西安交通大學
XI'AN JIAOTONG UNIVERSITY

2022-2023 学年第二学期

《编译器设计专题实验》

实验报告 5

学　　院：电信学部

班　　级：[redacted]

学　　号：[redacted]

姓　　名：[redacted]

二〇二三年 五月

目录

《实验 5-语法分析(一),熟悉流程和 bison 工具》

一、实验内容（必做）

1. 完成计算器实现高级运算：

要求：支持加减乘除和括号、支持负数和浮点数、支持报错（例如：除 0 错误）；

2. 完成计算器可实现布尔表达式

要求：输入以 true/false 组成的布尔表达式、分析表达式语法关系（||，&&，!）并求值、注意布尔表达式的优先级和结合顺序以及单目和多目运算符。

二、实验内容（选做）

flex，yacc 联合编程实现 SQL 编译器。

三、实验结果

（1）必做 1：算术运算

```
GuoSongjian(Thu May 11 16:49:53):~/compiler_exp/exp5$ ./calc
> 1+2
= 3.000000
> 1+2*3
= 7.000000
> 2*(1-5)
= -8.000000
> 3/0
Error: Divide 0
= inf
> -1.2 * 1.2
= -1.440000
> ^C
```

(2) 必做 2：布尔运算

```
GuoSongjian(Thu May 11 16:51:02):~/compiler_exp/exp5$ ./calc2
> true && false
= false
> true || false
= true
> !true || false
= false
> !false
= true
> ^C
```

(3) 选作：SQL 编译器

SELECT 语句：

```
GuoSongjian(Fri May 12 22:14:46):~/compiler_exp/exp5_extend$ ./sql
> SELECT CNO, SNO FROM TAB1, TAB2 WHERE CNO = 2023 AND SNO = 'XJTU'
SELECT:
Field: CNO
Field: SNO
FROM:
Table: TAB1
Table: TAB2
Condition: CNO = 2023
Condition: SNO = 'XJTU'
Condition: AND
Valid SELECT statement.
> SELECT CNO FROM
SELECT:
Field: CNO
syntax error
>
```

INSERT 语句：

```
GuoSongjian(Fri May 12 22:18:17):~/compiler_exp/exp5_extend$ ./sql
> INSERT INTO TAB (SNO, CNO) VALUES (123, 'TEST')
INSERT:
Table: TAB
Field: SNO
Field: CNO
Value: 123
Value: 'TEST'
Valid INSERT statement.
> INSERT INTO TAB (SNO, CNO)
INSERT:
Table: TAB
Field: SNO
Field: CNO
syntax error
>
```

UPDATE 语句：

```
GuoSongjian(Fri May 12 22:23:26):~/compiler_exp/exp5_extend$ ./sql
> UPDATE SC SET CNO = 12345, SNO = 'STRING' WHERE CNO < 100
UPDATE:
Table: SC
Set: CNO = 12345
Set: SNO = 'STRING'
Condition: CNO < 100
Valid UPDATE statement.
> UPDATE SC SET 123
UPDATE:
Table: SC
syntax error
> >
```

DELETE 语句：

```
GuoSongjian(Fri May 12 22:19:57):~/compiler_exp/exp5_extend$ ./sql
> DELETE FROM SC WHERE CNO <> 12 OR SNO = 'TEST'
DELETE:
Table: SC
Condition: CNO <> 12
Condition: SNO = 'TEST'
Condition: OR
Valid DELETE statement.
> DELETE FROM SC
DELETE:
Table: SC
No WHERE clause.
Valid DELETE statement.
> DELETE SC
syntax error
> >
```

四、源代码

(1) 算术运算

```
%{
# include "calc.tab.h"
extern int yyerror(const char *, ...);
%}

%%
"+"     { return ADD;  }
"-"     { return SUB;  }
"*"     { return MUL;  }
"/"     { return DIV;  }
"("     { return OP;   }
")"     { return CP;   }

([1-9][0-9]*)|0|([0-9]+\.[0-9]+) {
    double temp;
    sscanf(yytext, "%lf", &temp);
```

```
    yylval.double_value = temp;
    return DOUBLE_NUMBER;
}

\n      { return EOL;  }
[ \t]   { /* ignore white space */  }
.       { yyerror("Unknown character %c\n", *yytext);  }
%%

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

```
%{
#include <stdarg.h>
#include <stdio.h>
int yyerror(const char *, ...);
extern int yylex();
extern int yyparse();
%}

%union {
    int int_value;
    double double_value;
}
%token <double_value> DOUBLE_NUMBER
%token ADD SUB MUL DIV
%token OP CP
%token EOL
%type <double_value> exp factor term
%%

calclist: /* nothing */
    | calclist exp EOL { printf("= %lf\n> ", $2);  }
    | calclist EOL { printf("> ");  }
;

exp: factor
    | exp ADD factor { $$ = $1 + $3;  }
    | exp SUB factor { $$ = $1 - $3;  }
;

factor: term
    | factor MUL term { $$ = $1 * $3;  }
    | factor DIV term { $$ = $1 / $3;  if ($3 == 0) yyerror("Error:
Divide 0\n");  }
;
```

```
term: SUB DOUBLE_NUMBER { $$ = -$2; }
    | DOUBLE_NUMBER
    | OP exp CP { $$ = $2;  }
;
%%

int main()
{
    printf("> ");
    yyparse();
    return 0;
}

int yyerror(const char *s, ...)
{
    int ret;
    va_list va;
    va_start(va, s);
    ret = vfprintf(stderr, s, va);
    va_end(va);
    return ret;
}
```

(2) 布尔运算

```
%{
# include "calc2.tab.h"
extern int yyerror(const char *, ...);
%}

%%
"true"   { return TRUE;  }
"false"  { return FALSE;  }

"||"     {return OR; }
"&&"     {return AND; }
"!"      {return NOT; }

"("      { return OP;  }
")"      { return CP;  }

\n       { return EOL;  }
[ \t]    { /* ignore white space */  }
.        { yyerror("Unknown character %c\n", *yytext);  }
```

```
%%

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

```
%{
#include <stdarg.h>
#include <stdio.h>
int yyerror(const char *, ...);
extern int yylex();
extern int yyparse();
%}

%union {
   int bool_value;
}
%token TRUE FALSE
%token OR AND NOT
%token OP CP
%token EOL
%type <bool_value> exp factor term
%%

calclist: /* nothing */
   | calclist exp EOL { if ($2 == 1) printf("= true\n> "); else
printf("= false\n> ");  }
   | calclist EOL { printf("> ");  }
;

exp: factor
   | exp OR factor { $$ = $1 || $3;  }
   | exp AND factor { $$ = $1 && $3;  }
;

factor: NOT term { $$ =  !$2;  }
   | term { $$ = $1;  }
;

term: OP exp CP { $$ = $2;  }
   | TRUE  { $$ = 1; }
   | FALSE  { $$ = 0; }
;
%%

int main()
{
```

```
    printf("> ");
    yyparse();
    return 0;
}

int yyerror(const char *s, ...)
{
    int ret;
    va_list va;
    va_start(va, s);
    ret = vfprintf(stderr, s, va);
    va_end(va);
    return ret;
}
```

(3) SQL 编译器

```
%{
#include "y.tab.h"
extern int yyerror(const char *, ...);
%}

%%
SELECT                { return SELECT; }
FROM                  { return FROM; }
WHERE                 { return WHERE; }
AND                   { return AND; }
OR                    { return OR; }
NOT                   { return NOT; }
INSERT                { return INSERT; }
INTO                  { return INTO; }
VALUES                { return VALUES; }
UPDATE                { return UPDATE; }
SET                   { return SET; }
DELETE                { return DELETE; }
[1-9]+[0-9]*          { yylval.intval = atoi(yytext); return
INTEGER; }
[a-zA-Z]+[0-9a-zA-Z_]*  { yylval.strval = strdup(yytext); return
IDENTIFIER; }
'[^']*'               { yylval.strval = strdup(yytext); return
STRING; }
"<>"|"<"|"<="|">"|">="  { yylval.strval = strdup(yytext); return
OP; }
"="                   { return '='; }
","                   { return ','; }
```

```
"*"                  { return '*'; }
"("                  { return '('; }
")"                  { return ')'; }
[ \t]+               { /* ignore whitespace */ }
\n                   { return EOL; }
.                    { yyerror("Unknown character %c\n",
*yytext);  }
%%

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

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
int yyerror(const char *, ...);
extern int yylex();
extern int yyparse();
%}

%union {
   int intval;
   char * strval;
}
%token SELECT FROM WHERE INSERT INTO VALUES UPDATE SET DELETE EOL
%token <intval> INTEGER
%token <strval> STRING
%token <strval> IDENTIFIER
%token <strval> OP
%left AND OR
%nonassoc NOT

%%

sql: /* nothing */
   | sql select_statement EOL { printf("> "); }
   | sql insert_statement EOL { printf("> "); }
   | sql update_statement EOL { printf("> "); }
   | sql delete_statement EOL { printf("> "); }
   | sql EOL { printf("> "); }
;

select_statement: select_clause from_clause where_clause
   { printf("Valid SELECT statement.\n"); }
```

```
;

select_clause: SELECT '*' { printf("SELECT:\nField: All\n"); }
    | SELECT IDENTIFIER { printf("SELECT:\nField: %s\n", $2); }
    | select_clause ',' IDENTIFIER { printf("Field: %s\n", $3); }
;

from_clause: FROM IDENTIFIER { printf("FROM:\nTable: %s\n", $2); }
    | from_clause ',' IDENTIFIER { printf("Table: %s\n", $3); }
;

where_clause: /* empty */ { printf("No WHERE clause.\n"); }
    | WHERE condition
;

condition: expr
    | NOT condition %prec NOT { printf("Condition: NOT\n"); }
    | condition AND condition { printf("Condition: AND\n"); }
    | condition OR condition { printf("Condition: OR\n"); }
;

expr: IDENTIFIER OP INTEGER { printf("Condition: %s %s %d\n", $1,
$2, $3); }
    | IDENTIFIER OP STRING { printf("Condition: %s %s %s\n", $1,
$2, $3); }
    | IDENTIFIER OP IDENTIFIER { printf("Condition: %s %s %s\n",
$1, $2, $3); }
    | IDENTIFIER '=' INTEGER { printf("Condition: %s = %d\n", $1,
$3); }
    | IDENTIFIER '=' STRING { printf("Condition: %s = %s\n", $1,
$3); }
    | IDENTIFIER '=' IDENTIFIER { printf("Condition: %s = %s\n",
$1, $3); }
;

insert_statement: insert_clause '(' field_list ')' VALUES '('
value_list ')'
    { printf("Valid INSERT statement.\n"); }
;

insert_clause: INSERT INTO IDENTIFIER
    { printf("INSERT:\nTable: %s\n", $3); }
;
```

```
field_list: IDENTIFIER { printf("Field: %s\n", $1); }
    | field_list ',' IDENTIFIER { printf("Field: %s\n", $3); }
;

value_list: INTEGER { printf("Value: %d\n", $1); }
    | STRING { printf("Value: %s\n", $1); }
    | value_list ',' INTEGER { printf("Value: %d\n", $3); }
    | value_list ',' STRING { printf("Value: %s\n", $3); }
;

update_statement: update_clause set_clause where_clause
    { printf("Valid UPDATE statement.\n"); }
;

update_clause: UPDATE IDENTIFIER
    { printf("UPDATE:\nTable: %s\n", $2); }
;

set_clause: SET IDENTIFIER '=' INTEGER { printf("Set: %s = %d\n",
$2, $4); }
    | SET IDENTIFIER '=' STRING { printf("Set: %s = %s\n", $2,
$4); }
    | set_clause ',' IDENTIFIER '=' INTEGER { printf("Set: %s
= %d\n", $3, $5); }
    | set_clause ',' IDENTIFIER '=' STRING { printf("Set: %s
= %s\n", $3, $5); }
;

delete_statement: delete_clause where_clause
    { printf("Valid DELETE statement.\n"); }
;

delete_clause: DELETE FROM IDENTIFIER
    { printf("DELETE:\nTable: %s\n", $3); }
;

%%

int main() {
    while(1) {
        printf("> ");
        yyparse();
        printf("\n");
    }
```

```
    return 0;
}

int yyerror(const char *s, ...)
{
    int ret;
    va_list va;
    va_start(va, s);
    ret = vfprintf(stderr, s, va);
    va_end(va);
    return ret;
}
```

```
CC = gcc
LEX = flex
YACC = yacc
LIBS = -ll

all: sql

sql: y.tab.c lex.yy.c
	$(CC) -o $@ $^ $(LIBS)

y.tab.c: sql.y
	$(YACC) -d $<

lex.yy.c: sql.l
	$(LEX) $<

clean:
	rm -f sql y.tab.c y.tab.h lex.yy.c
```