關於C subset的描述

資工三 409410029 王美綺

一、Tokens

(1)關鍵字

RETURN: 'return';
DEFINE: 'define';
TYPEDEF: 'typedef';
SIZEOF: 'sizeof';

(2)資料型態（宣告變數):

CHAR_TYPE: 'char';
INT_TYPE: 'int';
SHORT_TYPE: 'short';
LONG_TYPE: 'long';
CONST_TYPE: 'const';
FLOAT_TYPE: 'float';
DOUBLE_TYPE: 'double';
UNSIGNED_TYPE: 'unsigned';
SIGNED_TYPE: 'signed';
STRUCT_TYPE: 'struct';
VOID_TYPE: 'void';
STATIC_TYPE: 'static';
VOLATILE_TYPE: 'volatile';
ENUM_TYPE: 'enum';

(3)註解方式:

COMMENT1: '//'(.)*'\n';
COMMENT2: '/*' (options{greedy=false;}: .)* '*/';

(4)program statement相關之邏輯、數學運算子

LT_OP: '<';
GT_OP: '>';
LE_OP: '<=';
GE_OP: '>=';
EQ_OP: '==';
NE_OP: '!=';

PLUS_OP: '+';
PP_OP: '++';
MINUS_OP: '-';
MM_OP: '--';
MULTIPLE_OP: '*';
DIVID_OP: '/';
MOD_OP: '%';
RSHIFT_OP: '>>';
LSHIFT_OP: '<<';
ASSIGN_OP: '=';
PA_OP: '+=';
MIA_OP: '-=';
MUA_OP: '*=';
DA_OP: '/=';
MOA_OP: '%=';
BITAND_OP: '&';
BITOR_OP: '|';
AND_OP: '&&';
OR_OP: '||';
NOT_OP: '!';
ARROW_OP: '->';
WAVE_OP: '~';
CARET_OP: '^';

(5) 其他標點符號

COMMA: ',';
SEMICOLON: ';';
LEFT_PAREM: '(';
RIGHT_PAREM: ')';
LEFT_BRACE: '{';
RIGHT_BRACE: '}';
LEFT_BRACKET: '[';
RIGHT_BRACKET: ']';
DOT: '.';
COLON: ':';

(6) 支援之程式控制結構

- 條件判斷:

IF: 'if';
ELSE: 'else';
BREAK: 'break';
SWITCH: 'switch';
CASE: 'case';
CONTINUE: 'continue';
DEFAULT: 'default';

- **迴圈:**

DO: 'do';
WHILE: 'while';
FOR: 'for';

(7) 函式

MAIN: 'main'
SCANF: 'scanf';
PRINTf: 'printf';
LITERAL : '"' (options{greedy=false;}: .)* '"';
LITERAL_CHAR: '\''(options{greedy=false;}: .)*'\'';
HEADER: '#'(options{greedy=false;}: .)*'\n';

(8) 數字和變數

DEC_NUM: ('0' | ('1'..'9')(DIGIT)*);
ID: (LETTER)(LETTER|DIGIT)*;
fragment LETTER : 'a'..'z' | 'A'..'Z' | '_';
fragment DIGIT : '0'..'9';
FLOAT_NUM: FLOAT_NUM1 | FLOAT_NUM2 | FLOAT_NUM3;
fragment FLOAT_NUM1: (DIGIT)+'.'(DIGIT)*;
fragment FLOAT_NUM2: '.'(DIGIT)+;
fragment FLOAT_NUM3: (DIGIT)+;

(9) 換行和空白

NULL: 'null'| '\\0';
NEW_LINE: '\n';
WS: (' '|'\r'|'\t')+;

二、contex free grammar

1. parser 起始點 program

```
program:HEADER{{ if (TRACEON) System.out.println("HEADER"); }} (struct | typedef | define)* VOID_TYPE MAIN'('')' '{' declarations statements RETURN DEC_NUM ';' '}'
        { if (TRACEON) System.out.println("VOID MAIN() {declarations statements return 0}"); };
```

（1）支援 HEADER #include <stdio.h>

（2）支援結尾 return 0

2. declartions(宣告變數方式)

```
declarations: type ID (',' ID)* ';' declarations
              { if (TRACEON) System.out.println("declartions: type ID, ID,... : declarations"); }
            | type ID '=' ('-'|) (DEC_NUM|FLOAT_NUM) (',' ID '=' ('-'|) (DEC_NUM|FLOAT_NUM))*';' declarations
              { if (TRACEON) System.out.println("declarations: type ID = NUM, ID = NUM,...: declarations"); }
            | type ID ('[' DEC_NUM ']') (',' ID ('[' DEC_NUM ']'))* ';' declarations
              { if (TRACEON) System.out.println("declarations:type (ID[])* ");}
            | { if (TRACEON) System.out.println("declarations: ");} ;


type: (CONST_TYPE|UNSIGNED_TYPE|SIGNED_TYPE|ENUM_TYPE|VOLATILE_TYPE|STATIC_TYPE|)
    (INT_TYPE ('*')* { if (TRACEON) System.out.println("type: INT"); }
    |FLOAT_TYPE ('*')*  { if (TRACEON) System.out.println("type: FLOAT"); }
    |CHAR_TYPE ('*')*  { if (TRACEON) System.out.println("type: CHAR"); }
    |DOUBLE_TYPE ('*')*  { if (TRACEON) System.out.println("type: DOUBLE"); }
    |SHORT_TYPE ('*')*  { if (TRACEON) System.out.println("type: SHORT"); }
    |LONG_TYPE ('*')*  { if (TRACEON) System.out.println("type: LONG"); }
    |VOID_TYPE ('*')*  { if (TRACEON) System.out.println("type: VOID"); });
```

(1) 支援型態包含第一大題 token 資料型態的所有，比較特別的包含 struct、typedef 、define

```
struct: STRUCT_TYPE ID '{' declarations '}' ';' { if (TRACEON) System.out.println("type: STRUCT"); };
typedef: TYPEDEF type ID ';' { if (TRACEON) System.out.println("type: typedef"); };
define: '#' DEFINE ID (FLOAT_NUM|DEC_NUM) { if (TRACEON) System.out.println("type: define"); };
```

```
#define width 80
typedef unsigned char mail;
struct ans{
        int a;
};
```

(3) statements

```
statement: ID '=' arith_expression ';'
         { if (TRACEON) System.out.println("ID = arith_expression;"); }
         | ID (PA_OP|MIA_OP) ID ';'
         { if (TRACEON) System.out.println("ID(+|-)=ID"); }
         | ID (PP_OP|MM_OP|MUA_OP|DA_OP|MOA_OP) ';'
         { if (TRACEON) System.out.println("ID(++|--)"); }
         | IF '(' arith_expression ')' '{' statements (BREAK ';'|CONTINUE';'|)'}' (ELSE IF '(' arith_expression')' '{' statements (BREAK ';'|CONTINUE';'|) '}')* (ELSE '{' statements (BREAK';'|
CONTINUE';'|) '}'|)
         { if (TRACEON) System.out.println("if if-elseif if-elseif-else"); }
         | FOR '(' (((type|) ID '=' ('-'|) (DEC_NUM|FLOAT_NUM))|) ';' (arith_expression|) ';' (arith_expression|) ')' '{' statements '}'
         { if (TRACEON) System.out.println("for(type ID = NUM; arith_expression; arith_expression) for(;;)"); }
         | WHILE '(' arith_expression ')' '{' statements (BREAK|CONTINUE|) '}'
         { if (TRACEON) System.out.println("while(arith_expression) {statements (break|continue|)}");}
         | SWITCH '(' ID ')' '{' ((CASE (ID | ((-'|)(DEC_NUM|FLOAT_NUM)) ':')* statements BREAK ';')* (DEFAULT ':' statements BREAK ';'|) '}'
         { if (TRACEON) System.out.println("switch(ID){ (case ID|NUM: statements break;)* default: statements break;");}
         | DO '{' statements '}' WHILE '(' arith_expression ')' ';'
         { if (TRACEON) System.out.println("do{ statements } while(arith_expression);");}
         | PRINTF '(' '"' printf_expression '"' ')' ';'
         { if (TRACEON) System.out.println("printf(literal,ID*)");}
         | SCANF '(' '"' scanf_expression '"')';'
         { if (TRACEON) System.out.println("scanf(literal,&ID*)");}
         | (COMMENT1 | COMMENT2)
         { if (TRACEON) System.out.println("COMMENT");}
         | PRINTF '(' '"' (ID|DEC_NUM|FLOAT_NUM) (ID|DEC_NUM|FLOAT_NUM)* '"' ')' ';'{ if (TRACEON) System.out.println("printf(text)");};
```

```
printf_expression: (PRINTF_D|PRINTF_F) printf_expression b | '"';
scanf_expression: (PRINTF_D|PRINTF_F) scanf_expression bb| '"';
b: ',' ID;
bb: ','&' ID;
PRINTF_D:'%d';
PRINTF_F:'%f';
```

- Assign
- If (含 if / if-else if / if-else if-else)
- For
- While
- Printf
- 支援基本的算術運算的 statement
- Scanf
- Switch case
- Do while
- Comment

紅色為額外功能

- **Assign**

```
a = a * (5+20*3/5) + 20;
```

test.c

```
ID = arith_expression;
```

- **if / if()else if / if()else if()else**

(1) if 這邊我是一起做得所以只要 parser 看到這三型其中一種都會印 if if-elseif if-elseif-else

(2)else if 這邊我設計是可以跑多個例如

```
while(people < 4){

        if(people == 0){
                printf("a get 100 scores");
                continue;
        }
        else if(people == 1){
                printf("b get 10 scores");
        }
        else if(people == 2){
                printf("c get 90 scores");
        }
        else{
                printf("d get 20 scores");
        }
        if(people == 3){
                break;
        }

}
```

test2.c

這邊我舉例用 2 個，也可以多個

```
if(average < pass_score){
        printf("average failed");
}
else{
        printf("average pass");
}
```

test2.c

不一定要有 else if 在中間，

也可以沒有 else

```
if(a < 100){

        printf("nothing");
}
```

test.c

(3)支援 **break** 和 **continue**

- **for-loop**

(1)這邊我支援可以把變數型態寫在 for 裡例如(也可以不用寫在裡面)

```
for(int i = 1; i < 5; i ++){
        a = a * i;
}
```

int 型態宣告 i 可寫在 for 裡

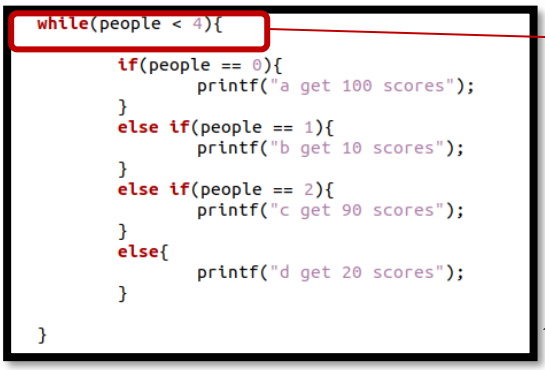
test.c

(2)有支援特殊，for 裡甚麼都不放

```
for(;;){ }
```

test.c

- **While-loop(基本款)**



while 裡的判斷式會在可以放各式各樣

test2.c

(1) 支援**break**和**continue**也可以不用放

- **支援基本的算術運算的 statement**

```
average = (a + b + c + d) / 4;
```

test2.c

```
a = a * b;
b ++;
```

test.c

```
a = a * (5+20*3/5) + 20;
```

test.c

可以是數字和變數、數字和數字，變數和變數，都可以計算

- **至少支援呼叫固定參數 (一個與兩個參數) 的 printf function**

(1) '%d'和變數個數一定要相同不同會跳 error

正常情況:	不常情況:
`printf("%d", a);`	`printf("%d %d", a);` `printf("%d", a, b, c, d);`
`printf(literal,ID*)`	test.c line 17:18 mismatched input ')' expecting COMMA test.c line 17:15 mismatched input ',' expecting RIGHT_PAREM

(2) printf裡可以放文字

```c
if(people == 0){
        printf("a get 100 scores");
}
else if(people == 1){
        printf("b get 10 scores");
}
else if(people == 2){
        printf("c get 90 scores");
}
else{
        printf("d get 20 scores");
}
```

test2.c

- **scanf(額外製作功能)**

```c
scanf("%d", &d); //input
```

test.c

```
scanf(literal,&ID*)
```

與 printf 一樣若數量不一致會跳 error

- **switch case(額外製作功能)**

```c
switch(b){

        case 0:
                printf("even");
                break;
        case 1:
                printf("odd");
                break;
        default:
                printf("Wrong");
                break;

}
```

test3.c

- **do-while(額外製作功能)**

```c
do{
        a = a * b;
        b ++;
}while(a < 300);
```

test.c

- **COMMENT(額外製作功能)**

```
scanf("%d", &d); //input
```

```
a = a * b; /* a bigger */
b ++;
c--;
```

支援兩種形式的註解

補:這些功能裡面都可以再包各式各樣的功能

```
while(people < 4){

        if(people == 0){
                printf("a get 100 scores");
        }
        else if(people == 1){
                printf("b get 10 scores");
        }
        else if(people == 2){
                printf("c get 90 scores");
        }
        else{
                printf("d get 20 scores");
        }

}
```

Ex. while 裡包 if

(4) 算術邏輯支援

```
arith_expression: ('!' | '~')* multExpr( '+' multExpr | '-' multExpr | '<' multExpr | '>' multExpr | LE_OP multExpr | '>''=' multExpr | EQ_OP multExpr | ('&'('&')) multExpr) | ('|'('|')) multExpr) |
 '!''=' multExpr | PP_OP | MM_OP )*;
multExpr: signExpr ('*' signExpr | '/' signExpr | '%' signExpr)*;
signExpr: primaryExpr | '-' primaryExpr;
primaryExpr: DEC_NUM | FLOAT_NUM | ID | '(' arith_expression ')';
```

! / ~ / + / - / < / > / >= / == /&& / || / & / | / != / ++ / -- / * / '/' / % / ++/ --
/+= / -= / << >> / ^ / -> / *= / /= / %=