Om Shet
TE C32
2103163

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment No : 1

Aim : Write a program to implement Lexical Analyzer

Theory :

- **Lexical Analysis**

⇒ Lexical analysis is the first phase of a compiler. The main task accomplished by lexical analysis is to identify the set of a valid word of the language that occurs in an input stream.

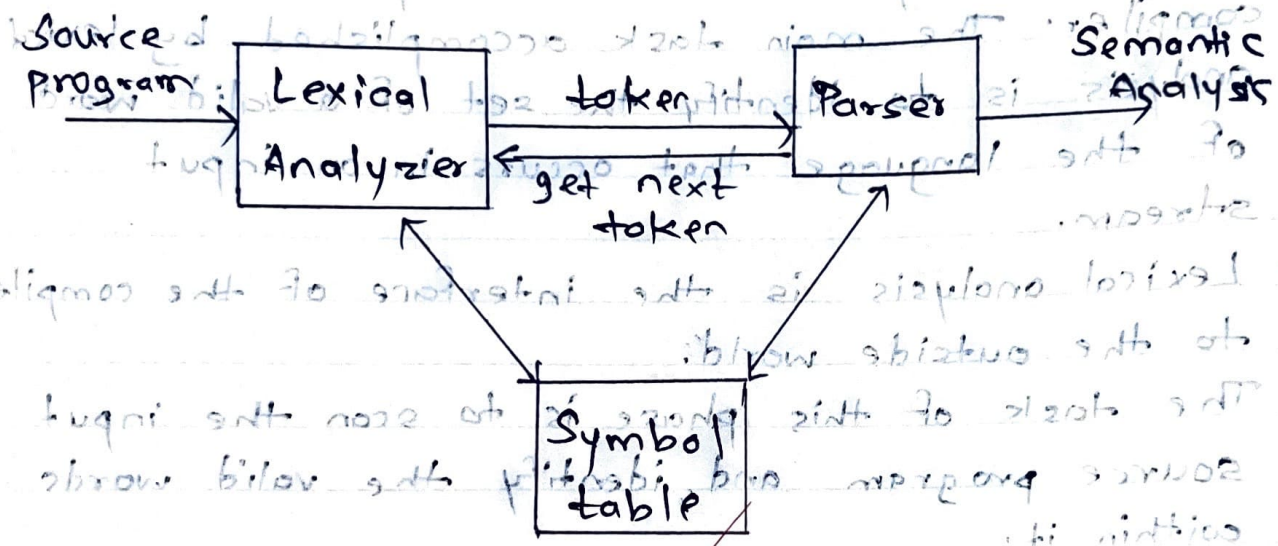
- Lexical analysis is the interface of the compiler to the outside world.
- The task of this phase is to scan the input source program and identify the valid words within it.
- It also does the additional job of removing cosmetics like

i) Extra white spaces, comments added by the user etc from the program.

ii) Expanding the user defined macros like #include and # define in C, reporting of foreign words that are the words, not belonging to the language of the source program if any.

- In computer science, lexical analysis is the first stage of processing the language.
- It is the process of converting a sequence of characters into a sequence of tokens.
- Stream of characters making up the source program or other input is read one at a

Role of Lexical Analyzer :



time and grouped into lexemes.

- A program function which performs lexical analysis is called a lexical analyzer, lexer or scanner.
- A lexical/lexer often exists as a single entity which is called by the parser.

• **Lexeme**

⇒ A lexeme is a basic unit of a language consisting of one word or several words. That corresponds to a set of words that are different forms of the same word

- For e.g., run, runs, running, ran are example or terms of same lexeme.

• **Tokens**

⇒ Lexical token/tokens are sequence of a characters that can be treated as a unit in the grammar of programming languages.

- For e.g.,

1) Type token ("id" {a-z, A-Z, 0-9}, "num" {0-9}...)

2) Punctuation tokens ("if", "void", "return", "begin", "call", "const", "do", "end", ...)

3) Non-tokens

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

- **Pattern**

⇒ A pattern is a set of a string in Input for which the same token is produced as output. These set of strings is described by a rule called "pattern".

- For e.g,

Token	Lexeme	Pattern
Literal	"hello"	chars with in " "
Number	0.31, 0.5	List of numbers.
if	IF	if
const	const	const

- **Conclusion :**

In this experiment, we learnt about the basics of lexical analysis.

22/02/2024

Code:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  string x;
  cout << "Enter the reqiured string" << endl;
  getline(cin, x, static_cast<char>(EOF));
  cout << "The Given string is :" << endl;
  cout << x << endl;

  set<string> sep = {"\n", ";", "\t"};
  set<string> key = {"int", "float", "cin", "cout", "for", "if",
"else"};
  set<string> op = {"=", "+", "-", "*", "%", "/", "<=", ">=", "=="};
  vector<int> token;

  stringstream ss(x);
  string word;
  vector<string> all;
  while (getline(ss, word, ' ')) {
    all.push_back(word);
  }

  for (int i = 0; i < all.size(); i++) {
    if (sep.find(all[i]) != sep.end()) {
      cout << all[i] << " "
           << "seperator" << endl;
    } else if (op.find(all[i]) != op.end()) {
      cout << all[i] << " "
           << "operator" << endl;
    } else if (key.find(all[i]) != key.end()) {
      cout << all[i] << " "
           << "Keyword" << endl;
    } else if (all[i][0] >= '0' and all[i][0] <= '9') {
      cout << all[i] << " "
           << "constant" << endl;
    } else {
      cout << all[i] << " "
           << "identifier" << endl;
    }
  }
```

```
  return 0;
}
```

Output:

```
Console   Shell
Run                                  25s on 09:59:44, 03/05 ✓
Enter the reqiured string
int main() {

cout<<"Hello";
return 0;

}
The Given string is :
int main() {

cout<<"Hello";
return 0;

}

int Keyword
main() identifier
{

cout<<"Hello";
return identifier
0;

}
 constant
```