**NATIONAL INSTITUTE OF TECHNOLOGY, DELHI**



COMPILER DESIGN PROJECT

MINI-C COMPILER

(CSC)

**WORKED ON, BY :**  VADLAMUDDI NEELVITTAL BHARATH (191210053)

VINAY CHOUDHARY (191210059)

VINAY JAISWAL (191210060)

**UNDER GUIDANCE OF :** DR. SHELLY SACHDEVA

NEHA BANSAL

KANIKA SONI

PRABHAT PUSHP

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **TOPIC** | **PAGE NO.** |
| 1. | OBJECTIVE | 3 |
| 2. | MINI-COMPILER SPECIFICATIONS | 4-7 |
| 3. | DOMAIN OF THE COMPILER | 8 |
| 4. | SAMPLE INPUT & OUTPUT | 9-15 |
| 5. | REFERENCES | 16 |
| 6. | NOTE OF THANKS | 17 |
|  |  |  |

**ABSTRACT**

**OBJECTIVE**

The main goal of this project is to design a mini compiler for a subset of the C language as part of the Compiler Design Lab(CO351) course. The compiler is to be built in four phases finishing at the Intermediate Code Generation Phase. The subset of the C language chosen is to include certain data types, constructs and functions as mentioned in the specifications below. The implementation will be carried out with LEX and YACC.

**PHASES OF THE PROJECT**

* ● Implementation of Scanner/Lexical Analyser
* ● Implementation of Parser
* ● Implementation of Semantic Checker for C language
* ● Intermediate Code generation for C language

**MINI-COMPILER SPECIFICATIONS :**

The compiler is going to support the following cases :

1. **Keywords** ​- int, break, continue, else, for, void, goto, char, do, if, return, while.
2. **Identifiers** ​-​ ​identified by the regular expression ( \_ |{letter})({letter}|{digit}| \_ ){0,31}.
3. **Comments**​ -​ ​single line comments (specified with // or /\* ... \*/), multi-line comments ( specified with /\* ... \*/).
4. **Strings** ​- can identify strings mentioned in double quotes.
5. **Preprocessor directives**​ - can identify filenames (stdio.h) after #include.
6. **Data types**​ - int,char (supports comma declaration).

7. **Arrays** ​-​ ​int A[n]  
 ○ Syntax

■ int A[3]={1,2,3};

​​

8. **Punctuators** ​-​ ​[ ] , <> , {} , , , : , = , ; , # , ” ” ,’ ’.

​9. **Operators** - arithmetic ( +, - , \* , / ) ,increment( ++ ) and decrement( -- ), assignment (

..........​ = ).

​​10.**Condition**​ - if else

○ Syntax:

■ if (condition == true){

//code

}

else{

//code

}

​11. **Loops -**

○ Syntax

* while(condition){

//code  
}

* for(initialization;condition;increment/decrement){  
  //code  
  }
* while(condition){  
  //code  
  }
* do{  
  //code

}while(condition);

The loop control structures that are supported are break,continue and goto.

12. **Functions-**

Void functions with no return type and a single parameter will be implemented

○ Syntax

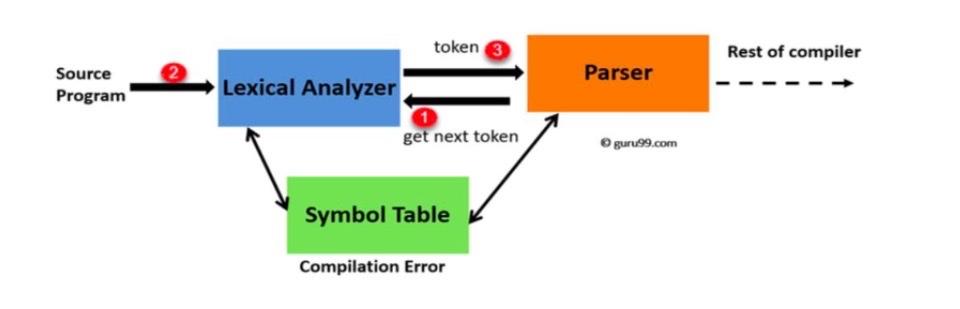
■ void sample\_function(int a){

//code

}

The mini compiler will be implemented in a straightforward fashion , using Lex and YACC tools starting off with the Scanner as the first module. If time permits we will add more features to cover a larger subset of the C language.

**ARCHITECTURE AND WORKING :**



**TOOLS TO BE USED :**

YACC, LEX

**DOMAIN OF THE COMPILER :**

The mini-compiler handles most of the cases of the C Language :

1. Identifies and removes comments.

2. Identifies the various operators in the language.

3. Checks for the validity of the identifiers.

4. Identifies the types of the variables, strings and numeric constants.

5. Ignore white spaces.

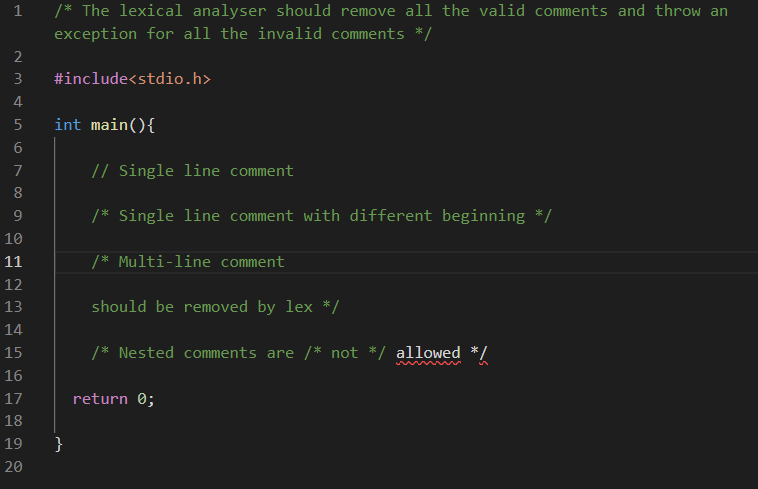
6. Able to identify the keywords, function definitions and loops.

Syntax is handled by yacc where grammar rules are specified for the entire language.

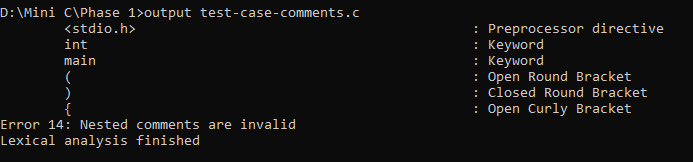
Semantics are handled using semantic rules for type checking while performing operations to ensure operations are valid.

**SAMPLE INPUT AND OUTPUT :**

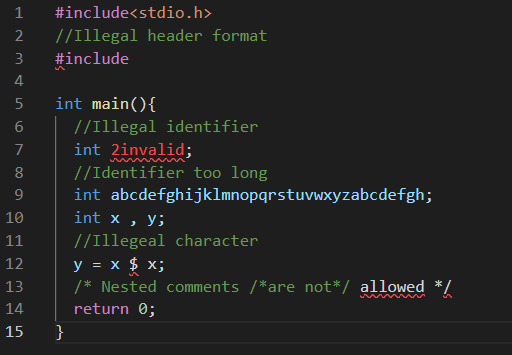
* **Input 1**

****

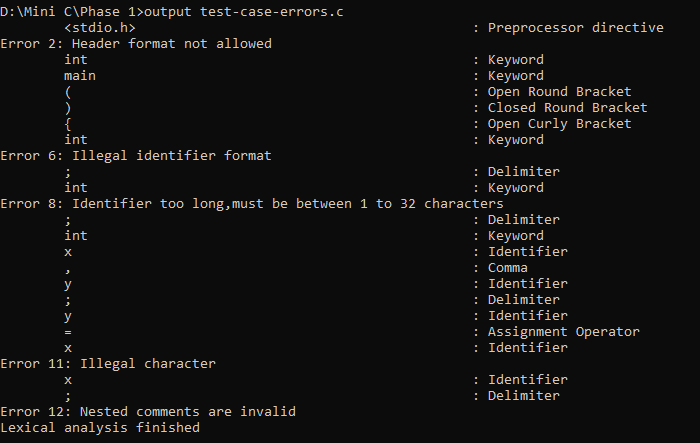
* **Output 1**

****

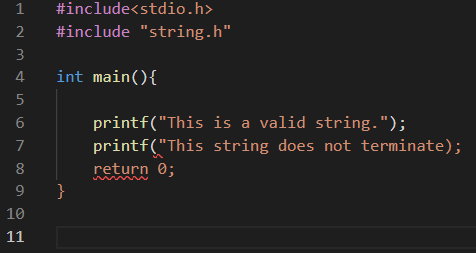
* **Input 2**

****

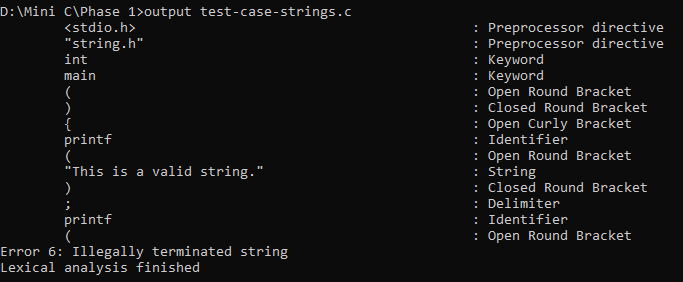
* **Output 2**

****

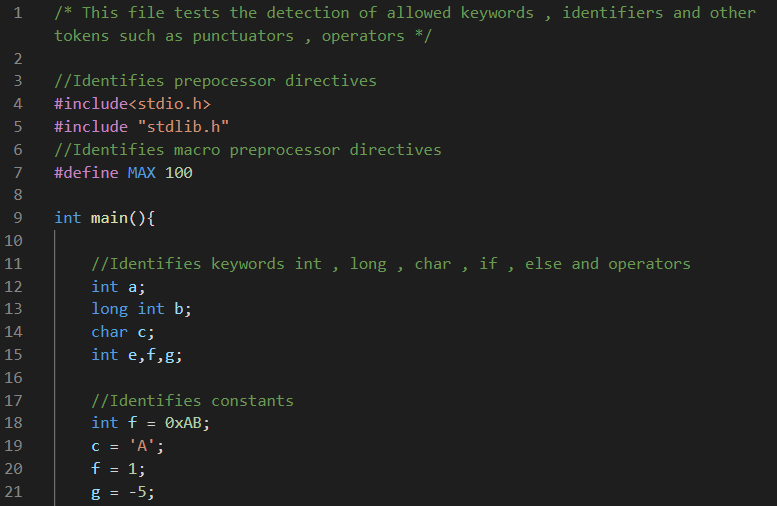
* **Input 3**

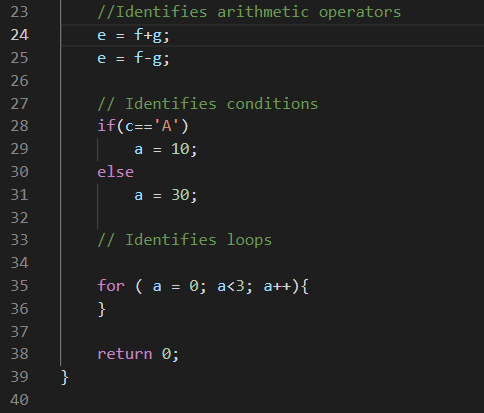
****

* **Output 3**

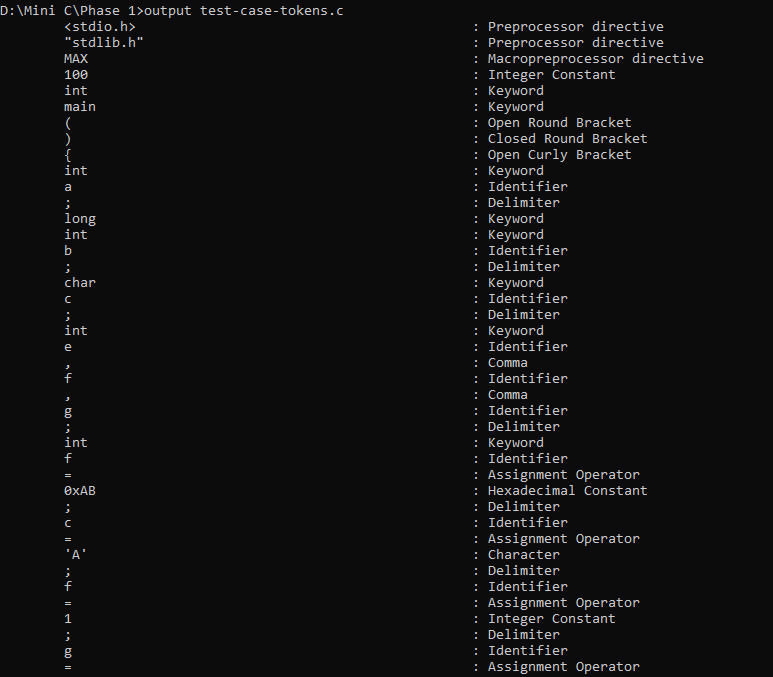
****

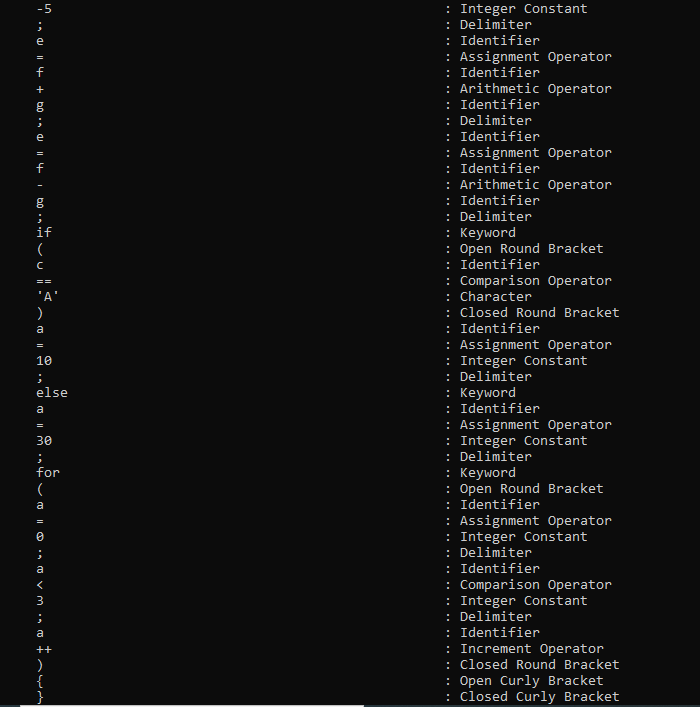
* **Input 4**

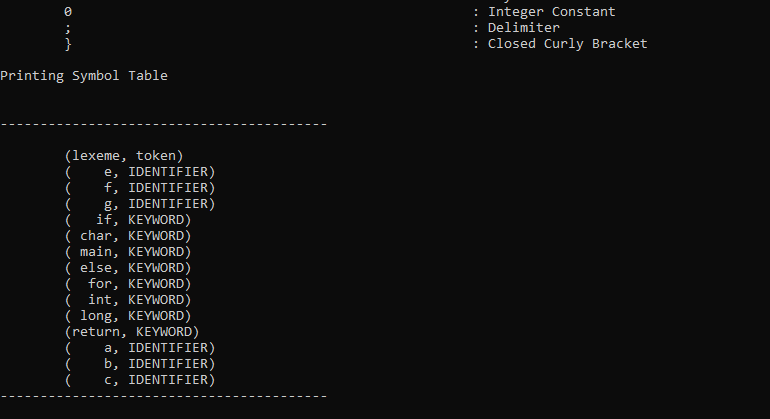
****

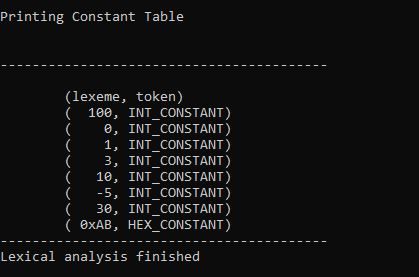
****

* **Output 4**

****

****

****

****

**REFERENCES :**

1. Aho A.V, Sethi R, and Ullman J.D. Compilers: Principles, Techniques, and Tools. Addison-Wesley, 1986

2. http://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid065185.pdf

3. http://cse.iitkgp.ac.in/~bivasm/notes/LexAndYaccTutorial.pdf

4. GeeksforGeeks.

**NOTE OF THANKS**

We are extremely grateful to our faculty-in-charge Dr.Shelly Ma’am for providing us such a great opportunity to work on this project and learn the basics of Compiler Design.

We extend our gratitude to TA’s in-charge, Neha Basal Ma’am, Kanika Soni Ma’am and Prabhat Pushp Sir for their constant support and guidance throughout the making of the project and making it a success.