“Pig” Dice Rolling

Game

Project 1

Victor Cuchilla

CSC-5

Dr. Lehr

2/4/17

RCC Winter 2017

Introduction:

For this assignment, I am going to program a simple dice game known as “Pig”. This dice game is called “Pig” because the goal of the game is for the players to hog the dice as they acquire points in order to win. The rules of the game can vary, but they generally have similar concepts. The players roll the die adding up their scores until a one is rolled, after which the next player will go, until all players have gone. The player who wins is usually the player with the most points at the end of the game or whichever player reaches a target score. In my version of the game there can be one to three players. This is because a single player may play the game with the goal being to reach a target score that the player declares at the beginning of the game in the program. If the player is playing alone they will play the game until either they win by reaching the score they set or if they roll a one, then the game is over for them and they can choose to play again. If more than one player is playing, then the game is played slightly differently. Instead of there being a win score, the game is won by the single player with the most points after each players’ turn as has ended. Although each player’s turn will end when a one is rolled, when the players rolls other digits on the die other than a one, there are other rules that adjust the score differently based off which number is rolled. The program explains the rules of the game to the players when it starts. After each game the program displays to the players who has won the game based off their scores and then it will ask if they would like to continue playing. Near the end of the program the player or players can choose to start the game over and if they do not wish to play the “Pig” game again then the program will exit. The program uses all the concepts that I have learned in CSC-5 from the Gaddis 8th Ed. Textbook chapters one through five.

Pseudocode:

//Comments, Name, Date, Purpose

//System Libraries

//iostream, Input output stream

//cstdlib, Random numbers

//ctime, Time to set the Seed

//string, Read strings

//cmath, Needed for certain math functions like pow

//iomanip, Manipulate output and calculation display

//fstream, To read data in from files and output data to files

//Declare Variables an Initialize nPlayers and their scores

//Number of players

//Number of turns each player has

// win score and each players score

//Set the win score to constant number and high

//Variable to keep playing

//Initialize scores to 0

//Name of winner string

//Instantiate and Open files

//Read input high score from HighScore file

//Explain the rules of the game

//Get the score to win from players

//Make sure win score is less than 2 million

//Display win score

//Format score to show will decimal point

//Get the number of players of the game

//Set the random number seed

//Play the game by -> Looping and take Dice statistics

//One Player game

//Declare character to roll and Prompt user to roll

//Get the key and start

//ignore enter in buffer key

//Call random number generator for the dice

//Get value from 1 to 6

//Score squared if die=2

//Ternary Operator for 3 and score equals 0

//score decrements if 3 is rolled

//add 4 and sqrt if die=4

//add 6 and double if die=6

//add the players score if they don't roll a 1

//Display the current # rolled and show current score

//Prompt use to roll again

//If player rolls a 1 declare loss

//If player wins then declare win

//Get the name of the winner

//Set High Score

//Write high score to file

//Decide if player wants to play again

//Replay by loop if player wants to play again

//End the 1 player game

//Two player Game

//Player 1 go

//Declare character to roll and Prompt user to roll

//Get the key and start

//ignore enter in buffer key

//Call random number generator for the dice

//Get value from 1 to 6

//Score squared if die=2

//Ternary Operator for 3 and score equals 0

//score decrements if 3 is rolled

//add 4 and sqrt if die=4

//add 6 and double if die=6

//add the players score if they don't roll a 1

//Display the current # rolled and show current score

//Prompt use to roll again

//If player rolls a 1 declare loss and player 2 go

//Compare scores

//Get the name of the winner

//Display which ever player is the winner

//Write high score to file

//Decide if players want to play again

//Replay if they wish to play again

//End the 2-player game

//Three player game

//Player 1 go

//Declare character to roll and Prompt user to roll

//Get the key and start

//ignore enter in buffer key

//Call random number generator for the dice

//Get value from 1 to 6

//Score squared if die=2

//Ternary Operator for 3 and score equals 0

//score decrements if 3 is rolled

//add 4 and sqrt if die=4

//add 6 and double if die=6

//add the players score if they don't roll a 1

//Display the current # rolled and show current score

//Prompt use to roll again

//If player rolls a 1 declare loss and have next player go

//Repeat until all players turns are over

//Compare scores

//Get the name of the winner

//Display which ever player is the winner

//Write high score to file

//Decide if players want to play again

//Replay if they wish to play again

//End the 3-player game

//If number of players is not a number end program or start over

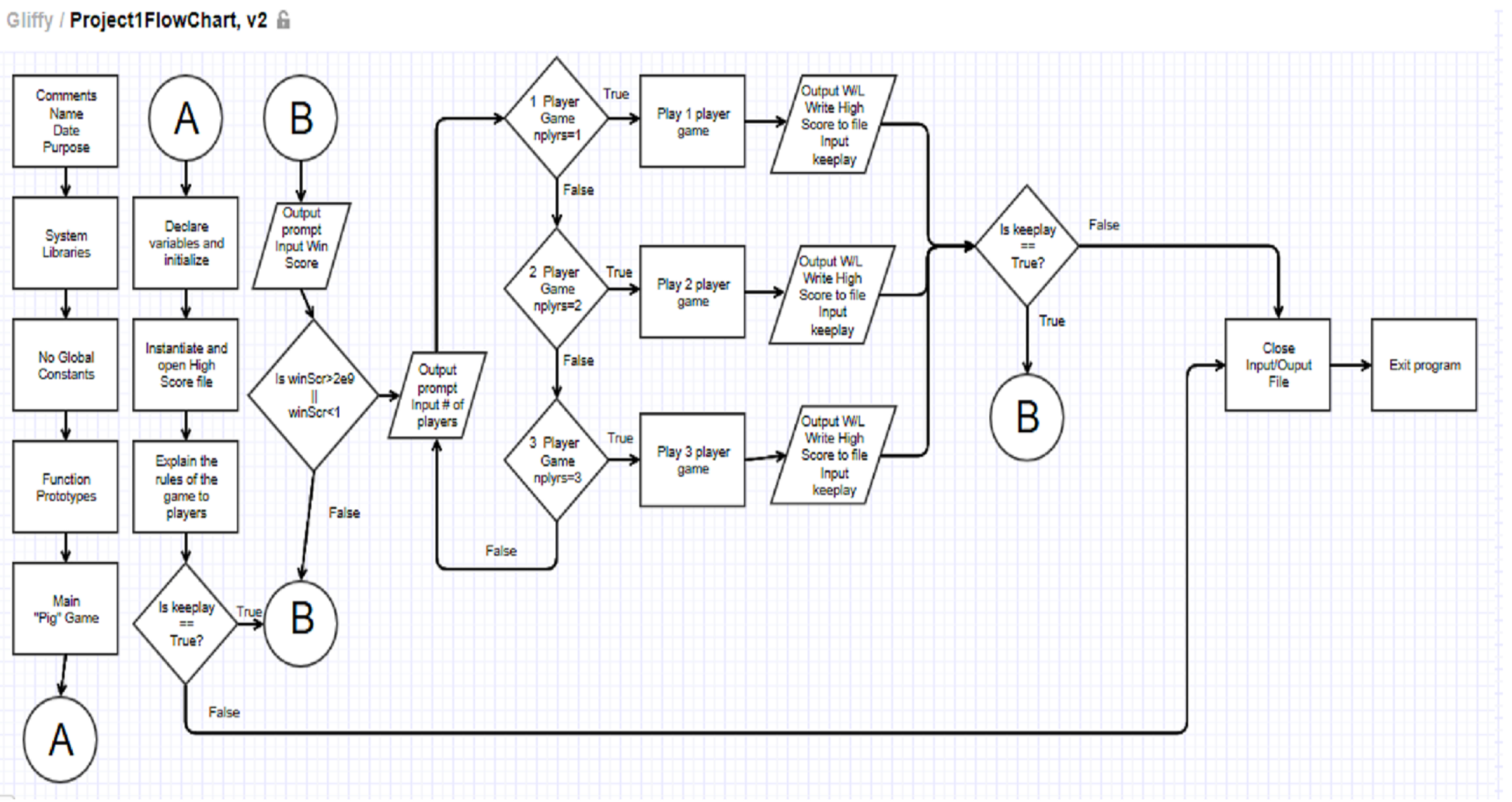
//Keep playing if player or players want

//Close files

//Close input file HighScore

//Close output file HighScore

//Exit the program



|  |  |  |  |
| --- | --- | --- | --- |
| Chapter | Section | Topic | Where in Code/Line# |
| 2 | 2 | cout | 50 |
|  | 3 | Libraries | iostream, iomanip, cmath, cstdlib, fstream, string, ctime |
|  | 4 | Variables/Literals | 30-35 |
|  | 5 | Identifiers | 30-35 |
|  | 6 | Integers | 30-35 |
|  | 7 | Characters | 95 |
|  | 8 | Strings | 39 |
|  | 9 | Floats No doubles | 34 |
|  | 10 | Bools | 35 |
|  | 11 | Sizeof\*\*\*\*\* |  |
|  | 12 | Variables 7 Characters or less | 30-35 |
|  | 13 | Scope\*\*\*\*\* |  |
|  | 14 | Arithmetic Operators | 67 |
|  | 15 | Comments 20%+ |  |
|  | 16 | Named Constants | 35 |
|  | 17 | Programming Style\*\*\*\*\* |  |
| 3 | 1 | cin | 67 |
|  | 2 | Math expressions | 100 |
|  | 3 | Mixing datatypes\*\*\*\*\* |  |
|  | 4 | Overflow/Underflow\*\*\*\*\* |  |
|  | 5 | Type Casting | 89 |
|  | 6 | Multiple Assignment\*\*\*\*\* |  |
|  | 7 | Formatting Output | 81 |
|  | 8 | Strings | 39 |
|  | 9 | Math Library | 106 |
|  | 10 | Hand Tracing\*\*\*\*\* |  |
| 4 | 1 | Relational Operators | 119 |
|  | 2 | If | 68 |
|  | 4 | If-else | 80 |
|  | 5 | Nesting | 68 |
|  | 6 | If-else-if | 74 |
|  | 7 | Flags\*\*\*\*\* |  |
| 68 | 8 | Logical Operators | 119 |
|  | 11 | Validating User Input | 68 |
|  | 13 | Conditional Operator | 104 |
|  | 14 | Switch | 92 |
| 5 | 1 | Increment/Decrement | 101, 104 |
|  | 2 | While | 304 |
|  | 5 | Do-While | 63 |
|  | 6 | For Loop | 101 |
|  | 11 | Files Input/Output both | 43-47 |
|  | 12 | No breaks in loops\*\*\*\*\* |  |
| \*\*\*\*\*Not required to show |  |  |  |