Jason Lim

CS 162

Project 5: Sushi Adventures Reflection

Design

1. Modify Character Base Class
   1. Constructor now takes string for name
   2. Each character has attributes of attack, defense, armor, and strength points
   3. There will be public functions such as the constructor, a virtual attack and defend function, the roll, a getter for how much strength is left, and getter for name, as well as a set name function
      1. Will use rand for the roll
   4. Recover function where the player will have a character recover
   5. Change def function type from void to int
   6. Will be an abstract class
2. Modify Derived classes
   1. Defend function will now be int data type
3. Character Node struct
   1. Struct will have character pointer
   2. Previous and next pointer
   3. Character Node constructor which takes parameters of character selected and name of character and creates a char node
   4. Destructor for Character Node
4. Container Class
   1. Private member variables for head and rear pointers to character Node struct
   2. Constructor for container
   3. addChar to container
   4. isEmpty determination for Boolean logic
   5. GetFrontChar to get the front char in the linked list
   6. removeFrontChar to remove the front char from linked list
   7. addBack which takes a char node to add the char in the back of the linked list
   8. addFront which takes a char node to add the char to the front of the linked list
   9. printContainer which will be used to print out the loser container
   10. Container destructor
5. Menu class
   1. Create and display menu
      1. Display all five characters
      2. Allow user to choose size of team
      3. Allow user to use the same character for player 1 and player 2
   2. Display results each round
      1. Display Player 1 char vs Player 2 char and who won the round
      2. Continue repeating this until the tournament is over
   3. Repeat menu to play game/continue or quit (2 options)
6. Game class
   1. Container pointers for player1, player2, and loser container
   2. Integer values private for number for roster player 1 and player 2
   3. Create game constructor which takes the roster #’s from player 1 and player 2 user input
   4. makeRoster function which takes a player parameter and creates the roster for the player asking user input for choose character and name of character
   5. Public function for fighting with the characters in the tournament
   6. Destructor to destroy the pointer objects dynamically created of character pointers
7. Input validation
   1. To validate input particularly integer input to play game
8. Main function
   1. Seeds random and calls the menu to play game

Fantasy Combat Game Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Input by User | Expected Output | Actual Output |
| Input Validation | Choosing character only takes 1-5 | Error Message if not valid entry | Error Message if not valid entry |
| Menu Test “1” | 1 to play game | Plays game | Plays game |
| Menu Test “2” | 2 to exit game | Exits game | Exits game |
| Loop Menu | 1 to play game again after loop, 2 to exit | Menu loops after play game | Menu loops after play game |
| Test memory leaks | Check valgrind and enter in values for testing a few character battles before mem leak check | No mem leaks | No mem leaks |
| Add own names | Accept any string for naming chars | String output | String output |
| Enter size of team | Allows up to 10 max, 1 min for team | Allows 1 to 10 | Allows 1 to 10 |
| Allow more than 1 same type char select | Can select more than 1 same type char | Allowed no errors | Allowed no errors |
| Lineup should be what user chars picked | Lineup if choose Barb Team A then Barb Team A shows up | All chars show up in lineup | All chars show up in lineup |
| Fighter at front of lineup battles | Front fighter fights the first round and then the next fighter in front fights | Front fighter fights | Front fighter fights |
| Winner gets put back in lineup | test 3 v 5 with 3 (blue men team A) and 5 barbarians on team B | Winner gets put back into lineup and healed | Winner gets put back into lineup and healed |
| Loser gets put in loser container | Test loser container print after battle | Loser container shows losers last loser to earliest loser | Loser container shows losers last loser to earliest loser |
| Game ends when no more fighters | 1 blue men vs 1 barbarian, game ends when barbarian dies | Game ends | Game ends |
| Display type of character name of 2 fighters and who won combat | 1 vs 1 make sure it displays char name, type of char, and who won combat (Blue men Team A vs Barbarian Team B) | Displays Blue Men Team A vs Barbarian Team B and Blue Men Team A wins | Displays Blue Men Team A vs Barbarian Team B and Blue Men Team A wins |
| Display final score of team after win and who won | 1 blue men vs 1 barbarian | +1 team A, 0 team B, Team A wins | +1 team A, 0 team B Team A wins |
| 10 v 10 | Max 10 v 10 | Able to be played through | Able to be played through |

Reflection Fantasy Combat Tournament

The project was a good introduction into using Linked Lists, stacks, and queues in order to create data structures for an existing project. I learned how to stack up objects as well as queue them up for fighting and playing the game. I also learned how tournament style games worked. I made new classes this time around such as a Container class to hold my containers, and leveraged knowledge from Lab 6 and 7 in Linked Lists to create the Container class and a CharacterNode class. I gained appreciation for Linked Lists and their ability to point to different nodes and create linked data structures. It was interesting to compare how the first few projects used arrays and some of the labs used vectors to the linked Lists we used in Project 4. Mostly, I followed the script and design to its full potential. I did encounter some problems such as the fighter queue not working correctly, but I fixed that using the correct pointer values.

At the beginning of Project 4, I outlined all the classes and conducted a design overview, created test cases, and created a diagram for class hierarchy for my Container class. I made sure to follow the design carefully and created the node and container classes so that they could be used to store character objects. I also made sure that all my derived classes, character class, menu, and input validation imported correctly from the previous Project 3. I also added the recovery function to my Character class and other functions in Game.cpp.

I modified the character bass class so that the constructor now took a string for a name. I also made sure that each character had the same attributes as in Project 3. I made sure that the recover function would work when the player defeats an enemy and recovers based on the damage taken during the fight. I also changed the def function to int from void in order to use it later on in my calculations. In the derived classes, I made sure that I removed the cout statements that were used to showcase strength and other attributes during a fight since this would not be needed anymore in Project 4. I also made sure that all the def functions in the derived classes were changed to int data type instead of void.

I created a character node struct that would have a character pointer and a previous and next pointer. I also made a Character Node constructor which took parameters such as the character selected, the name of the character, and created a char node. I created a destructor for the character node as well to destroy the dynamically created memory before the program ended.

In my Container class, I created private member variables for head and rear pointers that would point to the characterNode struct nodes. I created a constructor for my container as well as the interface for the container such as addchar, isEmpty, getFront, removeFront, addBack, addFront, and printContainer All of these functions were standard from Lab 6 and Lab 7 except they would be implemented so that we could use them for objects instead of existing data types such as integers. I modeled these functions after the functions I did in Lab 6 and Lab 7. I also made sure that there was a container destructor to destroy any dynamically created memory.

In my Menu class I made sure to create and display a menu and allow the user to choose the size of the team. The size of the team should be different or could be the same for Player 1 and Player 2. I also allowed the user to be able to select the same character for player 1 and player 2 and duplicate characters on the same team. I displayed the results of each round and displayed who won the tournament player 1 or player 2 by keeping track of the score. I continually looped the menu every time the tournament ended.

The game class now had pointers for player 1, player 2, and a loser container. The integer values for number for rosters for player 1 and player 2 would be private and would determine what the actual number of characters each team would choose. The game constructor would create a roster or container for the queue for character objects from player 1 and player 2 user input. The makeRoster function would create the name of the character as well as make the player choose the character. There would be a destructor to destroy the pointer objects of dynamically created character pointers

While debugging the project, I found issues such as the queue not working correctly so I fixed that by making sure that each character was being pointed to one after the other in ascending order. The issue was it was behaving like a stack and queue at the same time because there was something wrong with my pointer. I also did not know what to do with the number for roster for player 1 and player 2 as I originally thought that both teams had the same roster size and it would only be inputted once. I eventually created two private variables in the class for number for roster and those would then be used to determine the size of the team for both players.

Overall, I believe my design and debugging helped my code and my understanding of stacks, queues, and linked lists. I did not struggle as much in this project as I did at the beginning of lab 6 and since lab 6 and lab7 were very relevant to Project 4, I had an easier time with Project 4 in general. This project was a fun exercise into expanding existing projects and helped me understand data structures in more depth.