Final project CSC-17A 48983

# Author

David Hudkins II

HANGMAN

**INTRODUCTION:**

This project consists of the game hangman. This program is a throwback to elementary school times where hangman made learning words fun and frustrating. I chose to do this project because it is a game I am quite familiar with and was easier for me to visualize how the program should execute based on the functionality of the game. It is important because it utilizes arrays and character comparison through the use of classes, which is what object orientation in C++ is centralized around.

**HOW TO PLAY**:

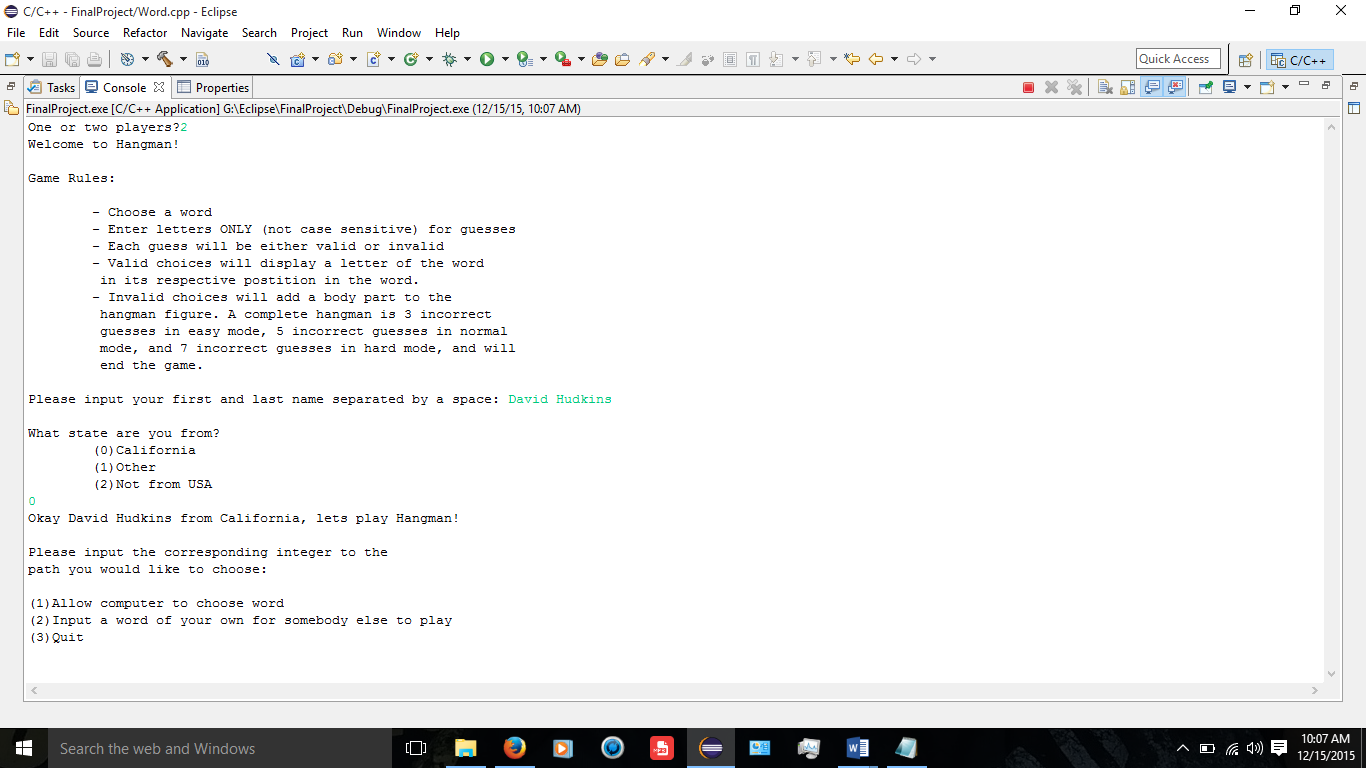
The game hangman is simple, albeit very difficult depending on the length of the word. The game begins by prompting the user to specify one or two players participating. The player then enters their personal information and begins by choosing a player entered word or a computer generated word from one of the three difficulty tiers. After this is specified, the game begins by displaying the number of characters represented by a field of the same number of dashes displayed below the empty board. The user has the opportunity to make 3, 5, and 7 incorrect guesses respective to easy, normal, and hard mode. The user defined category defaults to 5 incorrect guesses allowed. The user then enters their guess one character at a time either the hangman figure is completed, or the user completes the word, thereby winning the game. Each win is incremented and kept track of for later comparison with the second player for number of wins achieved. At the end of the game, the user is prompted to play again or quit. If they gameplay is repeated (personal information code excluded but retained). If they quit (and are the first player) then execution is passed to player number two and the same game is repeated for them. Upon exiting execution from player two, console displays who accumulated more wins, and then the program terminates.

**TOTAL LINES CONTAINED IN SOURCE FILE:**

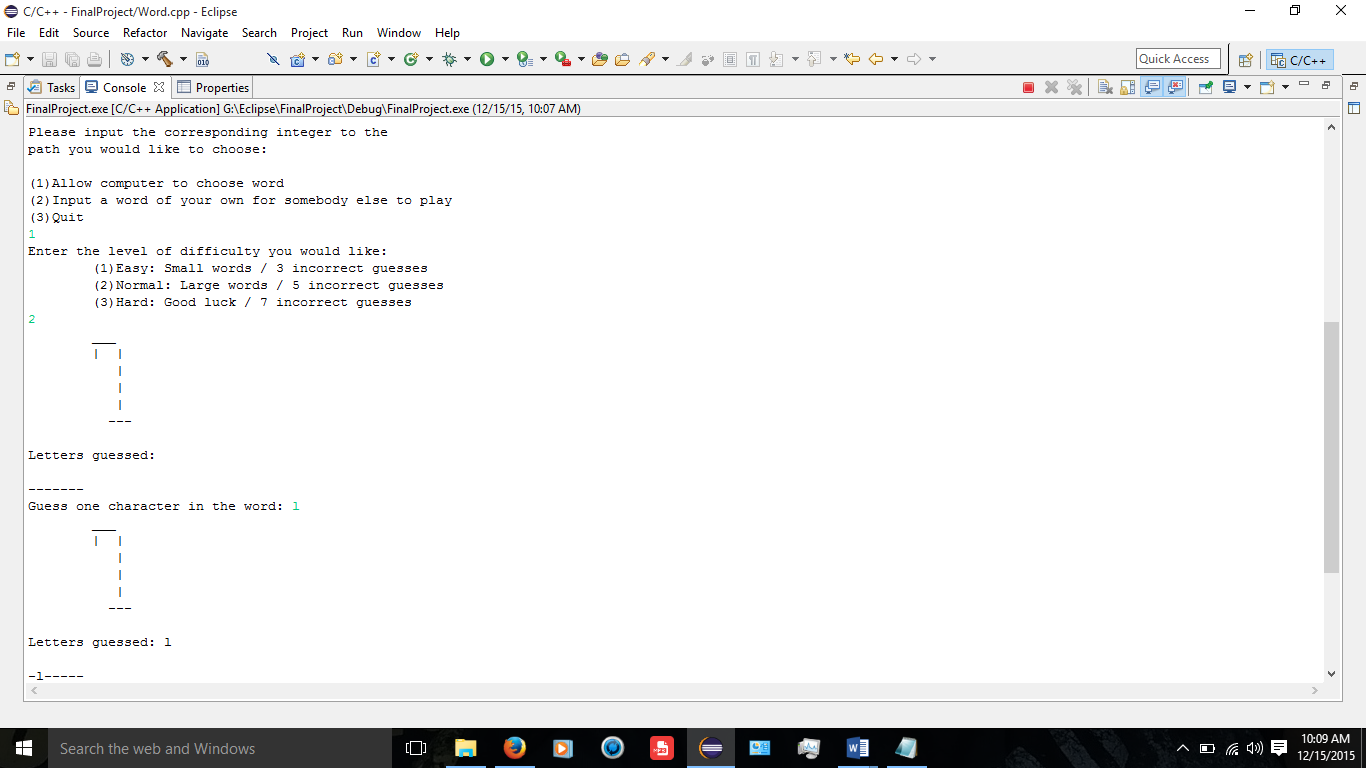
1008

**SAMPLE INPUT/OUTPUT**

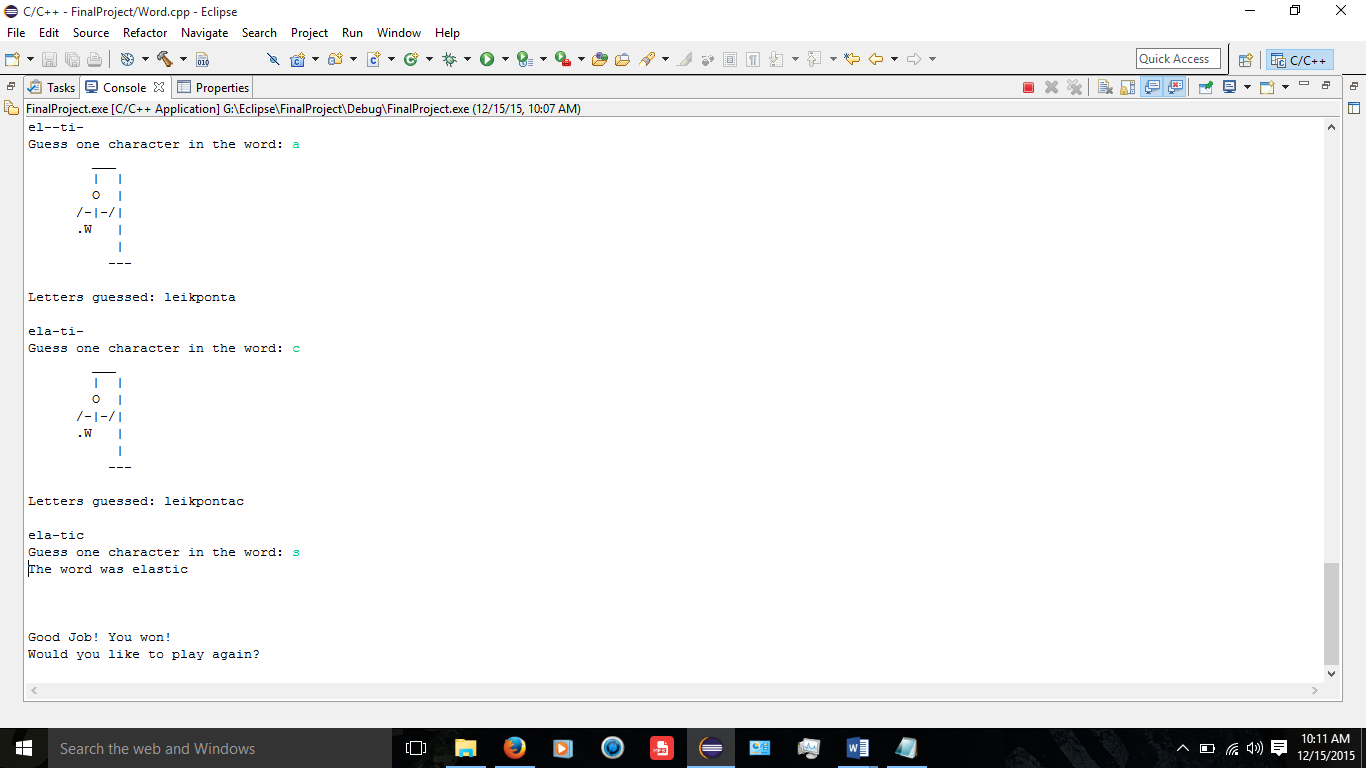
Beginning of program with two players chosen. First and last name entered along with location specification.



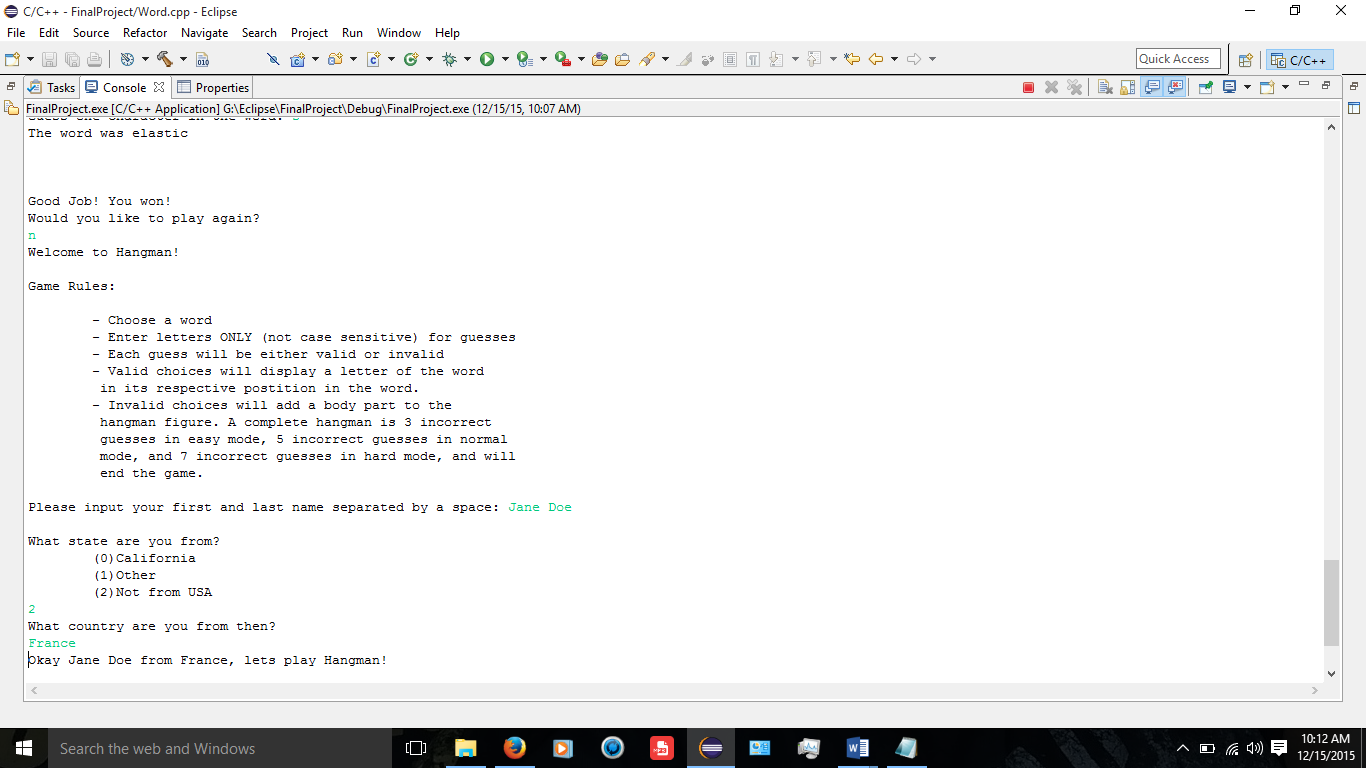
Computer generated word path chosen via integer and gameplay begins by entering characters. Hangman game board is displayed with each turn passed.



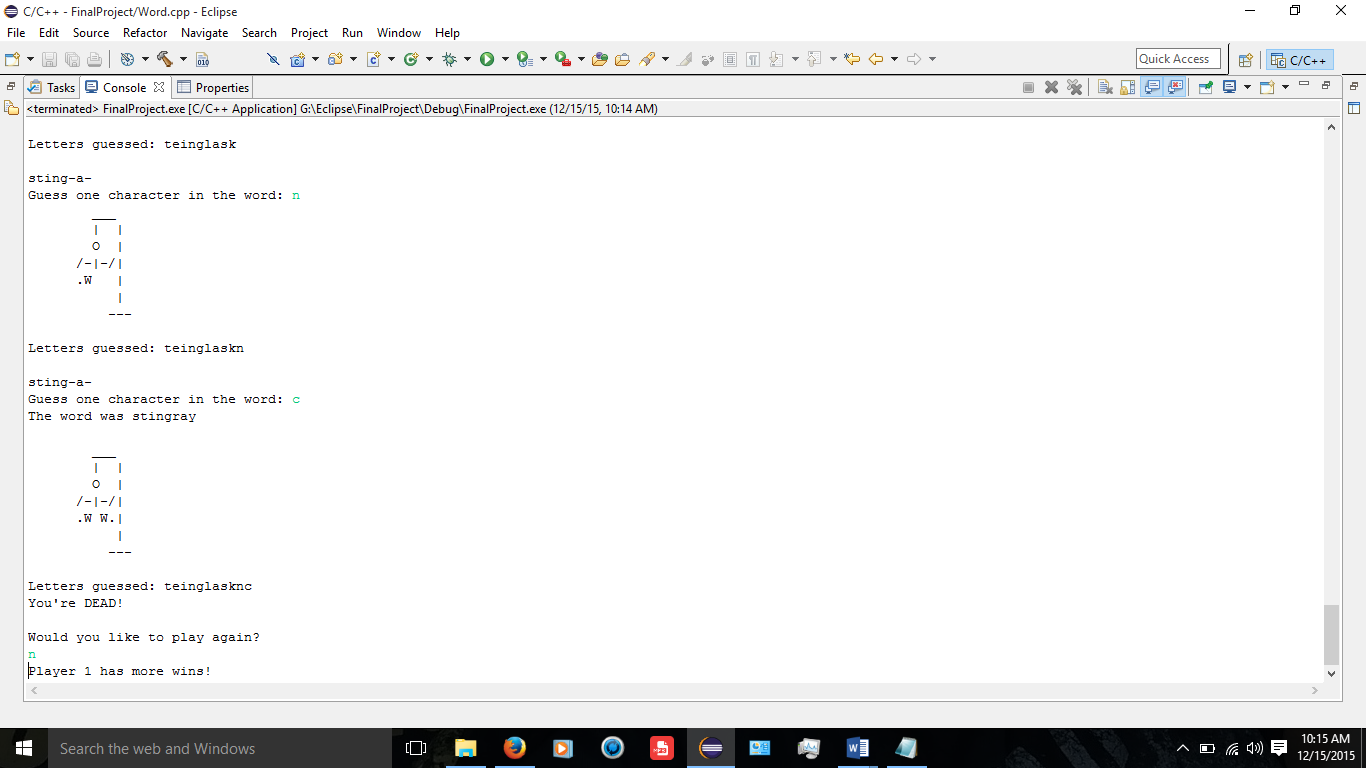
Incorrect guesses made and letter use tracked. Hangman character incremented based on difficulty. Winning condition met and tracked.



Player 1 chooses not to continue and player 2 begins their gameplay.



Player two completes hangman figure and console returns that player 1 had more wins.



**VARIABLES USED:**

* int
  + (static const) AR\_SIZE
  + diff used in Player class for difficulty setting
  + choice used in Winner class for menu choice
  + path used in Winner class for path choice
  + end used in Winner class for max moves
  + count1 used in Winner class for correct guesses
  + count2 used in Winner class for correct guesses
  + count3 used in Winner class for number of guesses
  + flag used in Winner class for first game phase
  + numWin used in Winner class for number of wins per player
  + chars used in Winner class for number of characters in word
  + incGues used in Winner class for number of guesses made
* Winner
  + p1 used in main for player 1’s class
  + p2 used in main for player 2’s class
* bool
  + win used in Player class for winning condition flag
* string
  + name used in Player class for user name
  + flName1 used in Word class for easy mode word file
  + flName2 used in Word class for normal mode word file
  + flName3 used in Word class for hard mode word file
* char
  + \*mark used in Word class for subscript of array guessed correctly
  + \*word used in Word class for array of character for game word
  + \*prgrss used in Word class for array of correct guesses
  + \*guesses used in Word class for array of character guesses
  + quit used in Winner class for choice of continuing game
* ifstream
  + inFile used in Word class for input file designation

**REFERENCES:**

Textbook: “starting out with C++ >>> From Control Structures through Objects” by Tony Gaddis (8th Edition)

Other than that I googled a few things and looked through stackoverflow.com mostly but no code was copied from anywhere.

**SOURCE CODE:**

/\*

\* main.cpp

\*

\* Created on: Oct 13, 2015

\* Author: David Hudkins II

\* Purpose: Execute the game hangman. User chooses either computer generated

\* word or the user chooses to input their own word. User then guesses

\* letters until word is completely deciphered or fails 6 times and

\* the hangman character is completed and the game is over.

\*/

//System Libraries

//User Libraries

**#include** "Winner.h"

//Function Prototypes

//Global Constants

//Program Execution

**int** **main**()

{

//Declare Variables

Winner p1;

Winner p2;

**int** numPs;

//Ask user one or two players

cout<<"One or two players?";

cin>>numPs; //INPUT- number of players

//Validate input

**while**(!cin >> numPs || numPs<1 || numPs>2) //Execute while input is broken

{

cin.clear(); //Clear keyboard buffer

cin.ignore(1000,'\n'); //Ignore characters until whitespace

cout << "Incorrect input. Please use an integer." << **endl** << **endl**; //Inform user of invalid input

cin>>numPs;

}

//Show game rules

p1.rules();

//Get information from user for game initialization

p1.setInfo();

//Play the game

p1.plyGame();

**if**(numPs==2)

{

//Show game rules

p2.rules();

//Get information from user for game initialization

p2.setInfo();

//Play the game

p2.plyGame();

//Tell users which player had more number of wins in the hangman games

**if**(p2>p1)

cout<<"Player 2 has more wins!";

**else**

cout<<"Player 1 has more wins!";

}

**return** 0; //Terminate Program

}

**---------------------------------------------------------------------------------------------------------------------**

/\*

\* Player.h

\*

\* Created on: Dec 1, 2015

\* Author: David Hudkins II

\* Purpose: Create a class Player to keep track of game player's data

\* which includes their name and location. This class also handles the

\* difficulty level of the game that the user chooses.

\*/

**#include**<iostream> //Console input/output streaming library

**#include** <string> //Allows use of string class

**using** **namespace** std; //Utilize standard namespace

//Structures

**enum** State {*CALIFORNIA*, *OTHER*, *NOTINUSA*}; //Data type for location

**struct** PlyInfo //Player information structure

{

string fstName; //Player first name

string lstName; //Player last name

**int** where; //Player location

};

**#ifndef** PLAYER\_H\_

**#define** PLAYER\_H\_

**class** Player

{

**protected**:

//Variables to be utilized by class Player and derived classes

**bool** win;

string name;

**int** diff;

**public**:

//Default Constructor

**Player**();

//Accessor Functions

**void** **rules**();

**int** **shwMenu**();

**int** **getMode**(){**return** diff;}

//Mutator Functions

**void** **setInfo**();

**void** **gmMode**();

//Destructor

**~Player**();

};

**#endif** /\* PLAYER\_H\_ \*/

**---------------------------------------------------------------------------------------------------------------------**

/\*

\* Player.cpp

\*

\* Created on: Dec 7, 2015

\* Author: David Hudkins II

\* Purpose: Define the member functions of the Player class.

\*/

**#include** "Player.h"

**#ifndef** PLAYER\_CPP\_

**#define** PLAYER\_CPP\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Player function is the constructor for the Player class

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Player::Player**()

{

win = 0;

diff = 0;

name = "0";

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* rules function accepts no parameters and displays the rules to the

\* game Hangman.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **Player::rules**()

{

//Display Rules

cout << "Welcome to Hangman!" << **endl** << **endl**;

cout << "Game Rules:" << **endl** << **endl**;

cout << "\t- Choose a word" << **endl**;

cout << "\t- Enter letters ONLY (not case sensitive) for guesses" << **endl**;

cout << "\t- Each guess will be either valid or invalid" << **endl**;

cout << "\t- Valid choices will display a letter of the word" << **endl**;

cout << "\t in its respective postition in the word." << **endl**;

cout << "\t- Invalid choices will add a body part to the " <<**endl**;

cout << "\t hangman figure. A complete hangman is 3 incorrect" << **endl**;

cout << "\t guesses in easy mode, 5 incorrect guesses in normal" << **endl**;

cout << "\t mode, and 7 incorrect guesses in hard mode, and will" << **endl**;

cout << "\t end the game." << **endl** << **endl**;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* shwMenu function accepts no parameters and displays the menu for the game

\* hangman. The player makes an integer choice through input and the function

\* returns that integer to the calling function.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**int** **Player::shwMenu**()

{

**int** choose;

//Display Menu

**do**

{

//Prompt user to input integer from menu

cout << "Please input the corresponding integer to the" << **endl**;

cout << "path you would like to choose: " << **endl** << **endl**;

cout << "(1)Allow computer to choose word" << **endl**;

cout << "(2)Input a word of your own for somebody else to play" << **endl**;

cout << "(3)Quit" << **endl**;

cin >> choose; //INPUT- Menu choice

//Validate input

**while**(!cin >> choose) //Execute while input is broken

{

cin.clear(); //Clear keyboard buffer

cin.ignore(1000,'\n'); //Ignore characters until whitespace

cout << "Incorrect input. Please use an integer." << **endl** << **endl**; //Inform user of invalid input

cin >> choose;

}

}**while**(choose != 1 && choose != 2 && choose != 3); //Execute until menu choice is valid

//Return menu choice to calling function

**return** choose;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* getInfo function accepts no parameters and asks user to input information

\* including their first and last name, as well as their state of residence.

\* It uses a structure PlyInfo to receive the players information. Enumerated

\* data type State is used to compare players choice of residence.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **Player::setInfo**()

{

PlyInfo Info; //Contains information about the player

string locatn; //Players geographical location

cout << "Please input your first and last name separated by a space: ";

cin >> Info.fstName; //INPUT- User's first name

cin >> Info.lstName; //INPUT- User's last name

cout << "\nWhat state are you from?" << **endl**;

cout << "\t(0)California" << **endl** << "\t(1)Other" << **endl** << "\t(2)Not from USA\n";

cin >> Info.where; //INPUT- User location

//Loop until input is not broken

**while**(!cin >> Info.where || Info.where > 2 || Info.where < 0)

{

cin.clear(); //Clear keyboard buffer

cin.ignore(1000,'\n'); //Ignore characters until whitespace

cout << "Incorrect input. Please use an integer." << **endl** << **endl**; //Inform user of invalid input

cin >> Info.where; //INPUT- User location

}

//Execute if user entered California location

**if**(Info.where == *CALIFORNIA*)

locatn = "California"; //Set location to California

//Execute if user entered other location

**else** **if**(Info.where == *OTHER*)

{

//Prompt user to enter their state

cout << "What state are you from then? " << **endl**;

cin.ignore(1000,'\n'); //Ignore until whitespace

**getline**(cin,locatn); //INPUT- User location

}

//Execute if user entered out of country

**else** **if**(Info.where == *NOTINUSA*)

{

//Prompt user to enter their country

cout << "What country are you from then?" << **endl**;

cin.ignore(1000,'\n'); //Ignore until whitespace

**getline**(cin,locatn); //INPUT- User location

}

//Execute if erroneous input was chosen

**else**

{

//Prompt user to enter their planet

cout << "If you aren't from the U.S. or any other location on this" << **endl**;

cout << " planet, then you must be an alien. What planet are you" << **endl**;

cout << " from, alien scum: " << **endl**;

cin.ignore(1000,'\n'); //Ignore until whitespace

**getline**(cin,locatn); //INPUT- User location

}

//Inform user of game start

cout << "Okay " << Info.fstName << " " << Info.lstName << " from ";

cout << locatn << ", lets play Hangman!" << **endl** << **endl**;

}

**void** **Player::gmMode**()

{

//Display options and prompt user for desired difficulty

cout<<"Enter the level of difficulty you would like: "<<**endl**;

cout<<"\t(1)Easy: Small words / 3 incorrect guesses"<<**endl**;

cout<<"\t(2)Normal: Large words / 5 incorrect guesses"<<**endl**;

cout<<"\t(3)Hard: Good luck / 7 incorrect guesses"<<**endl**;

cin>>diff; //INPUT- Difficulty level

//Validate input

**while**(!cin >> diff)

{

cin.clear(); //Clear keyboard buffer

cin.ignore(1000,'\n'); //Ignore characters until whitespace

cout << "Incorrect input. Please use an integer." << **endl** << **endl**; //Inform user of invalid input

cin >> diff; //INPUT- Difficulty setting

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* ~Player function is the destructor for the Player class

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Player::~Player**(){}

**#endif** /\* PLAYER\_CPP\_ \*/

**---------------------------------------------------------------------------------------------------------------------**

/\*

\* Winner.h

\*

\* Created on: Dec 7, 2015

\* Author: David Hudkins II

\* Purpose: Create Winner class to handle all data regarding the hangman

\* game. This class inherits two classes, Player and Word. The class

\* uses several character arrays for comparison and contains the actual

\* gameplay loop that executes to play and display the game hangman. After

\* the game is completed the class destructor is called.

\*/

//System Libraries

**#include** <cstring> //String operator library

**#include** <iomanip> //Formatting library

**#include** <stdlib.h> //For exit function

**#include** <new> //Need for bad\_alloc

//User Libraries

**#include** "Player.h"

**#include** "Word.h"

**#ifndef** WINNER\_H\_

**#define** WINNER\_H\_

**class** Winner : **public** Player, **public** Word

{

**protected**:

//Variables to be utitlized by Winner class and derived classes

**int** choice; //Menu choice

**int** path; //Determines how program will execute

**int** end; //Maximum number of moves

**int** count1; //Counter for correct guesses

**int** count2; //Counter for correct guesses

**int** count3; //Counter for number of guesses

**int** flag; //Flag for first phase to allow incorrect guess to increment

**int** numWin; //Holds the number of wins each player gets

string wrdPlay; //Word used to play hangman

**char** \*mark; //Subscript of array already guessed correctly

**char** \*word; //Word used to play hangman

**char** \*prgrss; //Keep track of game progression

**char** \*guesses; //Keep track of alphabetical guesses

**int** chars; //Number of characters in game word

**int** incGues; //Number of guesses player has made

**char** quit; //Choose to continue playing or quit

**public**:

//Default Constructor

**Winner**();

//Accessor Functions

**int** **getChoic**(){**return** choice;}

**void** **hangMan**();

//Mutator Functions

**int** **initGme**(**char**[], **char**[]);

**void** **setChoic**(**int** c){choice=c;}

**void** **plyGame**();

**void** **play**();

//Operator Overloading

**bool** **operator >**(**const** Winner &other);

//Destructor

**~Winner**();

};

**#endif** /\* WINNER\_H\_ \*/

**---------------------------------------------------------------------------------------------------------------------**

/\*

\* Winner.cpp

\*

\* Created on: Dec 7, 2015

\* Author: David Hudkins II

\* Purpose: Define member functions of the Winner class

\*/

**#include** "Winner.h"

**#ifndef** WINNER\_CPP\_

**#define** WINNER\_CPP\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Winner function is the constructor for the Winner class

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Winner::Winner**()

{

choice = 0;

path = 0;

incGues = 0;

count1 = 0;

count2 = 0;

count3 = 0;

flag = 0;

chars = 0;

end = 0;

numWin = 0;

wrdPlay = "0";

quit = '0';

//Verify memory sufficiency

**try**

{

mark = **new** **char**[*AR\_SIZE*];

}

**catch**(bad\_alloc &mark)

{

cout<<"Insufficient memory to allocate";

}

//Verify memory sufficiency

**try**

{

word = **new** **char**[*AR\_SIZE*];

}

**catch**(bad\_alloc &word)

{

cout<<"Insufficient memory to allocate";

}

//Verify memory sufficiency

**try**

{

prgrss = **new** **char**[*AR\_SIZE*];

}

**catch**(bad\_alloc &prgrss)

{

cout<<"Insufficient memory to allocate";

}

//Verify memory sufficiency

**try**

{

guesses = **new** **char**[*AR\_SIZE*];

}

**catch**(bad\_alloc &guesses)

{

cout<<"Insufficient memory to allocate";

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* hangMan function displays the hangman character based on the current

\* progression of the game at the time that it is called.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **Winner::hangMan**()

{

//Execute according to game difficulty chosen

**switch**(getMode())

{

//Execute if game is in easy mode

**case** 1:

//Display initial hangman figure

**if**(incGues==0)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<"|"<<**endl**;

cout<<setw(12)<<"|"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 1

**if**(incGues==1)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 2

**else** **if**(incGues==2)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-|-/|"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 3

**else** **if**(incGues==3)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-|-/|"<<**endl**<<setw(12)<<".W W.|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

**break**;

//Execute if game is in normal mode

**case** 2:

//Display initial hangman figure

**if**(incGues==0)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<"|"<<**endl**;

cout<<setw(12)<<"|"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 1

**if**(incGues==1)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 2

**else** **if**(incGues==2)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-| |"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 3

**else** **if**(incGues==3)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-|-/|"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 4

**else** **if**(incGues==4)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-|-/|"<<**endl**<<setw(12)<<".W |"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 5

**else** **if**(incGues==5)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-|-/|"<<**endl**<<setw(12)<<".W W.|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

**break**;

//Execute if game is in hard mode

**case** 3:

//Display initial hangman figure

**if**(incGues==0)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<"|"<<**endl**;

cout<<setw(12)<<"|"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 1

**if**(incGues==1)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 2

**else** **if**(incGues==2)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<" -| |"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 3

**else** **if**(incGues==3)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<" -|- |"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 4

**else** **if**(incGues==4)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-|- |"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 5

**else** **if**(incGues==5)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-|-/|"<<**endl**<<setw(12)<<"|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 6

**else** **if**(incGues==6)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-|-/|"<<**endl**<<setw(12)<<".W |"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

//Display if number of guesses is 7

**else** **if**(incGues==7)

{

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-|-/|"<<**endl**<<setw(12)<<".W W.|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

}

**break**;

//Default switch execution

**default**: cout<<"ERROR";

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* initGme function intitializes the arrays that are used in the execution of

\* the main gameplay. Two character arrays are accepted as parameters and the

\* array representing gameplay progress is initialized to the number of

\* characters in the gameplay word and then is initialized to blank spaces

\* representing their respective characters in their respective location in

\* the word. The function then returns the number of characters that are

\* contained in the gameplay word to the calling function.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**int** **Winner::initGme**(**char** word[], **char** progrss[])

{

//Declare variables

**int** numChar = 0; //Number of characters in gameplay word

//Loop until null terminator is encountered

**for**(**int** i=0; word[i]!='\0'; i++)

numChar = i; //Assign number of characters in gameplay word to variable

//Loop for number of characters in gameplay word

**for**(**int** i=0; i<=numChar; i++)

progrss[i] = word[i]; //Assign subscript of gameplay word array to subscript of progress array

//Loop for number of characters in gameplay word

**for**(**int** i=0; i<=numChar; i++)

progrss[i] = '-'; //Assign blank space to all subscripts of progress array

**return** numChar; //Return number of characters in gameplay word

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* plyGame function initializes the array that marks correct guess progression

\* at the beginning of a do-while loop. The menu choice is then displayed and

\* value chosen passed to the setChoic function. Paths are then executed based

\* said menu choice. The user can choose computer generated words, input their

\* own, or quit. If computer generated is chosen there are three difficulties

\* to choose from and the code will execute accordingly. The winning flag is

\* initialized to false and then the game begins. In the nested do-while loop

\* the play function is called to allow the player to input a guess. Then an if

\* loop is called to check for a correct guess. If so then an incorrect guess

\* is not counted and the player is allowed another guess. When the player gets

\* the word right, the win flag is set to true and the do-while loop terminates.

\* It also terminates if the user exceeds the maximum number of incorrect

\* guesses. When the loop terminates the user is asked if they would like to

\* continue. If yes the loop is repeated, if not the program is terminated.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **Winner::plyGame**()

{

**do**

{

//Initialize array that tracks user progression and correct guesses

**for**(**int** i = 0;i<*AR\_SIZE*;i++)

mark[i]='0';

//Display menu and allow user to choose gameplay settings

setChoic(shwMenu());

//Execute according to user's menu choice

**switch**(choice)

{

**case** 1: path = 1; //Computer generate word to play

**break**;

**case** 2: path = 2; //User input word to play

**break**;

**case** 3: cout << "Goodbye!";

**exit**(choice); //Terminate Program

}

//Computer generates word to play game

**if**(path == 1)

{

gmMode(); //Set difficulty

wrdPlay = genWord(getMode()); //Set game word to computer generated word

//Set number of incorrect guesses relevant to difficulty chosen

**switch**(getMode())

{

**case** 1:

end=3; //Easy mode-3 incorrect guesses allowed

**break**;

**case** 2:

end=5; //Normal mode-5 incorrect guesses allowed

**break**;

**case** 3:

end=7; //Hard mode-7 incorrect guesses allowed

**break**;

}

}

**else**

{

wrdPlay = usrWord(); //Set game word to user input word

end=5; //User mode automatically set to normal

}

**strcpy**(word,wrdPlay.c\_str()); //Convert string to character array

chars = initGme(word,prgrss); //Number of characters in the game word

win = **false**; //Initialize flag for win to false

//Loop for turns in game

**do**

{

//Call function to allow player to input character guess

play();

//Loop to check for winning condition

**for**(**int** i=0, chkWin=0; i<=chars; i++)

{

**if**(word[i]==prgrss[i]) //Execute if guess matches a character in the word

{

chkWin+=1; //Increment amount of letters guessed correctly

//Execute if location of score keeping is empty, indicating a new correct guess

**if**(mark[i]=='0')

{

mark[i]=word[i]; //Input and track correct guess

count1++; //Increment count to indicate correct guess this phase

flag=1; //Set flag to indicate at least one correct guess has been made

}

}

//Execute if number of correct guesses equals number of characters in the game word

**if**(chkWin==chars+1)

win=**true**; //Set winning flag to true

}

//Execute if current counter and holding counter evaluate as equal

**if**(count1 == count2 || flag==0)

incGues++; //Increment number of guesses

//Set counter for number of guesses to current counter for later comparison

count2 = count1;

}**while**(incGues < end && win==**false**); //Loop until max guesses or winning condition

//Inform user of what the game word was

cout << "The word was ";

//Loop to display game word

**for**(**int** i=0; i<=chars; i++)

cout << word[i];

cout<<**endl**<<**endl**;

//Execute if win flag is true

**if**(win==**true**)

{

cout << "\n\nGood Job! You won!" << **endl**; //Inform user of win

numWin+=1;

}

//Execute if win flag is false

**if**(win==**false**)

{

//Show completed hangman

cout<<setw(11)<<"\_\_\_"<<**endl**<<setw(12)<<"| |"<<**endl**<<setw(12)<<" O |";

cout<<**endl**<<setw(12)<<"/-|-/|"<<**endl**<<setw(12)<<".W W.|"<<**endl**<<setw(12)<<"|";

cout<<**endl**<<setw(13)<<"---"<<**endl**<<**endl**;

cout<<"Letters guessed: ";

//Show the letters already guessed

**for**(**int** i=0; i<count3; i++)

cout<<guesses[i];

//Indicate loss

cout<<"\nYou're DEAD!"<<**endl**<<**endl**;

}

//Prompt user to play again

cout << "Would you like to play again?" << **endl**;

cin >> quit; //INPUT- Play again or not

//Set guesses to 0 and counter for guesses to 0

incGues=0;

count3=0;

}**while**(quit=='Y' || quit=='y'); //Loop until user chooses to quit

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* play function deals with arrays that are the gameplay word array and the

\* gameplay progress array. The number of characters in the gameplay word is

\* handled as well. The function loops to display the progress and prompts the

\* user to input one character as a guess regarding the existence of the guess

\* as a character in the word. If the guess is correct, it is assigned to

\* its respective location in the progress array in as many places as it occurs.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **Winner::play**()

{

//Declare variables

**char** gChar; //INPUT- Character guess

//Show hangman progression

hangMan();

cout<<"Letters guessed: ";

//Show the letters already guessed

**for**(**int** i=0; i<count3; i++)

cout<<guesses[i];

cout<<**endl**<<**endl**;

//Loop for number of characters in gameplay word

**for**(**int** i=0; i<=chars; i++)

cout << prgrss[i]; //Display gameplay progress

//Prompt user to input guess

cout << **endl** << "Guess one character in the word: ";

cin >> gChar; //INPUT- Character guess

//Put guess in array of guesses

guesses[count3]=gChar;

//Loop for number of characters in gameplay word

**for**(**int** i=0; i<=chars; i++)

{

//Execute if guess is correct at gameplay word subscript

**if**(gChar == word[i])

prgrss[i] = gChar; //Assign guess to progress arrays respective location(s)

}

//Increment number of turns

count3+=1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* This function overloads the greater than operator to compare the number of

\* wins that each player had and returns which player has a greater number of

\* wins.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**bool** **Winner::operator >**(**const** Winner &other)

{

**if**(numWin > other.numWin)

**return** **true**;

**else**

**return** **false**;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* ~Winner function is the destructor for the Winner class

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Winner::~Winner**()

{

//Delete all dynamically allocated memory in the class

**delete**[] mark;

**delete**[] word;

**delete**[] prgrss;

**delete**[] guesses;

}

**#endif** /\* WINNER\_CPP\_ \*/

**---------------------------------------------------------------------------------------------------------------------**

/\*

\* Word.h

\*

\* Created on: Dec 3, 2015

\* Author: David Hudkins II

\* Purpose: Create class word to handle data the hangman program

\* uses regarding the word used to play the game.

\*/

**#include**<iostream> //Console input & output

**#include**<string> //Allows use of string class

**#include**<fstream> //File operations library

**#include**<ctime> //Time library

**#include**<cstdlib> //C-standard library

**using** **namespace** std;

**#ifndef** WORD\_H\_

**#define** WORD\_H\_

**class** Word

{

**protected**:

**static** **const** **int** *AR\_SIZE* = 100; //Size of array

ifstream inFile; //Input file designator

string flName1; //Input file name for easy

string flName2; //Input file name for normal

string flName3; //Input file name for hard

**public**:

//Default Constructor

**Word**();

//Accessor Functions

string **genWord**(**int**);

string **usrWord**();

//Destructors

**~Word**();

};

**#endif** /\* WORD\_H\_ \*/

**---------------------------------------------------------------------------------------------------------------------**

/\*

\* Word.cpp

\*

\* Created on: Dec 3, 2015

\* Author: David Hudkins II

\* Purpose: Define member functions of class Word

\*/

//User Libraries

**#include** "Word.h"

**#ifndef** WORD\_CPP\_

**#define** WORD\_CPP\_

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Word function is the constructor for the Word class

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Word::Word**()

{

flName1="0";

flName1="0";

flName3="0";

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* genWord function accepts two parameters that are a string array and

\* the size of that array. The function then assigns words from a file

\* to that array to be randomly seeded into a string variable that is passed

\* back into the calling function.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string **Word::genWord**(**int** difclty)

{

string gameWrd[*AR\_SIZE*]; //Array of words

**int** index = 0; //Index for array

flName1 = "word1.txt"; //Input file name

flName2 = "word2.txt"; //Input file name

flName3 = "word3.txt"; //Input file name

**switch**(difclty)

{

**case** 1:

//Open the word list file and assign words to array

inFile.open(flName1.c\_str()); //open file

//Execute until end of file or until max array size

**while**(!inFile.eof() && (index < *AR\_SIZE*))

{

**getline**(inFile, gameWrd[index]);//Assign word to array

index++; //Increment array index

}

inFile.close(); //close file

//Seed the random number generator according to time

**srand**(**time**(0));

//Return a random word from the word list file

**return** gameWrd[(**rand**() % 10)];

**case** 2:

//Open the word list file and assign words to array

inFile.open(flName2.c\_str()); //open file

//Execute until end of file or until max array size

**while**(!inFile.eof() && (index < *AR\_SIZE*))

{

**getline**(inFile, gameWrd[index]);//Assign word to array

index++; //Increment array index

}

inFile.close(); //close file

//Seed the random number generator according to time

**srand**(**time**(0));

//Return a random word from the word list file

**return** gameWrd[(**rand**() % 10)];

**case** 3:

//Open the word list file and assign words to array

inFile.open(flName3.c\_str()); //open file

//Execute until end of file or until max array size

**while**(!inFile.eof() && (index < *AR\_SIZE*))

{

**getline**(inFile, gameWrd[index]);//Assign word to array

index++; //Increment array index

}

inFile.close(); //close file

//Seed the random number generator according to time

**srand**(**time**(0));

//Return a random word from the word list file

**return** gameWrd[(**rand**() % 10)];

}

**return** 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* usrWord function accepts no parameters and allows the user to input their

\* own word for the hangman program that is no longer than 30 characters.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

string **Word::usrWord**()

{

//Declare variables

string word; //INPUT- User defined word for gameplay

//Prompt user to input game word

cout << "Please enter the word you would like to use to play hangman";

cout << " of no longer than 10 characters: " << **endl**;

cin >> word; //INPUT- User defined word for gameplay

**return** word; //Return user defined word for gameplay to calling function

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* ~Word function is the destructor for the Word class

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Word::~Word**(){}

**#endif** /\* WORD\_CPP\_ \*/

**---------------------------------------------------------------------------------------------------------------------**