Sean Reilly

Reflections – Assignment 4

Design Description

My initial thought was to use my code from the previous assignment and just add on to it to create the tournament but it became so complicated and huge that I couldn't even understand it anymore so I just decided to start from scratch, using bits of my old code. The first program kept crashing and was something like 3,000 lines so it was really difficult to find my errors as I was getting no compiler errors and nothing was obvious as to the issue. I also saw your comment from the previous week about how it would be nice if it was more automated and you didn't have to hit yes or no every time you wished to attack. This new design got rid of that old style and really abandoned a lot of last week's program which was risky because I got a 98 on it. I also decided to remove the dice class. This gave me less flexibility on the surface, and more code writing, but it worked out as I could customize it easier to specific creature types within the game. To put it more simply, the design has 3 files. A main, which houses the menu, the implementation of the fighting and the results. The type header, which houses the data for the characters you can play with. And the type class, which is the implementation file for the type header file.

Test Plan

I ended up writing 2 different programs for this assignment. My testing plan was the same for both and has been the same throughout the course. I isolate sections of code, make sure it works, then implement it into the larger picture.

Test Results

For the first iteration of the program, I got it to run but it didn't work properly. I used a nested for loop and it automatically shut down the program. Here is a section of the code I wrote:

```
switch (choice)
        {
        case 1:
        {
            cout << "How many characters on each team? ";
            cin >> teamTotal;

            for (int i = 0; i < 2; i++)
            {
                team = i + 1;

                cout << " " << endl;

                cout << "Thank you! Creating character number " << i + 1 << " now" <<
endl;

                for (int j = 0; j < teamTotal; j++)

                {

                    cin.ignore();

                    cout << "What would you like your character to be called?" <<
endl;

                    getline(cin, name);

                    cout << "Please pick the class you'd like to be " << endl;




                    cout << "Press 1 to be a goblin" << endl;

                    cout << "Press 2 to be a barbarian" << endl;

                    cout << "Press 3 to be the reptile people" << endl;

                    cout << "Press 4 to be the blue people" << endl;

                    cout << "Press 5 to be the shadow" << endl;

                    cout << " " << endl;

                    cout << "Please enter your choice now ";

                    cin >> character;

                }
```

This was pretty standard for me so far in this class but depending on how many characters you chose, it would shut the program down immediately after. I tried isolating it, posting on stack overflow, and I rewrote it so many times that I about smashed my computer. I decided to just start from scratch after this, because if the menu wasn't even working I literally could not check any other aspect of the code. Frustrating, but it gave me a chance to learn some new tricks.

For the second and final iteration I found a way to check if the program is within the parameters specified without using such a complicated while loop. This fixed the menu issue that I was having previously. I then copied a lot of my code from the previous assignment in order to segregate the types of characters away from the menu and everything else needed to implement the code. I then went down the list of the required parameters and implemented as it started working. Everything worked out really well. Definitely my best program to date.