correct
    nextPossibleScorePlayer1(pointsPlayer1);
}
else if (x == 2){
    cout<<randTopNum[4]<<" + "<<randBotNum[5]<<" = ";
    cin>>userInput[2][0];

    //Damage and Health System
    player1_num_wrong = answerCorrectorPlayer1(answerAdd,
          userInput);
    pointsPlayer1 = pointMeterPlayer1(player1_num_wrong);

    //Display for Health
    cout<<"                                   "<<
        "Player 1's Points: "<<(pointsPlayer1 + 50)<<endl;

    //Displays the next possible score if answer is correct
    nextPossibleScorePlayer1(pointsPlayer1);
}
else if (x == 3){
    cout<<randTopNum[8]<<" + "<<randBotNum[7]<<" = ";
    cin>>userInput[3][0];

    //Damage and Health System
    player1_num_wrong = answerCorrectorPlayer1(answerAdd,
          userInput);
    pointsPlayer1 = pointMeterPlayer1(player1_num_wrong);

    //Display for points
    cout<<"                                   "<<
        "Player 1's Points: "<<(pointsPlayer1 + 50)<<endl;
```

```
            //Displays the next possible score if answer is correct
            nextPossibleScorePlayer1(pointsPlayer1);
        }
        else if (x == 4){
            cout<<randTopNum[5]<<" + "<<randBotNum[1]<<" = ";
            cin>>userInput[4][0];

            //Damage and Health System
            player1_num_wrong = answerCorrectorPlayer1(answerAdd,
                    userInput);
            pointsPlayer1 = pointMeterPlayer1(player1_num_wrong);

            //Display for points
            cout<<"                                    "<<
                 "Player 1's Points: "<<(pointsPlayer1 + 50)<<endl;

            //Displays the next possible score if answer is correct
            nextPossibleScorePlayer1(pointsPlayer1);
        }
    }
                        }
if (player == 1){
    cout<<"Player: "<<arrName[player]<<endl;
    //Displays the math problem
    cout<<randTopNum[0]<<" + "<<randBotNum[9]<<" = ";
    cin>>userInput[0][1];

    player2_num_wrong = answerCorrectorPlayer2(answerAdd, userInput);
    pointsPlayer2 = healthMeterPlayer2(player2_num_wrong);

            //Display for Health
            cout<<"                                    "<<
                 "Player 2's Points: "<<(pointsPlayer2 + 50)<<endl;
            nextPossibleScorePlayer2(pointsPlayer2);
    cout<<randTopNum[2]<<" + "<<randBotNum[3]<<" = ";
    cin>>userInput[1][1];
```

```
player2_num_wrong = answerCorrectorPlayer2(answerAdd, userInput);
pointsPlayer2 = healthMeterPlayer2(player2_num_wrong);

        //Display for Health
        cout<<"                                    "<<
             "Player 2's Points: "<<(pointsPlayer2 + 50)<<endl;
        nextPossibleScorePlayer2(pointsPlayer2);
cout<<randTopNum[4]<<" + "<<randBotNum[5]<<" = ";
cin>>userInput[2][1];

player2_num_wrong = answerCorrectorPlayer2(answerAdd, userInput);
pointsPlayer2 = healthMeterPlayer2(player2_num_wrong);

        //Display for Health
        cout<<"                                    "<<
             "Player 2's Points: "<<(pointsPlayer2 + 50)<<endl;
        nextPossibleScorePlayer2(pointsPlayer2);
cout<<randTopNum[8]<<" + "<<randBotNum[7]<<" = ";
cin>>userInput[3][1];

player2_num_wrong = answerCorrectorPlayer2(answerAdd, userInput);
pointsPlayer2 = healthMeterPlayer2(player2_num_wrong);

        //Display for Health
        cout<<"                                    "<<
             "Player 2's Points: "<<(pointsPlayer2 + 50)<<endl;
        nextPossibleScorePlayer2(pointsPlayer2);
cout<<randTopNum[5]<<" + "<<randBotNum[1]<<" = ";
cin>>userInput[4][1];

player2_num_wrong = answerCorrectorPlayer2(answerAdd, userInput);
pointsPlayer2 = healthMeterPlayer2(player2_num_wrong);

        //Display for Health
        cout<<"                                    "<<
```

```
                "Player 2's Points: "<<(pointsPlayer2 + 50)<<endl;
             nextPossibleScorePlayer2(pointsPlayer2);
                   }
         }


//Both EndGameStatDisplayers is an example of
//Displays the the players' stats when the game ends
EndGameStatDisplayer(pointsPlayer1, pointsPlayer2, arrName);
    cout<<"                   Player's Stats\n";
    cout<<"Player 1: "<<arrName[0]<<", Player 2: "<<arrName[1]<<endl;
    cout<<"=======================================================\n";
    cout<<"Player 1 Answers |  Correct Answers  |  Player 2 Answers"
          <<endl;


EndGameStatDisplayer(userInput, answerAdd);
    cout<<arrName[0]<<" has wrong answers: "
          <<answerCorrectorPlayer1(answerAdd, userInput)<<"/5\n";
    cout<<arrName[1]<<" has wrong answers: "
          <<answerCorrectorPlayer2(answerAdd, userInput)<<"/5\n";


    //This is also an example of an argument list
    //linear searches which problem has 15 for its correct answer.
    similarAnswers = findingFifteen(correctAnswers, sizeCorrectAnswers,
          15);




    if (similarAnswers == -1){
       cout<<"Sorry there are no problems that has 15 as a correct"<<
             " answer\n";
    }
    else{
       cout<<"15 is the correct answer for problem: "
             <<(similarAnswers + 1)<<endl;


    }
```

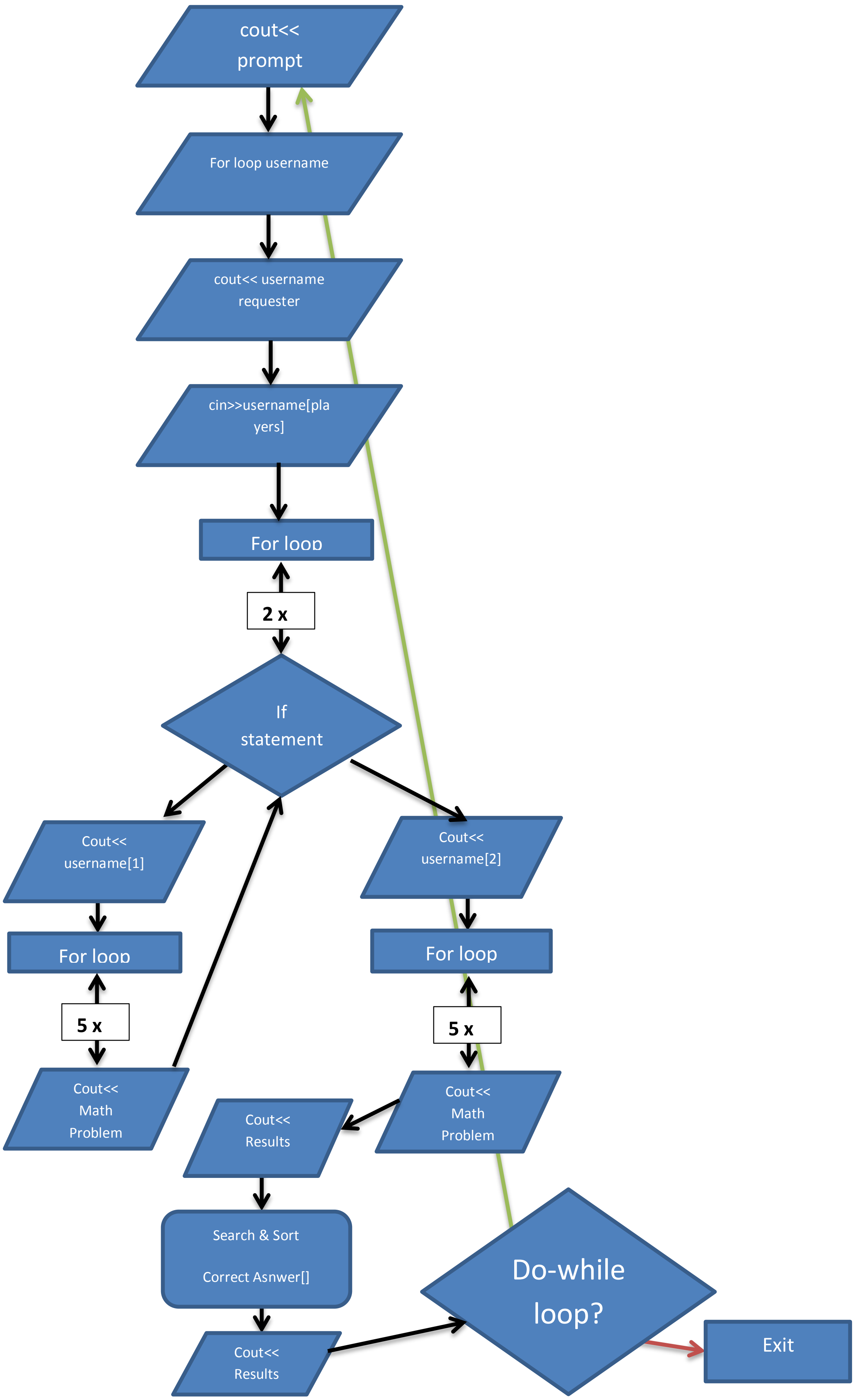
```
            cout<<"Would you like this program to restart itself? (y/n): ";

            cin>>restart;

            }while(restart == 'y' || restart == 'Y');

    return 0;

}




int answerCorrectorPlayer1(int answerAdd[], int userInput[][2]){

    int amount_of_wrong_answer = 0;


    for (int checker = 0; checker <= 4; checker++){

        if (userInput[checker][0] != answerAdd[checker]){

            amount_of_wrong_answer = amount_of_wrong_answer + 1;

        }

    }



return amount_of_wrong_answer;}


int answerCorrectorPlayer2(int answerAdd[], int userInput[][2]){

    int amount_of_correct_answer = 0;


    for (int checker = 0; checker <= 4; checker++){

        if (userInput[checker][1] != answerAdd[checker]){

            amount_of_correct_answer = amount_of_correct_answer + 1;

        }

    }



return amount_of_correct_answer;}


//pointMeterPlayer1(); and healthMeterPlayer2(); demonstrates returning values

//by returning totalHealth and current_Health.

int pointMeterPlayer1(int player1_num_wrong){

    int current_Health = 200, totalHealth;
```

```
   totalHealth = (current_Health - (player1_num_wrong * 50)) ;

return totalHealth;}

int healthMeterPlayer2(int player2_num_wrong){
   int current_Health = 200;

   current_Health = current_Health - (player2_num_wrong * 50) ;

return current_Health;}


//Parallel Arrays are used here since both array userInput and answerAdd uses
//the same subscript "i" to relate the players' answers to the correct ones
void EndGameStatDisplayer(int userInput[][2], int answerAdd[]){

   //Grid organizer for display of answers
        for (int i = 0; i <= 4; i++){
           if (answerAdd[i] > 10 && userInput[i][0] <= 10){
                cout<<"      "<<userInput[i][0]<<"       |        "
                   <<answerAdd[i]<<"      |      "
                   <<userInput[i][1]<<endl;
           }
           else if (answerAdd[i] > 10 && userInput[i][0] > 10){
                cout<<"      "<<userInput[i][0]<<"      |        "
                   <<answerAdd[i]<<"      |      "<<userInput[i][1]
                   <<endl;
           }
           else if (answerAdd[i] <= 10 && userInput[i][0] <=10){
                cout<<"      "<<userInput[i][0]<<"       |        "
                   <<answerAdd[i]<<"       |      "<<userInput[i][1]
                   <<endl;
           }
           else if (answerAdd[i] <= 10 && userInput[i][0] > 10){
                cout<<"      "<<userInput[i][0]<<"      |      "
```

```
                        <<answerAdd[i]<<"        |       "
                        <<userInput[i][1]<<endl;
                }
            }
}


//This contains an multiple parameters
//Declares the winner by finding the player with the most points
void EndGameStatDisplayer(int pPlayer1, int pPlayer2, string arrName[]){


    if (pPlayer1 > pPlayer2){
        cout<<"Player "<<arrName[0]<<" has won the game.\n";
    }
    else if (pPlayer2 > pPlayer1){
        cout<<"Player "<<arrName[1]<<" has won the game.\n";
    }
    else if (pPlayer1 == pPlayer2){
       cout<<"Player "<<arrName[0]<<" and Player "<<arrName[1]<<" both tied.\n";
    }


}



//This function displays the start up game's description and along with its
//rules also demonstrates the usage of default arguments
void rulesDisplay(const int one, const int two2, const int three3){
    cout<<"\nDescription: This program will provide a series of math equations "
            <<"starting with the first player.\n"
        <<"Then after when the first player's turn is complete the same example"
            <<" problems will redisplay for the\n"
        <<"second player to try and out do the first player.\n";
    cout<<"****************************** Rules *****************************\n";
    cout<<one<<") Win the game by scoring more points then your opponent.\n";
    cout<<two2<<") If an answer is wrong you will earn no points.\n";
    cout<<three3<<")The game will not accept an input value that is not in a "
            <<"form of a digit.\n";
```

```
}


//These functions demonstrates passing data by value by only changing the
//player's score value within its scopes
void nextPossibleScorePlayer1(int player1Points){
   cout<<"                                    Player 1's Next Possible"
         <<" Points: "<<(player1Points + 100)<<endl;
}



void nextPossibleScorePlayer2(int player2Points){
   cout<<"                                    Player 2's Next Possible"
         <<" Points: "<<(player2Points + 100)<<endl;
}


//Uses linear search to find which problem has 15 for its correct answer.
int findingFifteen(const int correctAnswers[], int sizeCorrectAnswers,
      int answerFifteen){


   int index = 0, position = -1;
   bool found = false;


   while (index < sizeCorrectAnswers && !found){
      if (correctAnswers[index] == answerFifteen){


         found = true;
         position = index;


      }
      index++;
   }
   return position;
}
```

Flow Chart



cout<< prompt
For loop username
cout<< username requester
cin>>username[players]
For loop
2 x
If statement
Cout<< username[1]
For loop
5 x
Cout<< Math Problem
Cout<< username[2]
For loop
5 x
Cout<< Math Problem
Cout<< Results
Search & Sort
Correct Asnwer[]
Cout<< Results
Do-while loop?
Exit