Project 2

Christina Torres

Fall 2023

CMSC 330

Professor Alin Suciu

**Project Approach**

The approach to this project was quite interesting. My knowledge of C++ is very limited As well as h files. After downloading the assignment information and taking a look at the instructions, my next step was to figure out how to run the skeleton code. There was some research involved. Eventually I ended up in y MINGW64 and running this command to get ./project2 to be ready to run:

xtina@xtinacodes MINGW64 c/Users/xtina/OneDrive - xtina.codes/UnicornMagic/CMSC 330 Advanced Programming T2/CMSC330-Project2-Torres

$ g++ -o project2 project2.cpp expression.h operand.cpp parse.cpp subexpression.cpp symboltable.cpp variable.cpp

After that I went to my VS code terminal to run the program. I also ensured that the input file was located with the correct path. From there I looked at the files and started finding my way around them to see the flow and clear the segmentation error. I then added a line counter and value to see where I was at. From there I took it step by step for each operation necessary.

The binary arithmetic operators were a little easier. I looked at the given minus and plus files for reference of syntax. I then looked up how to complete the necessary operations with C++. The relational operators were trickier because I had to figure out how to take in more tokens. I was able to do that with a few if statements. This was fun to figure out, and I spent a little too much time a few nights because it was so interesting.

Finally, I attacked the variable changes. I looked at the find method to see how to loop through a vector. Then I just changed the value from there. I looked up how to do try catches and had really good results returning integers. I was confused a bit with the instructions but was able to manipulate the try catch as I think it may have directed. Had a lot of fun with this as well. Figuring out how to access these things and when to access them was really interesting! Thank you!

**Test Cases**

|  |  |  |
| --- | --- | --- |
| # | INFO | P/F |
| 1 | **Unary expression.**  **Input:**  ((a + 4) ~), a = 3;  **Expected Output:**  Value = -7.000000 **Screenshots**: | PASS |
| 2 | **Binary arithmetic multiplication and decimals.**  **Input**:  ((y \* 2) + (x - 5)), x = 5.5, y = 10;  **Expected Output**:  Value = 20.500000 **Screenshot**: | PASS |
| 3 | **Binary arithmetic modulo and decimals.**  **Input**:  (6 % b), b = 2.5; **Expected Output**:  Value = 1.000000 **Screenshot**: | PASS |
| 4 | **Find the minimum of the left and right, and decimals.**  **Input**:  (10 < b), b = 4.3; **Expected Output**:  Value = 4.300000 **Screenshot**: | PASS |
| 5 | **Find the maximum of left and right, and decimals.**  **Input**:  (10 > b), b = 4.3; **Expected Output**:  Value = 10.000000 **Screenshot**: | PASS |
| 6 | **Find the average of left and right, and decimals.**  **Input**:  (e & 8), e = 5; **Expected Output**:  Value = 6.500000 **Screenshot**: | PASS |
| 7 | **Ternary:** If the expression to the left of the operator ? is not 0, the value of the expression is the value of the first expression after the operator  **Input**:  (f ? 1 : 2), f = 1; **Expected Output**:  Value = 1.000000 **Screenshot**: | PASS |
| 8 | **Ternary:** If the expression to the left of the operator ? is 0, the value of the expression is the value of the second expression after the operator  **Input**:  (f ? 1 : 2), f = 0; **Expected Output**:  Value = 2.000000 **Screenshot**: | PASS |
| 9 | **Quarternary:** If the expression to the left of the operator # is less than 0, the value of the expression is the value of the first expression after the operator #.  **Input**:  (g # 1 : 2 : 3), g = -3; **Expected Output**:  Value = 1.000000 **Screenshot**: | PASS |
| 10 | **Quarternary:** If it is equal to 0, the value of the expression is the value of the second expression after the operator #.  **Input**:  (g # 1 : 2 : 3), g = 0; **Expected Output**:  Value = 2.000000 **Screenshot**: | PASS |
| 11 | **Quarternary:** If it is greater than 0, the value of the expression is the value of the third expression after the operator #.  **Input**:  (g # 1 : 2 : 3), g = 5; **Expected Output**:  Value = 3.000000 **Screenshot**: | PASS |
| 12 | **Handling the same variable:**   * Reported as an error. * Replace the value of the previous.   **Input**:  (aa + 1), aa = 1, aa = 2; **Expected Output**:  Oops! Updating variable!  Value = 3.000000 **Screenshot**: | PASS |
| 13 | **Bonus! Just for fun! Motivational comments after each 😊** | PASS |

**Improvements and Learned Lessons**

Oh man I learned many lessons! There were many lessons about the syntax of the language itself. There were other lessons about the structure of the project and flow of data. The main takeaway was problem solving. This project gave me a slight boost of confidence that will help me in my following stages of learning.

Given my limited knowledge of C++ and header files, I found myself investing significant time in learning these concepts during the project. I will continue to proactively acquire a more solid foundation in C++ to improve this.

Secondly, the project highlighted the critical role of efficient debugging techniques. Segmentation errors posed formidable challenges, emphasizing the importance of effective debugging tools. To improve in this area, I will continue familiarizing myself with debugging C++ and header files. The lessons I learned from this project I will certainly take with me throughout my career. Thanks for everything this semester! You’re awesome!