EXPERIMENT – 1

Lexical Analyzer using C language

AIM:

To design and implement a lexical analyzer using c language

COURSE OUTCOME :

CO1 : Implement lexical analyzer and syntax analyzer using the tool LEX and YACC

ALGORITHM:

1. Start
2. Initialize the pointer fp in write mode and write into the file.
3. Open the written file in read mode.
4. Read the characters from a file and enter each string into a two dimensional array till the end of file.
5. Compare each string in the array with available key words. If match occurs print the string as a keyword. Else go to step v
6. Compare the string with available operators. If match occurs print string as operators. Else go to step vi.
7. Check whether the string is digit, if match occurs, identify the same as constant Else go to step vii.
8. Compare the string with available header files. If match occurs print the string as header file. Else go to step viii.
9. Print the string as identifier
10. Stop

PROGRAM

RESULT:

The program to implement the lexical analysis have been executed successfully and the course outcome has achieved.

EXPERMENT NO - 2

Lexical Analyzer using Lex Tool

AIM:

To design and implement a lexical analyzer using Lex Tool

COURSE OUTCOME :

CO1 : Implement lexical analyzer and syntax analyzer using the tool LEX and YACC

ALGORITHM

1. Start
2. Lex program contains three sections: definitions, rules, and user subroutines. Each section must be separated from the others by a line containing only the delimiter, %%.
3. Lex programs follow this basic structure: definitions %% rules %% user_subroutines
4. Include necessary headers in definition part
5. Define **patterns** using regular expressions and associate them with **actions in Rule part.**
    a. For the input matching [0-9]+ print it is a number
    b. For the input matching "if"|"else"|"while"|"int" print it is a keyword
    c. For the input matching [_a-zA-Z][a-zA-Z0-9]* print it is an identifier
    d. For the input matching ">"|"<"|"="|">="|"<="|"+"|"-"|"*"|"/" print it is an operator.
    e. For the input matching "'"|":"|";"|","|"("|")" print it is a special character.
6. The last section, include the `main()` function which calls the yylex() function.
    a. **`yylex()` scans the input**:
    b. Matches input against the regular expressions defined.
    c. Executes corresponding action when a match is found.
    d. Stores the matched string in `yytext`
7. Stop

RESULT:

Program for implementation of Lexical Analyzer using Lex tool has been executed successfully and Course outcome has achieved.

EXPERMENT NO-3. a

Count the number of Vowels and Consonants using LEX Tool

AIM:

To write a lex program to find out total number of vowels and consonants from the given input string using LEX tool

COURSE OUTCOME :

CO1 : Implement lexical analyzer and syntax analyzer using the tool LEX and YACC

ALGORITHM

1. Start
2. Lex program contains three sections: definitions, rules, and user subroutines. Each section must be separated from the others by a line containing only the delimiter, %%.
3. Lex programs follow this basic structure: definitions %% rules %% user_subroutines
4. Include necessary headers in definition part. Declare two variables for vowel count and consonant count
5. Define **patterns** using regular expressions and associate them with **actions in Rule part.**
    a. For the input matching [aeiouAEIOU] increment vowel count by 1
    b. For all others increment the consonant count by 1
6. The last section, include the `main()` function which calls the yylex() function. Main function takes the input from terminal and print the respective counts.
    a. **`yylex()` scans the input**:
    b. Matches input against the regular expressions defined.
    c. Executes corresponding action when a match is found.
    d. Stores the matched string in `yytext`
7. Stop

RESULT:

Program to find out total number of vowels and consonants has been executed successfully and Course outcome has achieved.

EXPERMENT NO-3. b

Check if a number is ODD or EVEN using LEX Tool

AIM:

To write a lex program to check if a given number is odd or even using LEX tool

COURSE OUTCOME :

CO1 : Implement lexical analyzer and syntax analyzer using the tool LEX and YACC

ALGORITHM

1. Start
2. Lex program contains three sections: definitions, rules, and user subroutines. Each section must be separated from the others by a line containing only the delimiter, %%.
3. Lex programs follow this basic structure: definitions %% rules %% user_subroutines
4. Include necessary headers in definition part.
5. Define **patterns** using regular expressions and associate them with **actions in Rule part.**
    a. For the regular expression [0-9]*[02468] print it is an even number
    b. For the regular expression [0-9]*[1357] print it is an odd number
6. The last section, include the `main()` function which calls the yylex() function.
    a. **`yylex()` scans the input**:
    b. Matches input against the regular expressions defined.
    c. Executes corresponding action when a match is found.
    d. Stores the matched string in `yytext`
7. Stop

RESULT:

Program to find out total number of vowels and consonants has been executed successfully and Course outcome has achieved.