Anthony Gutierrez

Professor Dr. Benyamin Ahmadnia

CSE 5700

April 20th, 2024

Midterm Project

The goal of this assignment was to implement a scanner and parser for a simplified language C--, designed to identify tokens and parse simple expressions. The scope of the project was basic operations, identifiers, and arithmetic expressions, while ignoring complex features like pointers or advanced structures.

My approach was simple, to create a scanner responsible for tokenizing the input code by identifying keywords, identifiers, literals, etc. I then created a parser that was responsible for parsing the tokenized input for identification of syntactic structures and evaluating expressions. I originally had a basic scanner and parser and expanded as I needed more tokenization. I found this process difficult in never using a language such as C--. Once the scanner was complete, I worked on the parser and expanded as much as I could with the resources I had.

The scanner included code that would skip comments and whitespace. I dealt with errors by having them be sent back to the parser and have an "x" take its place as a line and column indicator which helped me better understand my projects issues. A big issue I had with the project was handling whitespace, it took me a long time to figure out how to ignore whitespace and to be frank, my code doesn't always skip whitespace in certain instances. Something I was very proud to accomplish was the skipping of comments, I had no issues skipping comments and that was something I was proud of. I continued to tweak the parser and scanner for whitespace, and I got to a good spot where it does skip most whitespace. I had an issue with whitespace that was in between two identifiers which I didn't have a chance to fix I believe.

Once I had the parser and scanner able to compile, testing input was a breeze. I had it how the assignment asked us to complete it which looked like this "cat input_2.c-- | ./compile". I was successful in testing 2 different inputs that had checked multiple tokens and scenarios and came out with favorable findings which will be included in the zip file, but I will leave screenshots at the end just in case. (Examples will be attached below). Further improvements and continuous learning would work wonders for this assignment. This was the first time I have ever done this, and it was a nice learning experience for me. Once I have Windows, I would attempt to use Lex and Yacc instead of something manual which would really help for future projects. Overall, the project was a great learning experience, and I would really appreciate tips on how it could have been handled better.

Example #1.

Input file:

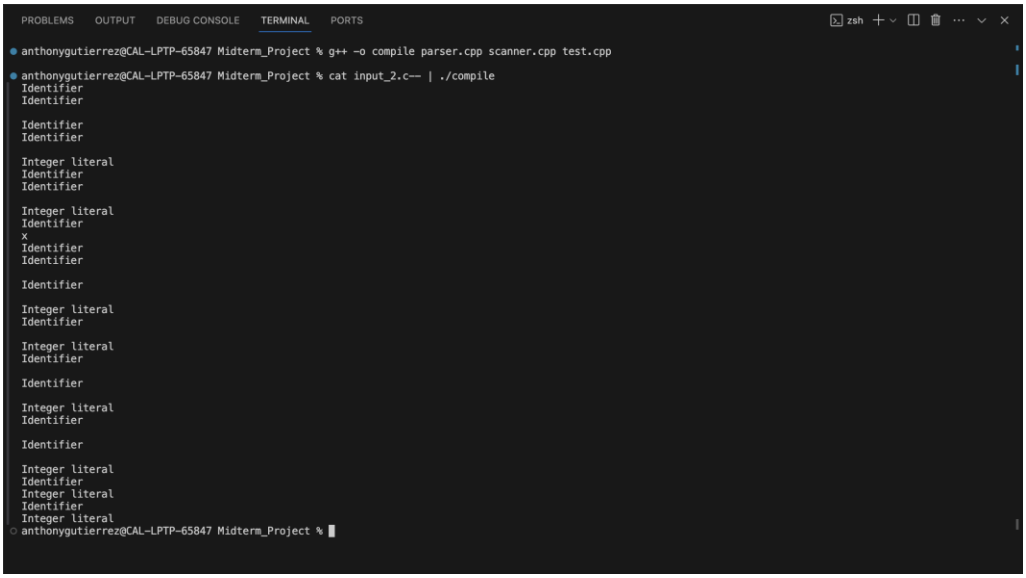
input_2.c--

```
int main() {
    int x = 5;
    int y = 10;

    if (x < y) {
        x = x + 1;
    } else {
        y = y - 1;
    }

    return 0;
}
```

Output:



(Example 2 included below)

Example #2:

Input file:

input.c--

```c
#include <stdio.h> // Preprocessor directive

int main() {
    int x = 10; // Variable declaration and initialization
    int y = 20;
    int z;

    if (x < y) { // If statement with comparison
        z = x + y; // Assignment and arithmetic operation
    } else {
        z = x - y;
    }

    // Simple loop to demonstrate a while structure
    while (z > 0) {
        z = z - 1;
    }

    printf("Result is %d\n", z); // Function call with string formatting

    return 0; // Function return statement
}

```

Output: