Project Report

# Compiler for Scientific Calculator Program

Developed by

**Neel Soni - IT150**

**Apar Solanki – IT148**

**Sonaraj Kashyap – IT151**

**Makawana Vivek - IT167**

GuidedBy

# Prof. NV Mam

Dept.ofInformationTechnology



**Department of Information TechnologyFacultyof Technology**

**Dharmsinh Desai University College Road, Nadiad-3870012021-2022**

# CERTIFICATE

This Is to certify that the project entitled “**Scientific** **Calculator Program”** is abonafide report of the work carried out by

1. Mr.Soni Neel, Student ID No : 20ITUOS018
2. Mr.Solanki Apar, Student ID No : 20ITUOS060
3. Mr.Sonaraj Kashyap, Student ID No : 21ITUBD006
4. Mr.Makawana Vivek Student ID No : 21ITUSD012

Of Department of Information Technology, semester 6th,under the guidance and supervision for the award of the degree of Bachelor of Technology at Dharmsinh Desai University,Nadiad(Gujarat).Theywere involved in Project in subject of “**Language Translator**” during the academic year 2021-2022.

Prof. N. V.Mam Desai(Lab Incharge)

Department of Information Technology,Facultyof Technology,

Dharmsinh Desai University, NadiadDate:

Prof.(Dr.)V.K.Dabhi,

Head, Department of Information Technology,Facultyof Technology,

Dharmsinh Desai University, NadiadDate:

**ProjectDetails:**

# Introduction

**GrammarName:**SentiGrammar

## ValidSentencesinLanguage:

1. 4 + 5 = 9
2. 2 - 19 = -17
3. 7 / 2 = 3.5
4. 7 \* 9 = 63

**Keywords:**CE

## Operators:

+ , - , / , \* , =

## Digit:

[0-9]+ int

[0-9]+(.[0-9]+) float

**Punctuation Marks:**

|  |  |  |
| --- | --- | --- |
| **Punctuation Mark’s Name:** | **Punctuation Marks** | **Punctuation Mark’s character** |
| New Line | [\n] | n |
| Eos | . | e |
| Separator | , | s |
| White Space | [/t] | w |

# First&Follow

## Grammar:

S -> X = E

E -> TE'

E'->+TE'| -TE'| ε

T -> FT'

T'->\*FT'| /FT'| ε

F ->(E)| num

num ->0|1|2|3|4|5|6|7|8|9

## Non-terminals:

 E | E’ | T | T’ | F | num

## Terminals:

0|1|2|3|4|5|6|7|8|9| ε | + | - | / | \* | X | =

## Output:

**First:**

FIRST[S] = { int }FIRST[E]={int}

FIRST[KW] = {means, how, many, and, convert, into}FIRST[D]={ int }

FIRST[OP] = {Joule, calary, watt, Hp, joule/second, kilowatt}FIRST[PUN] ={?. ,}

FIRST[Q]={?}

FIRST[G]={.}

FIRST[T] ={,}

## Follow:

FOLLOW[S] ={}

FOLLOW[E] ={means,how,many,and,convert,into }

FOLLOW[KW] = { means, how, many, and, convert, into, Joule, calary, watt, Hp,joule/second,kilowatt}

FOLLOW[D]=Joule

FOLLOW[OP] = { Joule, calary, watt, Hp, joule/second, kilowatt }FOLLOW[PUN] ={$}

FOLLOW[Q]={$}

FOLLOW[G]={$}

FOLLOW[T]={$}

# DFA&Algorithm

## DFA:

**Algorithm:**

while(true){

    switch(state){

        case0:

            c=getChar()

            if(c=="i") state=1

            elseif(c==isLetter(c)) state=4

            elseif(c==" ") state=6

            elseif(isDigit(c)) state=11

            elseif(c=="+"||c=="-"||c=="%"||c=="\*") state=16

            elseif(c=="'") state=21

            elseif(c==">="||c=="<="||c=="<"||c==">"||c==":="||c=="="||c=="<>") state=25

            elseif(c=="("||c==")"||c=="{"||c=="=}"||c=="["||c=="]"||c==",") state=27

            elseif(c==".") state=14

            break;

        case1:

            c=getChar()

            if(c=="f") state=2

            elseif(c==isLetter(c)|| c==isDigit(c)) state=4

            break;

        case2:

            state=3 *//acceptance state*

            break;

        case4:

            c=getChar();

            if(c==isLetter(c)|| c==isDigit(c)) state=4

            else state=5 *//acceptance state*

            break;

        case6:

            c=getChar();

            if(c==" "){

                state=7

                c=getChar();

                if(c==" "){

                    state=8

                    c=getChar();

                    if(c==" ") state=9

                }

            }

            else state=10 *//acceptance state*

            break;

        case9:

            c=getChar()

            state=10 *//acceptance state*

            break;

        case11:

            c=getChar()

            if(isDigit(c)) state=11

            elseif(c==".") state=12

            else state=13

            break;

        case12:

            c=getChar()

            if(isDigit(c)) state=12

            else state=13

            break;

        case14:

            c=getChar();

            if(isDigit(c)) state=12

            else state=15 *//failure state*

        case15:

            fail()

        case16:

            c=getChar()

            if(c=="\*") state=18

            else state=17 *//acceptance state*

            break;

        case18:

            c=getChar()

            if(c=="\*") state=19

            elsefail()

            break;

        case19:

            state=20 *//acceptance state*

            break;

        case21:

            c=getChar()

            if(c=="'") state=23 *//acceptance state*

            else state=22

            break;

        case22:

            c=getChar()

            if(c=="'") state=23

            else state=22

            break;

        case23:

            state=24 *//acceptance state*

            break;

        case25:

            state=26 *//acceptance state*

            break;

        case27:

            state=28 *//acceptance state*

            break;

        case3:case5:case10:case13:case17:case20:case24:case26:case28:

            state=0

            break;

        default:

            fail()

    }

}

boolisDigit(token t){

    if(t>=0&& t<=9)returntrue

    returnfalse

}

boolisLetter(token t){

    if(t>="A"&&t<="Z"|| t>="a"&&t<="z")returntrue

    returnfalse

}

fail(){print("Error:)");return;}

# ScannerPhaseinC++Language

#include<iostream>

#include<string>

usingnamespacestd;

boolisDigit(charc)

{

    if(c>='0'&&c<='9')

        returntrue;

    returnfalse;

}

boolisLetter(charc)

{

    if((c>='A'&&c<='Z')||(c>='a'&&c<='z'))

        returntrue;

    returnfalse;

}

voidfail()

{

    cout<<"Error :)"<<endl;

    return;

}

chargetChar()

{

    charch;

    cin>>ch;

    returnch;

}

intmain()

{

    intstate=0;

    charc;

    while(true)

    {

        switch(state)

        {

        case0:

            c=getChar();

            if(c=='i')

                state=1;

            elseif(isLetter(c))

                state=4;

            elseif(c==' ')

                state=6;

            elseif(isDigit(c))

                state=11;

            elseif(c=='+'||c=='-'||c=='%'||c=='\*')

                state=16;

            elseif(c=='\'')

                state=21;

            elseif(c=='>='||c=='<='||c=='<'||c=='>'||c=='='||c=='='||c=='<>')

                state=25;

            elseif(c=='('||c==')'||c=='{'||c=='}'||c=='['||c==']'||c==',')

                state=27;

            elseif(c=='.')

                state=14;

            break;

        case1:

            c=getChar();

            if(c=='f')

                state=2;

            elseif(isLetter(c)||isDigit(c))

                state=4;

            break;

        case2:

            state=3; *// acceptance state*

            break;

        case4:

            c=getChar();

            if(isLetter(c)||isDigit(c))

                state=4;

            else

                state=5; *// acceptance state*

            break;

        case6:

            c=getChar();

            if(c==' ')

            {

                state=7;

                c=getChar();

                if(c==' ')

                {

                    state=8;

                    c=getChar();

                    if(c==' ')

                        state=9;

                }

            }

            else

                state=10; *// acceptance state*

            break;

        case9:

            c=getChar();

            state=10; *// acceptance state*

            break;

        case11:

            c=getChar();

            if(isDigit(c))

                state=11;

            elseif(c=='.')

                state=12;

            else

                state=13;

            break;

        case12:

            c=getChar();

            if(isDigit(c))

                state=12;

            else

                state=13;

            break;

        case14:

            c=getChar();

            if(isDigit(c))

                state=12;

            else

                state=15; *// failure state*

        case15:

            fail();

            break;

        case16:

            c=getChar();

            if(c=='\*')

                state=18;

            else

                state=17; *// acceptance state*

            break;

        case18:

            c=getChar();

            if(c=='\*')

                state=19;

            else

                fail();

            break;

        case19:

            state=20; *// acceptance state*

            break;

        case21:

            c=getChar();

            if(c=='\'')

                state=23; *// acceptance state*

            else

                state=22;

            break;

        case22:

            c=getChar();

            if(c=='\'')

                state=23;

            else

                state=22;

            break;

        case23:

            state=24; *// acceptance state*

            break;

        case25:

            state=26; *// acceptance state*

            break;

        case27:

            state=28; *// acceptance state*

            break;

        case3:

        case5:

        case10:

        case13:

        case17:

        case20:

        case24:

        case26:

        case28:

            state=0;

            break;

        default:

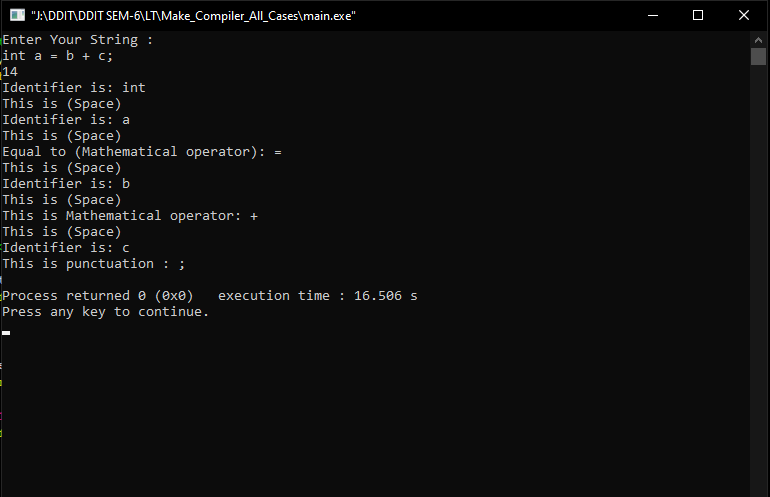
            fail();

        }

    }

}

**Output:**

****

# ScannerPhasein Lex

%%

if|else|begin|end|while|do|switch|until|case|exit{printf("%s this is keyword\n",yytext);}

[a-zA-Z][a-zA-Z0-9]\*{printf("%s this is identifier\n",yytext);}

[0-9]"."[0-9]+{printf("%s this is float-number\n",yytext);}

[0-9][0-9]+{printf("%s this is int-number\n",yytext);}

"["|"]"|"{"|"}"|"("|")"|","{printf("%s this is Pancuation-mark\n",yytext);}

">"|"<"|">="|"<="|"="|"!="{printf("%s Relational Operator.\n",yytext);}

"+"|"-"|"/"|"\*"{printf("%s Arithmetic Operator.\n",yytext);}

['][^']\*[']{printf("%s This is String.\n",yytext);}

"\*"{3}.\*"\*"{3}{printf("%s This is comment\n",yytext);}

" "{printf("%s This is white space.\n",yytext);}

"    "{printf("%s This is tab space.\n",yytext);}

.{printf("%s Unrecognized cha.\n",yytext);return0;}

%%

intyywrap(){}

intmain(){

yylex();

return0;

}

**Input:**

if (x < 10) {

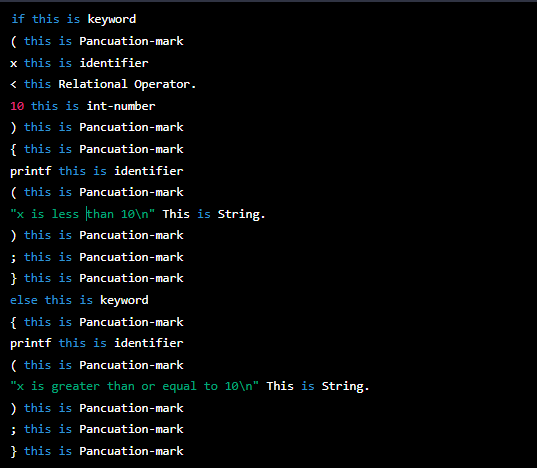
    printf("x is less than 10\n");

} else {

    printf("x is greater than or equal to 10\n");

}

**Output:**



# Yacc useful in our Language

## Code:

### Lexer(lex\_calculator.l)

%{

#include "calculator.tab.h"

%}

DIGIT [0-9]

LETTER [a-zA-Z]

WS [ \t\r\n]

%%

{DIGIT}+ { yylval = atoi(yytext); return INTEGER; }

{LETTER}+ { yylval.str = strdup(yytext); return NAME; }

{WS}+ ; /\* ignore whitespace \*/

"+" { return PLUS; }

"-" { return MINUS; }

"\*" { return TIMES; }

"/" { return DIVIDE; }

"(" { return LPAREN; }

")" { return RPAREN; }

"," { return COMMA; }

. { printf("Unknown token: %s\n", yytext); }

%%

int yywrap(void) {

return 1;

}

**Parser ( Calculator.y )**

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int yylex(void);

void yyerror(char const \*);

#define YYSTYPE union {

int integer;

char \*str;

}

typedef struct {

char \*name;

int n\_args;

double (\*func)(double \*);

} function;

function functions[] = {

{ "sin", 1, sin },

{ "cos", 1, cos },

{ "tan", 1, tan },

{ NULL, 0, NULL }

};

#define MAX\_ARGS 10

double call\_function(char \*name, int n\_args, double \*args) {

function \*f = functions;

while (f->name) {

if (strcmp(f->name, name) == 0 && f->n\_args == n\_args) {

return f->func(args);

}

f++;

}

yyerror("unknown function");

return 0;

}

double evaluate(char \*name, int n\_args, double \*args) {

if (n\_args > MAX\_ARGS) {

yyerror("too many arguments");

return 0;

}

if (name) {

return call\_function(name, n\_args, args);

} else {

return args[0];

}

}

%}

%token INTEGER NAME

%token PLUS MINUS TIMES DIVIDE LPAREN RPAREN COMMA

%left PLUS MINUS

%left TIMES DIVIDE

%nonassoc UMINUS

%%

input: /\* empty \*/

| input line

;

line:

'\n'

| exp '\n' { printf("= %g\n", $1); }

;

exp:

INTEGER { $$ = $1; }

| NAME LPAREN arglist RPAREN { $$ = evaluate($1, $3, $4); }

| exp PLUS exp { $$ = $1 + $3; }

| exp MINUS exp { $$ = $1 - $3; }

| exp TIMES exp { $$ = $1 \* $3; }

| exp DIVIDE exp { $$ = $1 / $3; }

| MINUS exp %prec UMINUS { $$ = -$2; }

| LPAREN exp RPAREN { $$ = $2; }

;

arglist:

/\* empty \*/

| exp { $$ = 1; $1list[0] = $1; }

| arglist COMMA exp { if ($1 >= MAX\_ARGS) { yyerror("too many arguments"); } else { $3list[$1++] = $3; }

$$

**Main Program : (main\_calculator.cpp)**

#include <stdio.h>

#include "calculator.tab.h"

int main() {

    yyparse();

    return 0;

}

void yyerror(char const \*s) {

    fprintf(stderr, "Error: %s\n", s);

}

**Header (Calculator.h)**

#ifndef CALCULATOR\_TAB\_H

#define CALCULATOR\_TAB\_H

#include <math.h>

extern "C" {

    int yyparse(void);

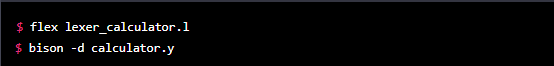
    void yyerror(char const \*);

}

#endif

**Compile:**

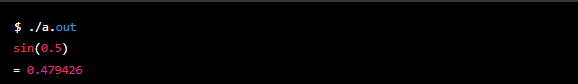
**1) Generate Lexer and Parser**

****

**2)Compile Main Program and link it with the lexer and parser**

**Screenshot (3553).png**

**3)Run Calculator**

****

# Conclusion

In conclusion, the project report on the creation of a calculator in D-lang using flex-tools, scanner code in C++, and Yacc tool code represents a comprehensive demonstration of the use of these powerful tools in building complex software applications.

The project showcases the ability of Lexical Analyzer and Parser Generator tools to recognize patterns and handle complex grammars in the input stream, and the use of compiler tools to generate optimized code for executing the program.

The report demonstrates the effective use of C++ and D-lang in building an interactive calculator with a user-friendly interface, along with the ability to handle basic arithmetic operations.

Overall, the project report represents a valuable contribution to the field of computer programming and software development, showcasing the power and flexibility of these tools in building complex software applications.

It can be a useful resource for students, researchers, and professionals interested in building software applications that require parsing and processing of input data.

[**Source Code - GitHub Link**](https://github.com/neel13062003/Compiler_Design_Project) **=** [**https://github.com/neel13062003/Compiler\_Design\_Project**](https://github.com/neel13062003/Compiler_Design_Project)