
Source Code

q1.l

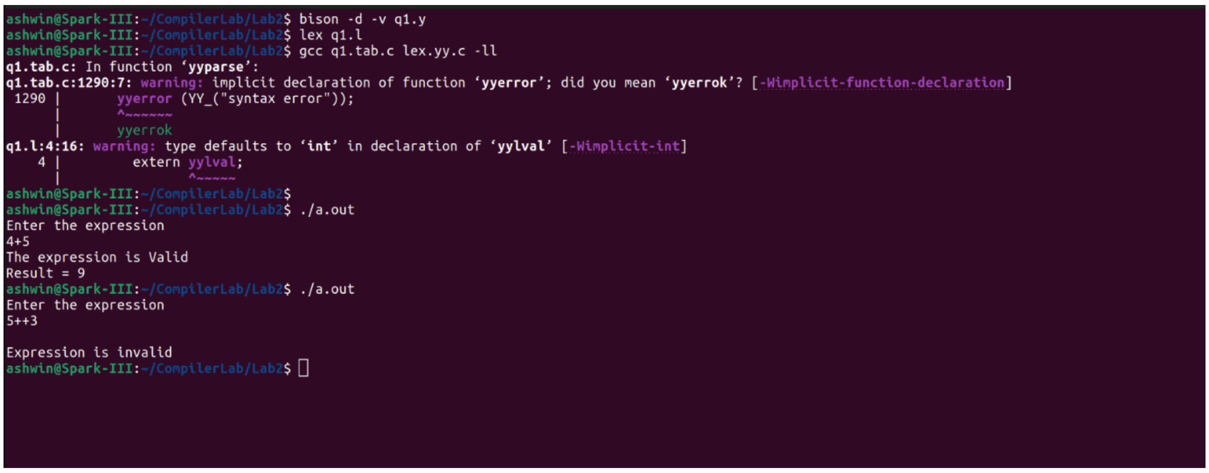
```
%{
    /* Definition section*/
    #include "q1.tab.h"
    extern yylval;
%}

%%
[0-9]+ {
            yylval = atoi(yytext);
            return NUMBER;
            }
[a-zA-Z]+ { return ID; }
[ \t]+       ; /*For skipping whitespaces*/
\n       { return 0; }
"<="    {return RELOPLTE;}
"==" {return RELOPE;}
"!="    {return RELOPNE;}
">=" {return RELOPGTE;}
"&&" {return BOOLAND;}
"||" {return BOOLOR;}
.        { return yytext[0]; }

%%

int yywrap() {
    return 1;
}
```

q1.y

```
%{
#include <stdio.h>
int yylex();
%}

%token NUMBER ID BOOLAND RELOPLTE RELOPE RELOPNE RELOPGTE BOOLOR
%right '='
%left BOOLOR
```

```
%left BOOLAND
%left RELOPE RELOPNE
%left RELOPGTE RELOPLTE '<' '>'

%left '+' '-'
%left '*' '/' '%'
%right '!'
%%
E : T    {
                printf("The expression is Valid \n");
                printf("Result = %d\n", $$);
                return 0;
            }

T :
    T '+' T { $$ = $1 + $3;}
    | T '-' T { $$ = $1 - $3; }
    | T '*' T { $$ = $1 * $3; }
    | T '/' T { $$ = $1 / $3; }
    | T '%' T { $$ = $1 % $3; }

    | T BOOLAND T {$$ = $1 && $3; }
    | T BOOLOR T {$$ = $1 || $3; }

    | T RELOPE T {$$ = $1 == $3;}
    | T RELOPNE T {$$ = $1 != $3; }
    | T RELOPGTE T {$$ = $1 >= $3; }
    | T RELOPLTE T {$$ = $1 <= $3; }
    | T '>' T {$$ = $1 > $3; }
    | T '<' T {$$ = $1 < $3; }

    | '!' NUMBER {$$ = !$1; }
    | '!' ID {$$ = !$1; }
    | '-' NUMBER { $$ = -$2; }
    | '-' ID { $$ = -$2; }
    | '(' T ')' { $$ = $2; }
    | NUMBER { $$ = $1; }
    | ID { $$ = $1; };
%%

int main() {
        printf("Enter the expression\n");
        yyparse();
}

/* For printing error messages */
int yyerror(char* s) {
    printf("\nExpression is invalid\n");}
```

Input & Output



```
ashwin@Spark-III:~/CompilerLab/Lab2$ bison -d -v q1.y
ashwin@Spark-III:~/CompilerLab/Lab2$ lex q1.l
ashwin@Spark-III:~/CompilerLab/Lab2$ gcc q1.tab.c lex.yy.c -ll
q1.tab.c: In function ‘yyparse’:
q1.tab.c:1290:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
 1290 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
q1.l:4:16: warning: type defaults to ‘int’ in declaration of ‘yylval’ [-Wimplicit-int]
    4 |         extern yylval;
      |                ^~~~~~
ashwin@Spark-III:~/CompilerLab/Lab2$
ashwin@Spark-III:~/CompilerLab/Lab2$ ./a.out
Enter the expression
4+5
The expression is Valid
Result = 9
ashwin@Spark-III:~/CompilerLab/Lab2$ ./a.out
Enter the expression
5++3

Expression is invalid
ashwin@Spark-III:~/CompilerLab/Lab2$ 
```

Source Code

lexer.l

```
%{
    #include "parser.tab.h"
    int countn=0;
%}
%option yylineno

alpha [a-zA-Z]
digit [0-9]
unary "++"|"--"

%%

"printf"                    { strcpy(yylval.nd_obj.name,(yytext)); return
PRINTFF; }
"scanf"                     { strcpy(yylval.nd_obj.name,(yytext)); return
SCANFF; }
"int"                       { strcpy(yylval.nd_obj.name,(yytext)); return INT;
}
"float"                     { strcpy(yylval.nd_obj.name,(yytext)); return
FLOAT; }
"char"                      { strcpy(yylval.nd_obj.name,(yytext)); return
CHAR; }
"void"                      { strcpy(yylval.nd_obj.name,(yytext)); return
VOID; }
"return"                    { strcpy(yylval.nd_obj.name,(yytext)); return
RETURN; }
^"#include"[ ]*<.+\.h>      { strcpy(yylval.nd_obj.name,(yytext)); return
INCLUDE; }
[-]?{digit}+                { strcpy(yylval.nd_obj.name,(yytext)); return
NUMBER; }
[-]?{digit}+\.{digit}{1,6}  { strcpy(yylval.nd_obj.name,(yytext)); return
FLOAT_NUM; }
{alpha}({alpha}|{digit})*   { strcpy(yylval.nd_obj.name,(yytext)); return ID;
}
{unary}                     { strcpy(yylval.nd_obj.name,(yytext)); return
UNARY; }
```

```
"<="                            { strcpy(yylval.nd_obj.name,(yytext)); return LE;
}
">="                            { strcpy(yylval.nd_obj.name,(yytext)); return GE;
}
"=="                            { strcpy(yylval.nd_obj.name,(yytext)); return EQ;
}
"!="                            { strcpy(yylval.nd_obj.name,(yytext)); return NE;
}
">"                             { strcpy(yylval.nd_obj.name,(yytext)); return GT;
}
"<"                             { strcpy(yylval.nd_obj.name,(yytext)); return LT;
}
"+"                             { strcpy(yylval.nd_obj.name,(yytext)); return ADD;
}
"-"                             { strcpy(yylval.nd_obj.name,(yytext)); return
SUBTRACT; }
"/"                             { strcpy(yylval.nd_obj.name,(yytext)); return
DIVIDE; }
"*"                             { strcpy(yylval.nd_obj.name,(yytext)); return
MULTIPLY; }
\/\/.*                          { ; }
\/\*(.*\n)*.*\*\/               { ; }
[ \t]*                          { ; }
[\n]                            { countn++; }
.                               { return *yytext; }
["].*["]                        { strcpy(yylval.nd_obj.name,(yytext)); return STR;
}
['].[']                         { strcpy(yylval.nd_obj.name,(yytext)); return
CHARACTER; }

%%

int yywrap() {
    return 1;
}
```

parser.y

```
%{
    #include<stdio.h>
    #include<string.h>
    #include<stdlib.h>
    #include<ctype.h>
    #include"lex.yy.c"
    void yyerror(const char *s);
    int yylex();
    int yywrap();
    void add(char);
    void insert_type();
    int search(char *);

    struct dataType {
        char * id_name;
        char * data_type;
        int size;
        int offset;
    }

    symbolTable[40];
    int count=0;
    int q;
    char type[10];
    int typesize=0;
    int globalOffset=0;

%}

%union {
    struct var_name {
        char name[100];
        struct node* nd;
    } nd_obj;
}

%token VOID
%token <nd_obj> CHARACTER PRINTFF SCANFF INT FLOAT CHAR NUMBER FLOAT_NUM ID LE
GE EQ NE GT LT STR ADD MULTIPLY DIVIDE SUBTRACT UNARY INCLUDE RETURN
%type <nd_obj> headers main body return datatype expression statement init
value arithmetic relop program

%%

program: headers main '(' ')' '{' body return '}' { ; }
;

headers: headers headers { ; }
| INCLUDE { ; }
;

main: datatype ID { } //main is like int main() here ig
;

datatype: INT { insert_type(); }
| FLOAT { insert_type(); }
| CHAR { insert_type(); }
| VOID { insert_type(); }
```

```
;

body: statement ';' { ; }
| body body { ; }
| PRINTFF { ; } '(' STR ')' ';' { ; }
| SCANFF { ; } '(' STR ',' '&' ID ')' ';' { ; }
;

statement: datatype ID { add('V'); } init { ; }
| ID '=' expression { ; }
| ID relop expression { ; }
| ID UNARY { ; }
| UNARY ID { ; }
;

init: '=' value { ; }
| { ; }
;

expression: expression arithmetic expression { ; }
| value { ; }
;

arithmetic: ADD
| SUBTRACT
| MULTIPLY
| DIVIDE
;

relop: LT
| GT
| LE
| GE
| EQ
| NE
;

value: NUMBER { ; }
| FLOAT_NUM { ; }
| CHARACTER { ; }
| ID { ; }
;

return: RETURN { ; } value ';' { ; }
| { ; }
;

%%
```

```c
int main() {
    yyparse();
    printf("\nSYMBOL   DATATYPE   SIZE    OFFSET \n");
    printf("_______________________________________\n\n");
    int i=0;
    for(i=0; i<count; i++) {
        printf("%s\t%s\t%d\t%d\t\n", symbolTable[i].id_name,
symbolTable[i].data_type, symbolTable[i].size, symbolTable[i].offset);
    }
    for(i=0;i<count;i++){
        free(symbolTable[i].id_name);
    }
    printf("\n\n");
}

int search(char *type) {
    int i;
    for(i=count-1; i>=0; i--) {
        if(strcmp(symbolTable[i].id_name, type)==0) {
            return -1;
            break;
        }
    }
    return 0;
}

void add(char c) {
    q=search(yytext);
    if(q==0 && c=='V') {
            symbolTable[count].id_name=strdup(yytext);
            symbolTable[count].data_type=strdup(type);
            symbolTable[count].size=typesize;
            symbolTable[count].offset=globalOffset;
            globalOffset+=typesize;
            count++;
    }
}

void insert_type() {
    strcpy(type, yytext);
    if(strcmp(type, "char") == 0) {
        typesize=1;
    }else{
        typesize=4;
    }
}
```

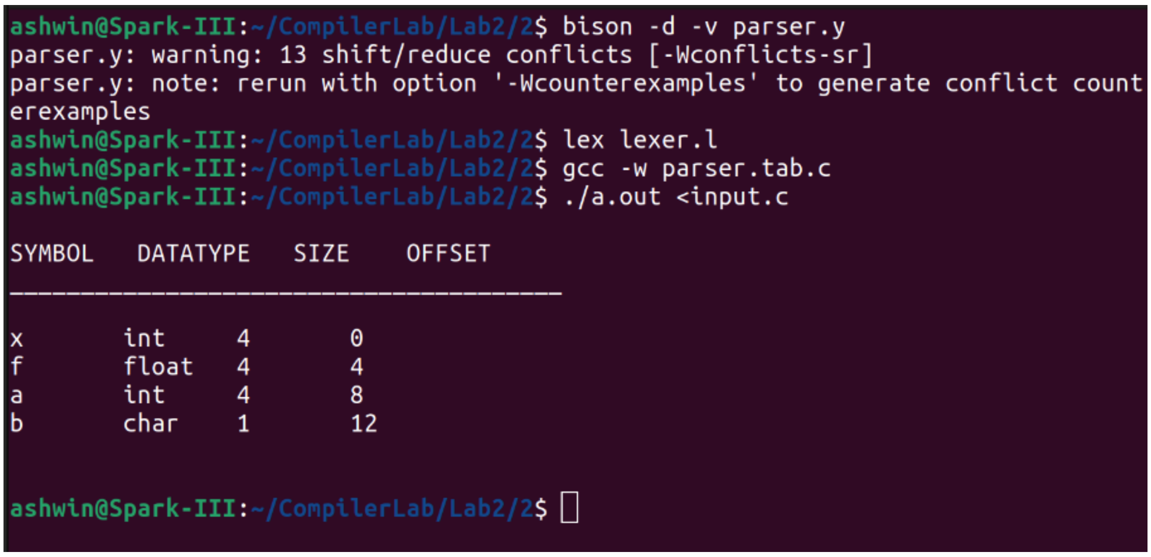
```
void yyerror(const char* msg) {
    fprintf(stderr, "%s\n", msg);
}
```

Input

input.c

```
#include<stdio.h>
#include<string.h>

int main() {
    int x=1;
    float f;
    int a=3;
    char b='a';
    a = x * 3 + 1*a;
}
```

Output



```
ashwin@Spark-III:~/CompilerLab/Lab2/2$ bison -d -v parser.y
parser.y: warning: 13 shift/reduce conflicts [-Wconflicts-sr]
parser.y: note: rerun with option '-Wcounterexamples' to generate conflict count
erexamples
ashwin@Spark-III:~/CompilerLab/Lab2/2$ lex lexer.l
ashwin@Spark-III:~/CompilerLab/Lab2/2$ gcc -w parser.tab.c
ashwin@Spark-III:~/CompilerLab/Lab2/2$ ./a.out <input.c

SYMBOL   DATATYPE   SIZE    OFFSET
---------------------------------------

x       int      4       0
f       float    4       4
a       int      4       8
b       char     1       12


ashwin@Spark-III:~/CompilerLab/Lab2/2$ 
```