

## & 用 C 語言寫簡易 C 語言 compiler

- 簡介：

當使用者輸入一 statement，此 compiler 會對其作回應。若使用者輸入的指令不合乎文法，此 compiler 會指出使用者犯了以下三種形式的錯誤：

1. **undefined identifier**
2. **unexpected token**
3. **unrecognized token**

( 設計時，錯誤優先順序為：3->2->1 )

( 先抓 unrecognized 比如@，為 lexical error。

接著抓 unexpected token 比如  $a++b*/c$  的 /，

為 syntactic error。

最後抓 undefined identifier，為 semantic error。)

- 作法：

將 C 語言的每一行指令依 token 拆開，在合併時依 parse tree 走，若成功走完便是一合乎文法的 statement。

- 使用方式：

一開始會要求使用者輸入一任意數字當 testNum，

按 Enter 後會輸出 Our-C running 即可使用此系統。

使用者可輸出任意 C 語言指令，直到使用者輸入 Done();

結束。

- 下圖是簡易版 C 之 compiler 執行結果：

```
Our-C running ...
String s;
> Line 1: undefined identifier: 'String'
string int;
> Line 1: unexpected token: 'int'
int hi, hi2[20], hi3 [30], hi4;
> Definition of hi entered ...
Definition of hi2 entered ...
Definition of hi3 entered ...
Definition of hi4 entered ...
hi = 10;
> update value = 10
Statement executed ...
hi = 10*((hi+1)*2)+3)+4;
> update value = 254
Statement executed ...
bool IsFrontBigger( int &a, int &b ) ( // if a>b true else false
> if( a > b )
{
    return true;
}
else
    return false;
}
Definition of IsFrontBigger() entered ...
x
```

Handwritten annotations on the screenshot:

- Red arrows pointing to "undefined identifier: 'String'" and "unexpected token: 'int'" with the text "← undefined identifier" and "← unexpected token" respectively.
- A blue arrow pointing to "update value = 10" with the text "← update value".
- A blue arrow pointing to "Definition of IsFrontBigger() entered ..." with the text "← Definition of function".

- 文法節錄 ( 以 EBNF 表示 ) :

```
user_input
: ( definition | statement ) { definition | statement }

definition
:          VOID Identifier function_definition_without_ID
| type_specifier Identifier function_definition_or_declarators

type_specifier
: INT | CHAR | FLOAT | STRING | BOOL

function_definition_or_declarators
: function_definition_without_ID
| rest_of_declarators

rest_of_declarators
: [ '[' Constant ']' ]
  { ',' Identifier [ '[' Constant ']' ] } ';'

function_definition_without_ID
: '(' [ VOID | formal_parameter_list ] ')' compound_statement

formal_parameter_list
: type_specifier [ '&' ] Identifier [ '[' Constant ']' ]
  { ',' type_specifier [ '&' ] Identifier [ '[' Constant ']' ] }

compound_statement
: '{' { declaration | statement } '}'

declaration
: type_specifier Identifier rest_of_declarators

statement
: ';'          // the null statement
| expression ';' /* expression here should not be empty */
| RETURN [ expression ] ';'
| compound_statement
| IF '(' expression ')' statement [ ELSE statement ]
| WHILE '(' expression ')' statement
| DO statement WHILE '(' expression ')' ';'

expression
: basic_expression { ',' basic_expression }

basic_expression
: Identifier rest_of_Identifier_started_basic_exp
| ( PP | MM ) Identifier rest_of_PPMM_Identifier_started_basic_exp
| sign { sign } signed_unary_exp romce_and_romloe
| ( Constant | '(' expression ')' ) romce_and_romloe

rest_of_Identifier_started_basic_exp
: [ '[' expression ']' ]
  ( assignment_operator basic_expression
    |
    [ PP | MM ] romce_and_romloe
  )
| '(' [ actual_parameter_list ] ')' romce_and_romloe
```