**Lexical Analyzer and Parser for My Programming Language**

The programming language consists of different rules, statements, tokens, identifiers etc. In this project, the syntax of the C++ is modified to develop a new programming language. The software is developed to parse the program and identify each operator, identifier, token and keyword.

# Language Basics

Following are some basics of the new programming language:

## Keywords

For if statement: condition

For else statement: otherwise

Loops: loop

Integer: integer

Character: charac

Etc.

## Punctuators and Identifiers

The punctuators and identifiers are developed in the initial phase of the compiler. Here are some of the examples implemented in this project.

(ch == ' ' || ch == '+' || ch == '-' || ch == '\*' ||

ch == '/' || ch == ',' || ch == ';' || ch == '>' ||

ch == '<' || ch == '=' || ch == '(' || ch == ')' ||

ch == '[' || ch == ']' || ch == '{' || ch == '}' ||

ch == '&' || ch == '|')

(str[0] == '0' || str[0] == '1' || str[0] == '2' ||

str[0] == '3' || str[0] == '4' || str[0] == '5' ||

str[0] == '6' || str[0] == '7' || str[0] == '8' ||

str[0] == '9' || isPunctuator(str[0]) == true)

## Grammar

A few examples of the grammar are:

<condition> -> condition<cond><expression\_list>

<cond> -> (<fact> = (.|&|\*))

<loop> -> loop<cond><expression\_list>

<cond> -> (<fact> = (integer|&++)

<expression\_list> -> { <product> | <addition> | <division>}

<addition> -> (<fact>{,<fact>})

<product> -> <prod\_term> {+<prod\_term>}

<prod\_term> -> <fact>{, <fact>}

<division> -> <div\_term> {+<div\_term>}

<div\_term> -> <fact>{, <fact>}

<fact> -> IDENT | integer

# Code Implementation

The input program is:

include <iostream>

using namespace std;

voiid main ()

{

integer a=5, b=3;

floatt c=8;

condition(a<b)

{

condition (a<5)

{

give (a+b)/c;

}

otherwise

{

give 0;

}

}

otherwise

{

give (b\*c)-a;

}

switchh(a)

{casse 'A':

input>>input;

output<<abcd;

default: break;

}

integer weeks = 3, days\_in\_week = 7;

strart\_loop (integer i = 1; i <= weeks; ++i) {

output << "Week: " << i << endl;

start\_loop (integer j = 1; j <= days\_in\_week; ++j) {

// conntinue if the day is an odd number

condition (j % 2 != 0) {

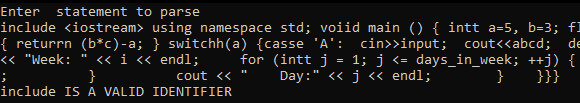
conntinue;

}

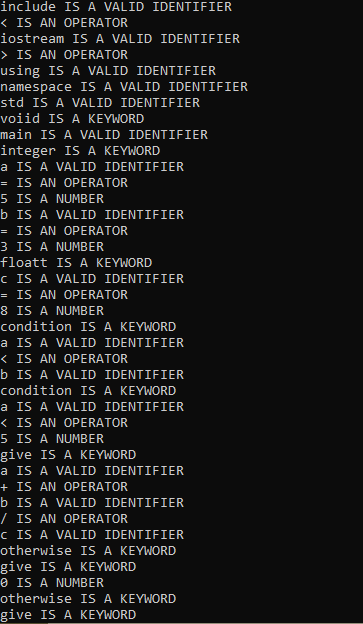
output << "Day:" << j << endl;

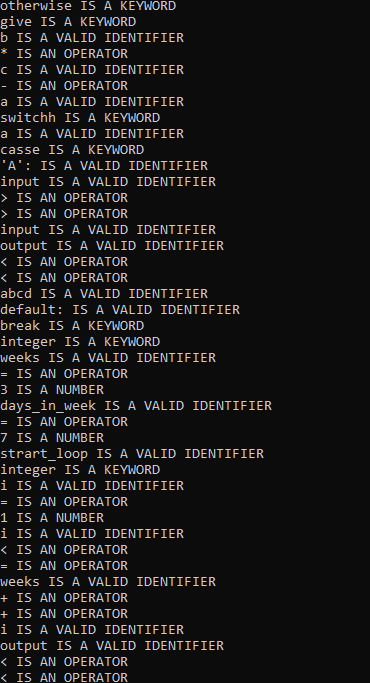
}}}}

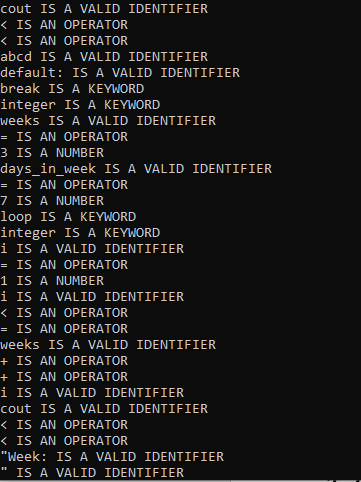
The keywords and identifiers are modified. Moreover, the program has all the complex functions such as nested if-else, switch statements, return, multiplications, divisions, break, etc. The developed program takes the input as:



The parse tree in the output can be seen as:







# Conclusion

The new language is the modified version of the C++. Similarly, the new identifiers, tokens, validators and operators can be defined. The initial phase of the compiler is developed using lexical analysis. This project has successfully developed the lexical analyzer and introduced few grammars, punctuators operators etc. for a new language.