Adam Wright

CS-162
Lab 3: Design and Reflection

Program Design

The problem in lab 3 is to create a game where the user can set the conditions of two dice and then to represent random behavior by using the c++ library pseudo-random function seeded with the time when the game instance is created, so that it returns a different pseudo-random value during each call. The other important problem in this lab is to create the behavior of the dice by using two die classes, and for the second to be a loaded die that outperforms the regular die but still returns values that are legitimate for the number of side on the chosen die.

Pseudocode:

cout a greeting

entering 1 continues the menu

entering 2 will exit

while loops to make sure that input is correct

main will create an instance of game

cin value for the number of rounds

press 1 for regular die and 2 for loaded - player 1

press 1 for regular die and 2 for loaded - player 2

cin number of die sides 4-20 - player 1

cin number of die sides 4-20 - player 2

array created for each player with the number of rounds as the size

if else blocks in game to create correct die or loaded die object instances

for loop for number of rounds

each loop calls the correct die random number method

each loop if one player's score is greater then increment player's score

print round's score

print game's score

print larger score's player as winner or print draw

call final menu function and return value to main to restart or quit

Test Table:

Testing Input	Expected Output	Actual Output
run program main	main prints first menu	main prints first menu
main calls menu	game prints press 1 or 2	game prints press 1 or 2
User enters 2	Program quits	program quits
User enters 1 to continue menu	program continues to the next prompt	program continues to the next prompt
user enters a char user enters a num other than 1 or 2 user enters a float	prompted to enter only 1 or 2 prompted to enter either 1 or 2 prompted to enter either 1 or 2	prompted to enter either 1 or 2 prompted to enter either 1 or 2 prompted to enter either 1 or 2
following Menu prompts	user asked for num rounds	user asked for num rounds
enters num between 1 - 1000 user enters a char user enters a num outside 1 - 1000 user enters a float	player one die type propmted to enter between 1 and 1000 propmted to enter between 1 and 1000 propmted to enter between 1 and 1000	moves to num columns propmted to enter between 1 and 1000 propmted to enter between 1 and 1000 propmted to enter between 1 and 1000
following Menu prompts	user asked for player 1 die type	user asked for player 1 die type
user enters 1 or 2 user enters a char user enters a num other than 1 or 2 user enters a float	variable set and menu continues prompted to enter only 1 or 2 prompted to enter either 1 or 2 prompted to enter either 1 or 2	variable set and menu continues prompted to enter either 1 or 2 prompted to enter either 1 or 2 prompted to enter either 1 or 2
following Menu prompts	user asked for player 2 die type	user asked for player 2 die type
user enters 1 or 2 user enters a char user enters a num other than 1 or 2 user enters a float	variable set and menu continues prompted to enter only 1 or 2 prompted to enter either 1 or 2 prompted to enter either 1 or 2	variable set and menu continues prompted to enter either 1 or 2 prompted to enter either 1 or 2 prompted to enter either 1 or 2
following menu prompts	user asked for player 1 sides	user asked for player 1 sides
enters an num between 4 - 20 user enters a char user enters a num outside 4 - 20 user enters a float	moves to player 2 sides propmted to enter between 4 and 20 propmted to enter between 4 and 20 propmted to enter between 4 and 20	mmoves to player 2 sides propmted to enter between 4 and 20 propmted to enter between 4 and 20 propmted to enter between 4 and 20
following menu prompts	user asked for player 2 sides	user asked for player 2 sides
enters an num between 4 - 20 user enters a char user enters a num outside 4 - 20 user enters a float	game prints propmted to enter between 4 and 20 propmted to enter between 4 and 20 propmted to enter between 4 and 20	game prints propmted to enter between 4 and 20 propmted to enter between 4 and 20 propmted to enter between 4 and 20
game prints	die type, sides are correct	die type, sides are correct
each printed round after rounds printed	round scores in sides range total and correct winner displayed	round scores in sides range total and correct winner displayed
end menu method called	play again or quit displayed	play again or quit displayed
User enters 1	return to game menu	return to game menu
User enters 2	quit	quit
user enters a char user enters a num other than 1 or 2 user enters a string	propmted to enter only 1 or 2 prompted to enter either 1 or 2 prompted to enter either 1 or 2	prompted to enter either 1 or 2 prompted to enter either 1 or 2 prompted to enter either 1 or 2

Reflection:

The main obstacle that I ran into when implementing the program was figuring out how to create the game instances. I originally planned to create the four possible die and loaded die instances at the beginning of the game and then to use setters to set the values of the two that would be chosen. I also tried to create the instances in if else blacks and then call them outside of the scope of the conditional that they were created in, but that

quickly because obvious that I was going astray. It became clear that I needed to declare two arrays and then to populate the arrays in the conditional where I created the die instances. Then I needed to fill the array with the game values and then they would persist outside of the scope of the conditional, so that the game values could be used in the rest of the program. Another question was how to improve the results of the loaded die. I decided to just up the lowest possible value into rand and then if it goes over the number of sides possible then it will return the highest number possible for the number of sides on the die. Another obstacle was that in feedback from project one I found that my input validation needed work. It accepted floats and stopped at the decimal rather than rejecting them. I turned to stack overflow and saw a function to make sure that only integers are being passed and not characters or periods or spaces. It takes a string as input and returns true if it only contains the numbers 0 - 9. The idea came from Ferdinando Randisi with the second most popular solution on this stack overflow thread - https://stackoverflow.com/questions/2844817/how-do-i-check-if-a-c-string-is-an-int. I read about other solutions, but this was the first that I really liked and it seems to work great. The solution does mean that I need to remember to reset the bool that I am using before each menu block or else all the later blocks will not be checked because they will all return true on the first pass.

One thing that has been on my mind is that I have not been able to reuse any class or function files in whole as if they were a library of my own. It would be nice if my input validation or menu functions could be refactored so that they could be reused without needing to retype them, but the output messages and the tests must be different each time. I plan to spend more time considering how these elements could be reused better.