KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Technology

Title: Designing a simple compiler using Flex and Bison

Course Title: Compiler Design Laboratory

Submitted by:

Rakib Mahmud

Roll: 1807077

Year: 3rd

Semester: 2nd

Department of Computer Science and Engineering

Khulna University of Engineering and Technology, Khulna

Submission date: 20/12/2022

Objectives:

1.To know about the compiler
2.To know the translation of a high level language into a low level language
3. To know the top down parser and the bottom up parser .
4. To know the Flex and Bison for implementation of a compiler using C programming language .
5.To create a new language and it's semantic and syntactic rules .
6.To check some different type of input and their output of the compiler .
7.To implement the Regular Expression ,Context Free Grammar in the compiler .

Introduction:

A compiler is a computer program that translates computer code written in one programming language into another language. The name compiler is primarily used for programs that translate source code from a high-level programming language to a lower level language to create an executable program.

Flex and Bison:

Lex is a program that generates lexical analyzer. It is used with YACC parser generator. The lexical analyzer is a program that transforms an input stream into a sequence of   tokens. .It reads the input stream and produces the source code as output through implementing the lexical analyzer in the C program

Bison is a general-purpose parser generator that converts a grammar description (Bison Grammar Files) for an LALR(1) context-free grammar into a C program to parse that grammar. The Bison parser is a bottom-up parser. ... Compile the code output by Bison, as well as any other source files .

Using Flex and Bison:

1. bison -d main.y
2. flex main.l
3. gcc lex.yy.c main.tab.c -o app
4. app

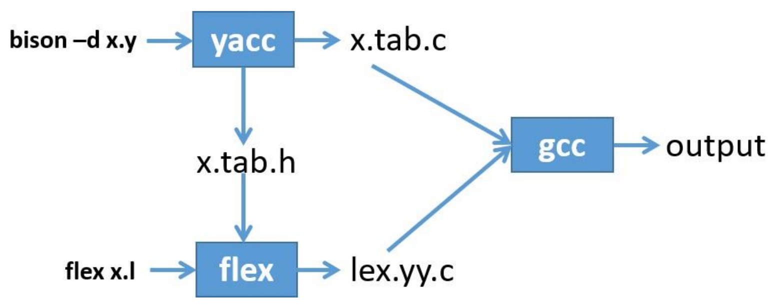

Fig-1: Workflow of Flex and Bison

My tokens:

INT, FLOAT, CHAR, NUM, VAR, PLUS, MINUS, MULT, DIV, MOD, INC, DEC, ASSIGN, GT, LT, EE, GTE, LTE, NTE, FBS, FBC, SBS, SBC, COMA, SM, COLON, IF, ELSE, FOR, SWITCH, CASE, DEFAULT, PRINT

Procedure

1.The code is divided into two part flex file (.l) and bison file (.y) .
2.Input expression check the lex (.y) file and if the expression satisfies the rule then it check the CFG into the bison file .
3.it's a bottom up parser and the parser construct the parse tree .firstly ,matches the leaves node with the rules and if the CFG matches then it gradually goes to the root .

CFG

Context-free grammars (CFGs) are used to describe context-free languages. A context-free grammar is a set of recursive rules used to generate patterns of strings. A context-free grammar can describe all regular languages and more, but they cannot describe *all* possible languages.

CFG Used:

```
program: MAIN FBS FBC SBS cstatement SBC {


                              printf("\nsuccessful compilation\n"); }
           ;

cstatement:
```

```
          | cstatement statement

          | cstatement cdeclaration

        | cstatement print_fun
          ;

cdeclaration:  TYPE ID1 SM        { printf("\nvalid declaration\n"); }
                     ;

TYPE : INT

    | FLOAT

    | CHAR
    ;

ID1  : ID1 COMA VAR      {
                         if($3.varvalue<26)
                         {

                                        if(check[$3.varvalue] == 1)
                                        {
                                           printf("error: multiple declaration for
%c\n",$3.varvalue+'a');
                                        }
                                        else
                                        {
                                           check[$3.varvalue] = 1;
                                        }
                         }
                         else
                         {
                               if(check[$3.varvalue] == 1)
                                      {
                                          printf("error: multiple declaration for  ");
                                   for(k=0;k<$3.length;k++)
                                              {
                                                printf("%c",$3.str[k]);
                                              }
                                              printf("\n");
                                        }
                                          else
                                          {
                                                check[$3.varvalue] = 1;
                                          }
                            }
                    }

        |VAR          {
                            if($1.varvalue<26)
                            {

                                                if(check[$1.varvalue] == 1)
                                                {
                                                        printf("error: multiple declaration for
%c !!!\n",$1.varvalue+'a');
                                                }
                                                else
                                                {
                                                        check[$1.varvalue] = 1;
                                                }
                            }
                            else
                                    {
                                     if(check[$1.varvalue] == 1)
                                          {
                                             printf("error: multiple declaration for  ");
                                      for(k=0;k<$1.length;k++)
                                                 {
                                                   printf("%c",$1.str[k]);
                                                 }
                                                 printf("\n");
                                          }
                                        else
                                        {
                                             check[$1.varvalue] = 1;
                                          }
                                    }
                      }
        ;
```

```
statement: expression SM

        |  print_fun

        | VAR ASSIGN expression SM                  {

                                                       symbols[$1.varvalue]=$3;
                                                        $$ = $3;
                                                      // printf("assign symbol table
%lf\n",symbols[$1.varvalue]);

           //printf("assign %lf\n",$3);
                                                        if($1.varvalue<26)
                                                                      {
                                                               if(check[$1.varvalue] != 1)
                                                               {
                                                                                    printf("error:undeclared variable is
%c \n",$1.varvalue+'a');
                                                                                    }

                                                        }
                                                        else
                                                        {
                                                               if(check[$1.varvalue] != 1)
                                                               {
                                                                                    printf("error: undeclared variable is
");

                                                       for(k=0;k<$1.length;k++)
                                                                       {
                                                                         printf("%c",$1.str[k]);
                                                                       }
                                                                       printf("\n");
                                                                            }
                                                         }

                                                                }
        |IF FBS expression FBC  SBS  statement SBC    {      if($3)
                                                            {
                                                                  printf(" if statement successfully execute !!!\n");

                                                            }
```

```
                                        }

        |IF FBS expression FBC  SBS  statement SBC ELSE SBS statement SBC {
                                                        if($3)
                                                       {
                                                                                  printf("if
statement successfully execute !!!\n");

                                                                          }

                                                       else
                                                       {

printf("else statement successfully execute !!!\n");
                                                                          }
                                                 }
        |FOR FBS forcase SM forcase SM forcase  FBC SBS statement SBC    {
                                                      int inc=$7-$3;
                                                      int count=0;
                                                      int dec=$3-$7;
                                                      if(flag==1){
                                                                for(i=$3;i<value;i+=inc)
                                                                       {
                                                                     count++;
                                                                        }
                                                               printf("ForLoop executes
%d times\n",count);

                                                            }
                                                       if(flag==2){
                                                                for(i=$3;i>value;i-=dec)
                                                                       {
                                                                     count++;
                                                                        }
                                                               printf("ForLoop executes
%d times\n",count);

                                                            }

                                                          else {
                                                                printf("ForLoop not
execute\n");
                                                                        }
                                                                flag=0;
                                                      }

      | SWITCH FBS expression FBC SBS switchcase Default SBC {   printf("Switch
statement successfully execute !!!\n");
                                                    temp1=0;}
        ;



expression: NUM                                 { $$ = $1;
                                                                //printf("NUM: %lf\n", $1);
                                                                }

        | VAR                           {
                        $$ = symbols[$1.varvalue];
                        s_temp=symbols[$1.varvalue];
                             }

        | expression PLUS expression    { $$ = $1 + $3;
                                                                        //printf("%d plus
expression\n",$$);
                                                                         }

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

        | expression MULT expression    { $$ = $1 * $3; }

        | expression DIV expression     {       if($3)
                                        {
                                                $$ = $1 / $3;
                                        }
                                        else
                                        {
                                                $$ = 0;
                                                printf("\nerror: division by zero\t");
                                        }
                                }

      | expression MOD expression    {      if($3>0)
```

```
                                        {
                                              $$=(int)$1%(int)$3;
                                             }
                                                  else
                                                    {       $$=0;
                                                            printf("error: modulus by zero \n");

                                                    }
                                                    }



| expression INC          {$$ = $1++; }

| expression DEC          {$$ = $1--; }

  | expression LT expression  {        if($1 < $3){

                                              $$=1;
                                              value=$3;
                                              flag=1;

                                             }
                                         else  {
                                                $$=0;
                                             }

                                }

  | expression GT expression {
                                        if($1 > $3){
                                              $$ =1;
                                              value=$3;
                                              flag=2;

                                             }
                                         else  {
                                                $$=0;
                                             }

                                 }

| expression EE expression      {        if($1==$3)
```

```
                                        {
                                            $$=1;
                                        }
                                        else
                                                    {
                                                        $$=0;
                                                    }
                                            }

        | expression NTE expression     {        if($1!=$3)
                                        {
                                            $$=1;
                                        }
                                        else
                                                    {
                                                        $$=0;
                                                    }
                                  }

        | expression GTE expression  { $$ = $1 >= $3;}

        | expression LTE expression  { $$ = $1 <= $3; }

          | FBS expression FBC                { $$ = $2;     }
          ;



forcase:VAR ASSIGN expression {     symbols[$1.varvalue]=$3;
                                $$=$3;



                      }

          |expression { $$=$1; }
        ;




switchcase:
           | switchcase Case
```

```
            ;

Case : CASE NUM COLON statement  {
                                        if(s_temp==(int)$2) {
                                                temp1=1;
                                                printf("Case No : %.0f execute and
Result  is : %f \n",$2,$4);
                                              }
                              }
                            ;

Default : DEFAULT COLON statement  {
                                        if(temp1==0) {
                                              printf("Default Result is :  %f \n",$3);
                                             }
                            }
          ;



print_fun: PRINT FBS variable FBC SM     {
                                    temp--;
                                        printf("printed Value is :");
                                    for(i=0;i<=temp;i++)
                                    {
                                             printf("%f  ",print_array[i]);
                                             }
                                    printf("\n");
                                    temp=0;i=0;

                           }
         ;

variable: variable COMA VAR     {

                           if($3.varvalue<26)
                           {

                                           if(check[$3.varvalue] == 1)
                                           {

print_array[temp++]=symbols[$3.varvalue];
```

```
                                                  }
                                                  else
                                                  {
                                                     printf("error:undeclared variable  is
%c\n",$3.varvalue+'a');
                                             }
                              }
                              else
                                      {
                                         if(check[$3.varvalue] == 1)
                                              {

print_array[temp++]=symbols[$3.varvalue];
                                              }
                                       else
                                             {
                                                       printf("error: undeclared variable is
");
                                             for(k=0;k<$3.length;k++)
                                                                    {

printf("%c",$3.str[k]);
                                                                    }
                                                                    printf("\n");
                                              }
                                      }

                            }

  |VAR                { if($1.varvalue<26)
                              {

                                                  if(check[$1.varvalue] == 1)
                                                  {

        print_array[temp++]=symbols[$1.varvalue];
                                                  }
                                                  else
                                                  {
                                                       printf("error:undeclared variable  is
%c\n",$1.varvalue+'a');
                                             }
```

```
                        }
                        else
                                  {
                                   if(check[$1.varvalue] == 1)
                                         {
                                           print_array[temp++]=symbols[$1.varvalue];
                                         }
                                       else
                                       {
                                           printf("error: undeclared variable is \n");
                                   for(k=0;k<$1.length;k++)
                                                 {
                                                  printf("%c",$1.str[k]);
                                                 }
                                                 printf("\n");
                                         }
                                  }

       }
;
```

Sample Input and Output:

Input	Output
Int main_function Fbs Fbc	valid declaration
	error: multiple declaration for rakib
Sbs	
	valid declaration
char rakib$$	error: multiple declaration for a
char rakib$$	
int a##b##a##d##c##e$$	valid declaration
float k$$	valid declaration
k<-62.22$$	printed Value is :9.000000
	if statement successfully execute !!!
rakib<-d$$	else statement successfully execute !!!
	printed Value is :25.000000
	printed Value is :25.000000 10.000000
a<-1 $$	Case No : 1 execute and Result is : 13.000000
c<-10$$	Switch statement successfully execute !!!
a<- 5 Plus 4 $$	printed Value is :9.000000
Print Fbs a Fbc $$	ForLoop executes 4 times

```
d<-2$$



If Fbs a Lt 10 Fbc
 Sbs
 b<-2$$
 Sbc

If Fbs a Gt 11 Fbc
 Sbs
 b<-20$$
 Sbc
Else
 Sbs
  b<-25$$
 Sbc

Print Fbs b Fbc $$

Print Fbs b##c Fbc $$

c <- 1 $$
 Switch Fbs c Fbc
Sbs
Case 1:5 Plus 8 $$
Case 2:5 Plus 5 $$
Default:7 Plus 7 $$
Sbc

Print Fbs a Fbc $$

For Fbs c <- 10 $$ c Gt 2 $$ c <- c Minus 2
Fbc
Sbs
  5 Plus 3 $$
Sbc

Print Fbs a Fbc $$




Sbc



*Hi, How  are you
Are you from kuet*

@I am rakib
```

```
printed Value is :9.000000

successful compilation
MultiComment successfully executed
single Comment successfully executed
```

Features of this compiler

1. Main function
2.Comments
3.Variable declaration
4. IF Block
5. IF ELSE Block
6.Variable assignment
7. For loop
8. Print function
9. Switch Case
10.Mathematical Expression
    Addition, Subtraction, Multiplication, Division
    .

Discussion

This is a bottom up parser and the parser generate a set of tokens. In a program Conditional logic, Loops, Variable declaration, Mathematical function, array, header file are used. Unfortunately, shift reduce problem occur in the compilation time. If any grammar match with the input text then the compiler shows the token is declared.

Conclusions

This compiler is similar like the python language and this compiler written into C programming language.

References:

- https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/
- https://www.geeksforgeeks.org/bison-command-in-linux-with-examples/