Paulo Garcia, Roel
2/2/2016
Assignment 0: Tokenizer
ReadMe

Our tokenizer project composes of mainly 6 elements to work properly. The first element is the Tokenizer Structure. The second element is the parser function that is used by our GetNextToken function. The next three elements are put to use inside the parser function. They are the checkDecimal, checkWord, and checkSpecial functions.

1.) Tokenizer structure -The tokenizer structure composes of:

```
    char * input;
    int track;
```

when we use the TKcreate function, we set the track to 0, malloc the size of the string input and copy it to the char * input.

2.)Parser - The GetNextToken takes in the parameters of a TokenizerT structure and then manipulates its elements with the parser function in order to keep track of tokens and traverse through the input array. The parser works by going through a given input's characters 1 by 1, skipping any white spaces it comes a cross and keeps a count i, which starts at the tk element track= 0.

```
parser(input,tk->track) //track initially equals to 0.
```

Once it comes across a character that is not a white space, it then sends the address of and value i and the input string into 1 of three functions. Once the function ends, track's value is changed to i, and a new token is returned depending on which function is given.

3.) checkDecimal - if the character was a number,

checkDecimal would be called. This functions increments i for character in input[i] that matches, a number. If the first number when placed in i is a 0, it checks for either hexadecimal or octal numbers. Else it just prints decimal integer 0.

4.) checkWord - increments the i counter for every character that is part of an alphabet , number, or underscore. Once recorded, it stores the count subtracted by the original count amount of the characters into a new token. This function then checks this token if its a keword in the checkKeyword function.

5.) checkSpecial - essentially does the same thing as the other two functions, except it checks for special characters. If there is a possibility that two special characters can be considered, then it checks for that too. (e.g. plus equal "+=")

6.) The while loop in the main function, repeats the GetNextToken function for a created Tokenizer till tk's track is equal to the length of the input string.

****
****
Special Cases: Tokenizer will parse mostly every character and integer and will rarely consider a token as an unknown token or error.

```
./tokenizer "0902 07.2.1234"
decimal integer "0"
decimal integer "902"
octal "07"
period "."
float "2.1234"
```

Tokenizer will parse variable and function names according to the c language.

```
./tokenizer "Hello World Hello_world123 123abc"
word "Hello"
```

```
word "World"
word "Hello_world123"
decimal integer "123"
word "abc"

Tokenizer recognizes keywords in the C language

./tokenizer "char int sizeof char1 integer"
character keyword "char"
integer keyword "int"
sizeof keyword "sizeof"
word "char1"
word "integer"
```