Project2 Syntax analyzer C subset 描述:

Subset:

Data type:

Int, float, double, long, long long, char

Keywords:

If, else if, else, for, while, break, continue

constant :

Integer\_constant:'0'..'9'+;

Floating\_point\_constant:'0'..'9'+ '.' '0'..'9'+;

Char\_constant: '\''(.)?'\'';

String\_constant: '"'(.)\*'"';

特殊token: identifier, WS, COMMENT

在parsing 的處理時，我使用main(){ statments}作為start symbol

(int main 或 void main 都可以)。

statements : statement statements| (epsilon)

而statement有可能由以下的expressions (Expr)組成:

1. primaryExpr:

Integer\_constant | Floating\_point\_constant | Identifier | '(' arith\_expression ')'

作為parsing計算類的expressiont處理時最優先被處理的行為。

1. arith\_expression: multExpr('+' multExpr| '-' multExpr)\*

作為 + 或 – 的parsing 單位，計算類的expressiont處理時最後被處理的行為。

1. signExpr: primaryExpr | '-' primaryExpr

primaryExpr 的正負號，是計算類expression 第二處理的計算行為

1. multExpr: signExpr( '\*' signExpr | '/' signExpr| '%' signExpr)\*

x 或 / 的parsing單位，計算類statements第三處理的行為。

1. boolExpr:

(options{backtrack=true;}:

'!' '(' boolExpr ')'|

primaryExpr('>' | '>=' | '<' | '<=' | '&' | '&&' | '==' | '|' | '||' | '^' )primaryExpr

primaryExpr

)

為回傳值可以當作boolean值的計算類statements

1. argExpr:

( options{backtrack=true;}:

(primaryExpr| Char\_constant| String\_constant) (',' (primaryExpr| Char\_constant| String\_constant))\* | (epsilon) )

)

作為參數的parsing rule, 可以是沒有參數(即epsilon) 或是1個或是n個。

e.g. a,b,c , a

我的statement 包含以下:

1. 宣告: data type identifier

Parsing 的結果分成兩種:

無初始化的宣告: e.g. int a;

就只會印出 data\_type declaration 而已；

有初始化的宣告: e.g. int a = 5;

我則把後面初始化的部分視為assign，先印出data\_type，然後進行assign 的parsing，然後印出(init-statment)，最後印出declaration-statement。

1. assign :

有normal assign形式為 identifier ‘=’ arith\_expression ‘;’

Normal assign會進去arith\_expression 裡面進行parsing。

也有iterative assign : i++, i--, ++i, --i的形式。

1. condition\_statements:

包括

if\_statement ( if(boolExpr) ),

else\_if\_statement ( else\_if(boolExpr) ),

else\_statement( else )

接在condition\_statements 後面的statements我將它們的格式寫成

if\_then\_statement: statement | ‘{‘ statements ‘}’

在進入if\_then\_statement parsing之前，會印出 if-then-statement Begin

這一次的if\_then\_statement parsing完之後，會印出 if-then-statement End

1. loop\_statment

包括

for\_statement (

for( (assign|declaration\_with\_init) boolExpr ‘;’ Identifier '=' arith\_expression) ),

e.g.: for(int I = 0;i<5;i=i+1),

while\_statement ( while(boolExpr) )

在loop\_statement後面的statement s我將它們的格式寫成

in\_loop\_statement: statement | ‘{‘ statements ‘}’

在parsing In\_loop\_statement 之前，會印出 (in loop)

Paring 完後會印出 (loop-end)

1. CONTINUE ';'
2. BREAK ';'
3. printf: ‘printf’ ‘(‘ argExpr ‘)’:

跟function 基本一樣，為了printf特別寫一個parsing rule

1. function: identifier ‘(’ argExpr ‘)’

作為識別函數呼叫的paring rule

1. return\_statment :

RETURN (Identifier | Integer\_constant | Floating\_point\_constant | Char\_constant

| String\_constant | ) ';'

作為return 運算使用。

輸出:

上述9類statement在一開始都會印出 statement ->

然後進入paring。等到這個statement結束前，會把該statement給印出來。

e.g. else if(a < b){ a = a\*(2+1);} (在test2.c 可以找到)

statement->

match Id, primary-Expr // a

match Id, primary-Expr // b

boolExpr // a<b

else-if statement // if(a<b)

(if-then-statement Begin) // {

statement->

match Id, primary-Expr // a

match pri, signExpr // a

match int-constant, primary-Expr // 2

match pri, signExpr // 2

mulExpr // 2

match int-constant, primary-Expr // 1

match pri, signExpr // 1

mulExpr // 1

match + arithExpr // 2+1

match (), primary-Expr // (2+1)

match pri, signExpr // (2+1)

match \*, mulExpr // a\*(2+1)

arithExpr // a\*(2+1)

normal assign statement // a = a\*a(2+1);

(if-then-statement End) // }

Test file的特色:

Test1.c : for loop, while loop

Test2.c : if ,else if, else

Test3.c : printf(“”, a…), function