Project 2

Yahtzee

CIS 17C
Juan Castellon
12/15/18

Introduction

Title: Yahtzee

Yahtzee is point-based game played with 6 six-sided dice. The game is played by the players initially rolling the dice to see who goes first based on the highest rolling player. Players then take turns rolling the dice to see who can get the highest rolls and then choosing how to score themselves based on the rolls.

The point system is as follows:

Score 1s: Add up all 1s and add them to your total points.
Score 2s: Add up all 2s and add them to your total points.
Score 3s: Add up all 3s and add them to your total points.
Score 4s: Add up all 4s and add them to your total points.
Score 5s: Add up all 5s and add them to your total points.
Score 6s: Add up all 6s and add them to your total points.
Score 3 of a Kind: Multiply 3 of the same roll together.
Score 4 of a Kind: Multiply 4 of the same roll together.
Score Full House: If 3 dice are of one value and 2 are of another value, gain 25 points.
Score Short Straight: Gain 30 points if you roll 3 dice sequentially.
Score Long Straight: Gain 40 points if you roll 4 dice sequentially.
Score Yahtzee: Gain 100 points if all dice are same value. 50 points for each additional Yahtzee.
Score Chance: Add up all dice together.

The game lasts 10 turns and the player with the highest number of points is the winner.

Summary:

Project Size: ~820 lines of code

I did Yahtzee as my project because it's been an extension of my previous versions of Yahtzee ever since CSC-5. I implemented a binary tree to decide the order of the players when they roll. The binary tree uses a recursive sort to sort itself. I used a recursive function to roll the dice, with a variable being passed through the functions to determine when the recursive function should stop. I also used a recursive bubble sort to sort the dice array itself.

Recursive Sort Implementations:
Recursive Bubble Sort : Line 607
Recursive Tree Sort : Line 573

Recursion Implementation:
Recursive Dice Roll : Line 600

Binary Tree Implementation:
Binary Tree + Utility Functions : Lines 547 - 583

Pseudo Code:

--main.cpp--

--Function Prototypes--

Prototypes for starting the game
First player function for seeing which player goes first
Binary Tree prototypes
Dice utility prototypes

Ending the game:
Winner function for outputting the final scores and placements

--Function Prototypes End--

--Enter Main--

Set random number seed

--Declare Variables--

Vector of type Player

Game related variables:
Decision to play the game or not
Game lasts 10 turns, starts at 1

Member Function Variable :
Name, score, yahtzee counter, the 2 names

--End of Variable Declaring--

Initialize some variables that need to be initialized:

Introduction to game

Input decision to play or not
Input validation regarded decision

Exit if no, continue if yes

Use cin.ignore to skip to next line

Input user names

Roll to determine who goes first

Determine which person is player 1 and player 2

Turn do-while loop

Organize vector of objects for the first person to be on top

Output winner’s stats

Exit main

--Turn Function--
Declare and initialize some variables
Ask the user if they want to reroll
If they answer yes, while loop until conditions are not met
Output categories for scoring
Prompt user to choose
Validate input
Use scoring function
Output score and return it
Switch statement for all of the scoring possibilities:
Score 1s: Add up all 1s and add them to your total points.
Score 2s: Add up all 2s and add them to your total points.
Score 3s: Add up all 3s and add them to your total points.
Score 4s: Add up all 4s and add them to your total points.
Score 5s: Add up all 5s and add them to your total points.
Score 6s: Add up all 6s and add them to your total points.
Score 3 of a Kind: Multiply 3 of the same roll together.
Score 4 of a Kind: Multiply 4 of the same roll together.
Score Full House: If 3 dice are of one value and 2 are of another value, gain 25 points.
Score Short Straight: Gain 30 points if you roll 3 dice sequentially.
Score Long Straight: Gain 40 points if you roll 4 dice sequentially.
Score Yahtzee: Gain 100 points if all dice are same value. 50 points for each additional Yahtzee.
Score Chance: Add up all dice together.
Output score and return it
--End of Turn Function--

--End of main.cpp--

--Player.h--

Public :

Declare default constructor
Declare Constructor
Declare Destructor

Declare Setter Functions
Declare Getter Functions

Increment Yahtzee Function
Increment Chance Function
Increment Turn

Overloaded ++ Operator Function
Overloaded -- Operator Function

Private :
Declare a bunch of game variables (name, score, rolls, etc)

--End of Player.h--

--Player.cpp--

Constructor Function

Setter Functions
Getter Functions

Game Functions (AKA Increment Functions)

Operator Overload Functions for ++ and --

--End of Player.cpp--