**Documentation**

This document aims to explain the functions used in **scanner.l** file.

The input to the file is a C program and output is a stream of tokens and symbol table.

**Rules included:**

1. The rules defined in decreasing order of priority are tab space, newline, int, char, void, while, if, else, printf, scanf, return, Integer constant, identifier.
2. This is followed by rules that handle various other symbols and operators with suitable precedence order to ensure correct functioning of the lexer.

E.g. rule for ‘**%c**’ has higher priority than ‘**%**’. If it were the other way round, scanner would recognise **%** as operator and **c** as identifier. It will never recognise **%c** as a special character.

1. Both single line and multi line comments are handled.
2. The last rule ensures that no illegal token is formed and sent to the next phase of compiler. It prints out if any illegal token appears in input file.

**Implementation of Symbol Table**

The concept of Chain Hashing is used for the symbol table.

A structure used for storing all information about a token is as follows:

struct DataItem   
{  
 char\* text; // stores the token  
 char\* type; // type of token  
 int lineno; // line no. on which token is found  
 struct DataItem \* next; // pointer to next data item  
 int attr; // Attribute value  
};

The variable **int att** is unique for a particular token. All multiple occurrences of single token are stored in symbol table with same attribute value.

Following subroutines are defined as part of implementation of symbol Table

1. void insert\_hash(char\* text, char\* type, int lineno)
   1. Arguments are token, its type and current line number.
   2. It inserts the token in symbol table.

2. int hashCode(char \* key)

1. Argument is the token and a unique number is returned which is used as an index to hash the token.

3. void display\_hash()

1. A simple subroutine that prints out the symbol table

|  |  |
| --- | --- |
|  |  |
|  | { |

**How to get the output**

Clone the repository. Next type the commands:

* lex scanner.l
* cc lex.yy.c
* ./a.out < tc1.c

You can input test cases in this fashion. There are four test cases each in the files tc1.c, tc2.c, tc3.c, and tc4.c. The output consists of the stream of tokens in the order in which they are generated. The symbol table is displayed too.