Michelle Wilderotter
Jesse Strivelli
Read Me

This program accepts a string of characters from the command line. First it checks for the correct number of arguments (2). Next it allocates memory for the root and next node of a linked list data structure, which will hold the tokens. Each TokenizerT struct holds two arrays of characters. One array holds the token, and the second holds the type of the token. The program calls the function TKCreate.

TKCreate accepts the string of characters, and iterates through each first character of a token, and calls other functions to put the full token into the data structure. Comparing the character using if else statements, the program breaks down the character into either the start of a word, number, operator, or escape character. The operators with possibly multiple characters have their own functions, and the single character operators are in one function. Once the first character is matched to a word, number, operator or escape, it goes into a separate function depending on which it is. If the character is white space it iterates to the next char.

In these secondary functions, the rest of the token is iterated through and put into the node. The types are also assigned to each token. For words, the function returns when the token reaches a non-letter character. After it returns, the words are checked against "c keywords" by calling the ckeywordcmp function. If a keyword is found the type changes from word to c keyword.

For the operator functions, once the longest possible operator is checked for, the function returns. If the operator is not caught with the beginning functions, its checked in the "other" function.

For the escape characters, if a character has a hex value that is the ASCII value of an escape character it gets sent to the escape function. This function does not affect the tokens or data structure it just prints out the escape character and an error message.

For the number functions, if the first character was zero, it goes into the zero function where it checks whether the number is a 0, hexadecimal, octal, or a float. Depending on the type, it will go through the string until it finds a character that is not allowed in the token, and then it will return. If the float or hex number is malformed the function back tracks in the string and the token will just be a zero. The next token will either be the operator "." Or "x", but no longer part of a hex or float number. The same algorithm is applied for numbers 1-9 in a separate function.

After the string is iterated through the function calls the print method, which goes through the data structure. Each node is printed by calling the TKGetNestToken method. Next the TKDestroy method is called to free each node after printing it.