UIT1601 - Compiler Design
YAAC Programs

Name:	M Badri Narayanan
Reg No:	185002018
Semester:	VI
Date:	April 4, 2021

---

1 Evaluate a Postfix Expression

Program

Lexer File

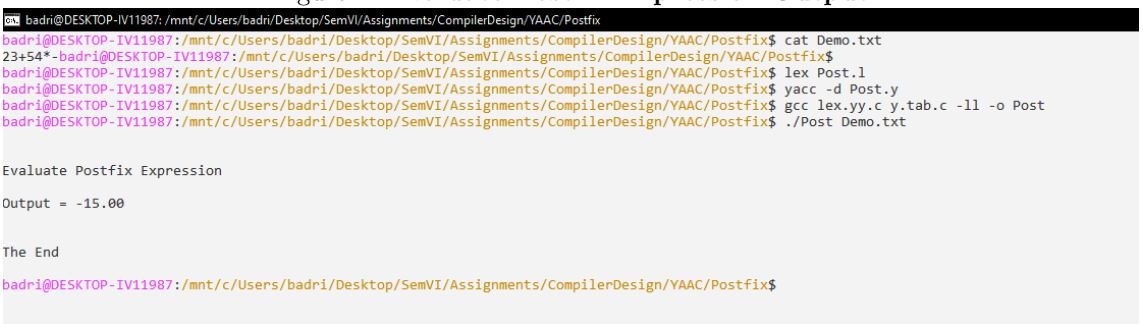
```
%{
#include <stdio.h>
#include "y.tab.h"
#include <math.h>
int yyparse();
%}
%%
[0-9] {yylval.dval=atof(yytext); return NUMBER;}
[t];
n return 0;
. {return yytext[0];}
%%
void yyerror(char * str)
{
        printf("\nInvalid Expression! \n");
}
int main(int argc, char **argv)
{
    printf("\n\nEvaluate Postfix Expression \n\n");
    if(argc != 2)
    {
        fprintf(stderr,"Please Enter file as second argument!\n");
        return 1;
    }
    yyin = fopen(argv[1], "rt");
    if(yyin == NULL)
    {
        fprintf(stderr,"File not found!\n");
        return 1;
    }
    yyparse();
    printf("\n\nThe End\n\n");
    return(0);
}
```

YAAC Program

```
%{
        #include<stdio.h>
        float output=0;
        int yylex();
        void yyerror(const char *s);
        int yyparse();
%}
%union
{
        float dval;
}
%token <dval> NUMBER
%left '+' '-'
%left '*' '/'
%nonassoc UMINUS
%type <dval> state
%type <dval> exp
%type <dval> N
%%
state : exp N {};
exp : NUMBER {$$=$1;output=$$;}
| exp exp '+' {$$=$1+$2;output=$$;}
| exp exp '-' {$$=$1-$2;output=$$;}
| exp exp '*' {$$=$1*$2;output=$$;}
| exp exp '/' {$$=$1/$2;output=$$;}
;
N : {printf("Output = %0.2f\n",output);}
;
%%
```

Output

Figure 1: **Evaluate Postfix Expression Output**



2 Identify valid arithmetic expression

Program

Lexer File

```
%{
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
int flag;
int yyparse();
%}
%%
[0-9]+ {
yylval=atoi(yytext);
return NUMBER;
}
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
    return 1;
}
void yyerror()
{
    printf("\nInvalid Arithmetic Expresion!\n");
    flag=1;
}
int main()
{
    printf("\n\nIdentify Valid Arithmetic Expression\n\n");
    printf("Enter Any Arithmetic Expression which can have operations(+,-,*,/): ");
    yyparse();
    if(flag == 0)
    {
        printf("\nValid Arithmetic Expression!\n");
    }
    printf("\n\nThe End\n\n");
    return(0);
}
```

YAAC Program

```
%{
#include <stdio.h>
int flag=0;
int yyparse();
int yylex();
void yyerror(const char *s);
%}
%token NUMBER
%left '+' '-'
%left '*' '/'
%%
ArithmeticExpression: E
{
printf("\nResult = %d\n", $$);
return 0;
};
E:E'+'E {$$=$1+$3;}
|E'-'E {$$=$1-$3;}
|E'*'E {$$=$1*$3;}
|E'/'E {$$=$1/$3;}
| NUMBER {$$=$1;}
;
%%
```

Output

Figure 2: **Identify valid arithmetic expression Output**

```
badri@DESKTOP-IV11987: /mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Arithmetic
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Arithmetic$ lex Arithmetic.l
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Arithmetic$ yacc -d Arithmetic.y
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Arithmetic$ gcc lex.yy.c y.tab.c -ll -o Arithmetic
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Arithmetic$ ./Arithmetic


Identify Valid Arithmetic Expression

Enter Any Arithmetic Expression which can have operations(+,-,*,/): 9*4/2

Result = 18

Valid Arithmetic Expression!


The End

badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Arithmetic$
```

3 Recognize the identifier that starts with a letter followed by number or digits

Program

Lexer File

```
%{
#include<stdio.h>
#include "y.tab.h"
int valid;
int yyparse();
int yylex();
int yyerror();
%}
%%
[a-zA-Z_][a-zA-Z_0-9]* return letter;
[0-9] return digit;
. return yytext[0];
\n return 0;
%%
int yywrap()
{
    return 1;
}
int yyerror()
{
    printf("\nNot an Identifier!\n");
    valid=0;
    return 0;
}
int main()
{
    printf("\n\nRecognise identifier that starts with a letter followed by number or
    ↪  digits\n\n");
    printf("Enter a name to tested for identifier: ");
    yyparse();
    if(valid)
    {
        printf("\nIt is an Identifier!\n");
    }
    printf("\n\nThe End\n\n");
    return(0);
}
```

YAAC Program

```
%{
#include <stdio.h>
int valid=1;
int yylex();
int yyerror();
%}
%token digit letter
%%
start : letter s
s : letter s
| digit s
|
;
%%
```

Output

Figure 3: **Recognize identifier Output**

```
badri@DESKTOP-IV11987: /mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Identifier
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Identifier$ lex Identifier.l
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Identifier$ yacc -d Identifier.y
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Identifier$ gcc lex.yy.c y.tab.c -ll -o Identifier
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Identifier$ ./Identifier


Recognise identifier that starts with a letter followed by number or digits

Enter a name to tested for identifier: B18

It is an Identifier!


The End

badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Identifier$
```

4 Simple calculator using lex and yaac

Program

Lexer File

```
%{
#include <stdio.h>
#include "y.tab.h"
extern int yylval;
int flag;
int yyparse();
%}
%%
[0-9]+ {
yylval=atoi(yytext);
return NUMBER; }
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
    return 1;
}
void yyerror()
{
    printf("\n Invalid Expression \n\n");
    flag=1;
}
int main()
{
    printf("\n\nArithmetic Calculator \n\n");
    printf("\nEnter an Arithmetic Expression: ");
    yyparse();
    printf("\n\nThe End \n\n");
    return 1;
}
```

YAAC Program

```
%{
#include <stdio.h>
int flag=0;
int yyparse();
int yylex();
void yyerror(const char *s);
%}
%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
ArithmeticExpression: E
{
    printf("\nResult = %d\n",$$);
    return 0;
};
E:E'+'E {$$=$1+$3;}
|E'-'E {$$=$1-$3;}
|E'*'E {$$=$1*$3;}
|E'/'E {$$=$1/$3;}
|E'%'E {$$=$1%$3;}
|'('E')' {$$=$2;}
| NUMBER {$$=$1;}
;
%%
```

Output

Figure 4: **Calculator Output**

```
badri@DESKTOP-IV11987: /mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Calculator
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Calculator$ lex Calculator.l
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Calculator$ yacc -d Calculator.y
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Calculator$ gcc lex.yy.c y.tab.c -ll -o Calculator
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Calculator$ ./Calculator


Arithmetic Calculator


Enter an Arithmetic Expression: 5*5

Result = 25


The End

badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/YAAC/Calculator$
```