Compiler Design

Programming HW2

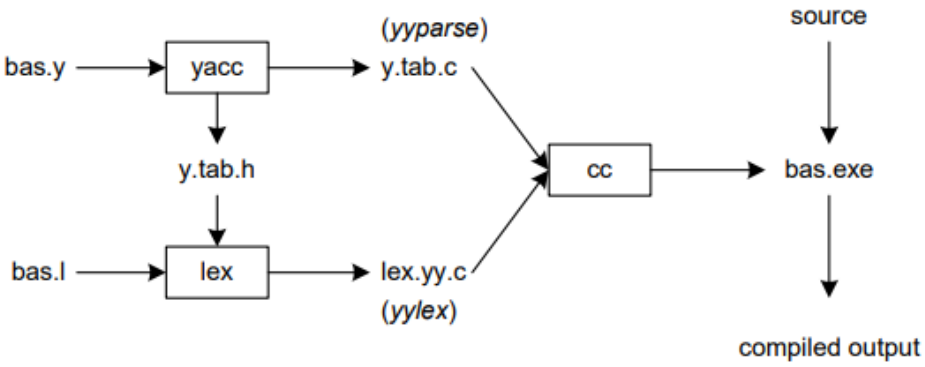
Parser

Student Name: 張睿麟
Student ID: B11015030

目錄：

1. 程式架構說明：

如下圖所展示的內容，yacc 會產生 y.tab.c 檔，內含 yylval 的 union 型別內容，當 lex 在 scan 輸入程式時，就會使用到 y.tab.c 將 Token 從 yytext 中賦值。所以 yacc 與 lex 編譯的順序為前者再來是後者。編譯成執行檔後，將要解析的程式碼(*.qv)檔案，輸入給該執行檔，並輸出解析結果，儲存成 resultCheck.c。



2. Lex 程式說明：

使用上個作業的檔案，當讀到每個 Regex 時，會 Return 對應的 Token，若有必要會設 yylval 的值。將原本寫在 lex 的主程式（main function）寫到 yacc 的檔案中。

```
"fun"               { return (FUN); }
"main"              { return (MAIN); }
"ret"               { return (RET); }
"print"             { return (PRINT); }
"println"           { return (PRINTLN); }
"=="                { printf("EQUAL_CMP: %d, %s\n", T_EQUAL_CMP, yytext); }
"!="                { printf("INEQUAL_CMP: %d, %s\n", T_INEQUAL_CMP, yytext); }
">"                 { printf("LARGER_THAN: %d, %s\n", T_LARGER_THAN, yytext); }
"<"                 { printf("SMALLER_THAN: %d, %s\n", T_SMALLER_THAN, yytext); }
">="                { printf("LARGER_EQUAL: %d, %s\n", T_LARGER_EQUAL, yytext); }
"<="                { printf("SMALLER_EQUAL: %d, %s\n", T_SMALLER_EQUAL, yytext); }

"\\\\"              { printf("ESCAPE_SEQUENCE: %d, %s\n", T_ESCAPE_SEQUENCE, yytext); }
"\\n"               { printf("ESCAPE_SEQUENCE: %d, %s\n", T_ESCAPE_SEQUENCE, yytext); }
"\\t"               { printf("ESCAPE_SEQUENCE: %d, %s\n", T_ESCAPE_SEQUENCE, yytext); }
"\\'"               { printf("ESCAPE_SEQUENCE: %d, %s\n", T_ESCAPE_SEQUENCE, yytext); }
"\\""               { printf("ESCAPE_SEQUENCE: %d, %s\n", T_ESCAPE_SEQUENCE, yytext); }
"\\?"               { printf("ESCAPE_SEQUENCE: %d, %s\n", T_ESCAPE_SEQUENCE, yytext); }

{TERMINAL}          { return (yytext[0]); }
{INT}               { yylval.node.sval = strdup(yytext); yylval.node.type = strdup("int");
return (NUMBER); }
{REAL}              { yylval.node.sval = strdup(yytext); yylval.node.type = strdup("double");
return (NUMBER); }
{IDENTIFIER}        { yylval.node.sval = strdup(yytext); yylval.node.type = strdup("unknown");
return (IDENTIFIER); }
```

3. Yacc 程式說明：

a. Union type

Union type 會定要 Token 可能的型別，Token 可以為 int, float, char*也可以是 struct 型別。這裡定義的型別，Token 只能為其中一種，且在 lex 讀到特定 Token 時，就可以將對應的值賦值上去。

```
%union {
    int ival;
    float fval;
    char *sval;
    struct Node{
        char *sval;
        char *type;
    }node;
    struct ExpressionNode{
        char *sval;
        char *type;
        int size;
    }expression_node;
};
```

b. Tokens, Terminals, Non-terminals

從 Lex 讀到的關鍵字，並將關鍵字定義一個 Token name 後，就可在 Production Rule 分析，Tokens 及 Non-terminals 可以在前面進行型別宣告，以便後續作翻譯時，能夠取得必要的值，不同的 non-terminals 宣告不同的型別，也可以減少資源的使用，和提升程式設計師撰寫時的效率。

```
%token <sval> FUN MAIN VAR VAL INT REAL PRINT PRINTLN RET
%token <node> NUMBER IDENTIFIER STRING_LITERAL
%type <sval> program functions function blocks block declarations declaration
%type <sval> type statements statement param_declarations param_declaration
%type <expression_node> expression vector term factor array_declaration
```

c. Symbol Table

定義 symbol tables 的型別為 struct，並使用 Linked-list 的方式建立 stack 特性的 symbol table 具有不同層級的關係，為了方便使用 symbol table，建立了「新增」、「刪除」、「列印」、「全局搜尋」、「該層搜尋」等函式。

```
typedef struct {
    char *idName;
    char *type;
    int size;
} Symbol;

typedef struct SymbolTable {
    Symbol symbols[MAX_SYMBOL_TABLE_SIZE];
    int top;
    struct SymbolTable *next;
} SymbolTable;

SymbolTable *createSymbolTable();
void pushSymbolTable();
void popSymbolTable();
void addSymbolTable(char* name, char* type, int size);
char* searchType(char* name);
int searchSize(char* name);
int isExist(char* name);
int isExistCurrentScope(char* name);
char* searchTypeCurrentScope(char* name);
void printSymbolTableStack();
```

d. 其他函式

建立不同的函式，用於在做翻譯工作時，可以達成特定的需求，如「將 vector 以 0 填滿」、「強制轉型」、「四則運算的型別檢查」。

```
char* vectorFillZeros(const char* s, int size);
char* typeCoercion(char* type1, char* type2);
bool arithmeticTypeChecking(char* type1, char* type2);
```

e. Error

根據程式輸入狀況以及需求，設計對應的報錯內容，以達到編譯器設計之目的。

```
ERROR: duplicate declaration
ERROR: too many dimensions
ERROR: cannot assign to val type
ERROR: mismatched dimensions
ERROR: two variables types are not compatible
ERROR: not found the identifier
```

f. 設計特色（design feature）

i. 型別（type）

因為考慮到 qv 語言較簡單，所以 symbol table 儲存型別時，會大致分為 int 和 double(real)和其他這三類，再依據宣告不同的樣態，給予前綴和後綴。例如，function 就讓相關的 identifier 的型別叫做 int-function，若為 vector，且內容值為 int 型別，則定義為 int-vector，若為 val 的 int 則稱 const-int。

ii. 陣列(vector <-> array)

在 symbol table 和部分的 non-terminals 紀錄 size 此資訊，可以知道 vector 型別的 identifier 的 dimension 大小。所以可以在宣告陣列變數時，得知是否有宣告太多的變數，或是可將尚未填滿的空間設值為 0。

也是因為有使用 symbol table 紀錄，無須刻意新增新的 operator（例如 'ip' for inner-product）即可在該 semantic action 中判別型別，區分陣列和非陣列的不同運算需求。

iii. Name scoping

使用 linked-list 實現並維護不同 scope 的 identifier，以此實現在 customize function 宣告的參數變數，不會與外為和主程式中的 identifier 互相影響。在解析到 '{' '}' 花括號時，新增及刪除當前的 symbol table，達成並排和巢狀 scope。

```
block { pushSymbolTable(); } '{' blocks '}' { popSymbolTable(); } blocks
```

iv. 已宣告函式

為了實現原始 C 語言無法達成的事情，例如使用 print() 函式列印陣列，或是直接使一維陣列做相加，則使用已經寫好的函式，並在程式的 header 區塊中添加函式，函式完整內容可詳見附件。

v. 運算子的優先級別

透過設計 production rule，達成不同運算子的優先順序，無需透過 yacc 提供的功能達成（可詳見 iThome 鐵人賽 Yacc 優先級介紹[2]），負號以及括號有最高優先級，再者是乘法及除法，最後才是加法和減法。

```
expression:
          term
          | expression '+' term
          | expression '-' term
term:
          term '*' factor
          | term '/' factor
          | factor
factor:
          '(' expression ')'
          | '-' factor
          | IDENTIFIER
          | IDENTIFIER '(' vector ')'
          | NUMBER
          | STRING_LITERAL
          | '{' vector '}'
```

vi. Real(double)輸出格式

因為使用 ’%f’ 或 ’%lf’ 時，他們會輸出小數點後的數字 0，所以將 real 型別轉換成 C 語言的輸出格式時，使用 ’%g’ 格式化輸出，可使輸出內容簡潔。

4. Makefile 檔案：

a. `$make test TEST_INPUT=<testFilePath>`
輸入此指令，並將 <testFilePath> 換成測資的檔案路徑，將會編譯 yacc 和 lex 並且使用 gcc 編譯、產生執行檔，接著把測資檔案交給執行檔分析，並產生 C code 寫成 resultCheck.c。

b. `$make testC`
輸入此指令，直接將該目錄底下的 resultCheck.c 使用 gcc 編譯並執行，並將結果顯示在 terminal（終端機）。

c. `$make clean`
將刪除 parser 執行檔、y.tab.c、y.tab.h、lex.yy.c 檔案。

```
# Compiler and flags
CC = gcc
CFLAGS = -ll

# Bison and Flex files
BISON_FILE = parser.y
FLEX_FILE = scanner.l

# Generated files
BISON_C_FILE = y.tab.c
BISON_H_FILE = y.tab.h
FLEX_C_FILE = lex.yy.c

# Output executable
OUTPUT = syntax_analyzer

# Test input file
TEST_INPUT ?= test_input.txt

# Test C file
TEST_C_FILE = checkResult.c
TEST_C_OUTPUT = checkResult

.PHONY: all clean test

all: $(OUTPUT)

$(OUTPUT): $(FLEX_C_FILE) $(BISON_C_FILE)
  $(CC) -o $@ $^ $(CFLAGS)

$(BISON_C_FILE) $(BISON_H_FILE): $(BISON_FILE)
  yacc -d $(BISON_FILE)

$(FLEX_C_FILE): $(FLEX_FILE)
  lex $(FLEX_FILE)

test: $(BISON_C_FILE) $(FLEX_C_FILE) $(OUTPUT)
  ./$(OUTPUT) < $(TEST_INPUT)

testC: $(TEST_C_FILE)
  $(CC) -o $(TEST_C_OUTPUT) $(TEST_C_FILE)
  ./$(TEST_C_OUTPUT)

clean:
  rm -f $(OUTPUT) $(BISON_C_FILE) $(BISON_H_FILE) $(FLEX_C_FILE)
```

5. 測資結果：

a. 基本測資（老師提供的 sample test data）：

sample1.qv

Variable declaration 1 (int) **(compulsory)**

```
fun main () {
    var i: int;
    i = 10;
    print(i);
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_id_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_id_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int main() {
    int i;
    i = 10;
    printf("%d", i);
}
```

Output

```
jovi@zhangruilindeMacBook-Air:~/Syntax-Analyzer ⌥⌘1
✘ jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main make test TEST_INPUT=tes
t.qv
./syntax_analyzer < test.qv
Symbol Table Stack:
--- Symbol Table ---
------------- Info -------------
| Compile Complete.             |
| Generating C code...          |
| Please checkout output.c file. |
--------------------------------
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ± make testC
gcc -o checkResult checkResult.c
./checkResult
10%
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ±
```

sample2.qv

Variable declaration 2 (int and real) **(compulsory)**

```
fun main () {
    var i: int = 10;
    var j: real = 3.1415;
    print(i);
    print(j);
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_id_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_id_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int main() {
    int i = 10;
    double j = 3.1415;
    printf("%d", i);
    printf("%g", j);
}
```

Output

```
jovi@zhangruilindeMacBook-Air:~/Syntax-Analyzer ⌥⌘1
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ± make test TEST_INPUT=tes
t.qv
./syntax_analyzer < test.qv
Symbol Table Stack:
--- Symbol Table ---
-------------- Info --------------
| Compile Complete.              |
| Generating C code...           |
| Please checkout output.c file. |
----------------------------------
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ± make testC
gcc -o checkResult checkResult.c
./checkResult
103.1415%
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ±
```

sample3.qv

Simplest type system (int and real) **(compulsory)**

```
fun main () {
    var i: int = 10;
    var j: real = 3.1415;
    var k: int = i + j;
    var l: real = i + j;
    print(k);
    print(l);
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_id_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_id_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int main() {
    int i = 10;
    double j = 3.1415;
    int k = i + j;
    double l = i + j;
    printf("%d", k);
    printf("%g", l);
}
```

Output

```
jovi@zhangruilindeMacBook-Air:~/Syntax-Analyzer ⌥⌘1
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer  main ± make test TEST_INPUT=tes
t.qv
./syntax_analyzer < test.qv
Symbol Table Stack:
--- Symbol Table ---
-------------- Info --------------
| Compile Complete.              |
| Generating C code...           |
| Please checkout output.c file. |
----------------------------------
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer  main ± make testC
gcc -o checkResult checkResult.c
./checkResult
1313.1415%
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer  main ±
```

sample4.qv

Arithmetic (int and real) 1 **(compulsory)**

```
fun main () {
    var radius: real = 5;
    var pi: real = 3.1415;
    var area: real = radius * radius * pi;
    print(area);
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_id_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_id_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int main() {
    double radius = 5;
    double pi = 3.1415;
    double area = radius * radius * pi;
    printf("%g", area);
}
```

Output

```
jovi@zhangruilindeMacBook-Air:~/Syntax-Analyzer ⌥⌘1
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ± make test TEST_INPUT=tes
t.qv
./syntax_analyzer < test.qv
Symbol Table Stack:
--- Symbol Table ---
-------------- Info --------------
| Compile Complete.               |
| Generating C code...            |
| Please checkout output.c file. |
----------------------------------
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ± make testC
gcc -o checkResult checkResult.c
./checkResult
78.5375%
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ±
```

sample5.qv

Operator precedence **(compulsory)**

```
fun main () {
    var i: real = 1.5;
    var j: real = 3.14;
    var k: real = 2.8;
    print(i + j * k);
    print("\n");
    print(i * (j + k));
    print("\n");
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_id_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_id_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int main() {
    double i = 1.5;
    double j = 3.14;
    double k = 2.8;
    printf("%g", i + j * k);
    printf("\n");
    printf("%g", i * ( j + k ));
    printf("\n");
}
```

Output

```
jovi@zhangruilindeMacBook-Air:~/Syntax-Analyzer
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ± make test TEST_INPUT=tes
t.qv
./syntax_analyzer < test.qv
Symbol Table Stack:
--- Symbol Table ---
-------------- Info --------------
| Compile Complete.              |
| Generating C code...           |
| Please checkout output.c file. |
----------------------------------
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ± make testC
gcc -o checkResult checkResult.c
./checkResult
10.292
8.91
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ±
```

sample6.qv

Vector declaration and successful arithmetic **(compulsory)**: 1D vector with at most 100 dimensions.

```
fun main () {
    var vi1: real[5] = {5, 3, 4, 1, 2}; // vi1[0]~vi1[4]
    var vi2: real[5] = {2, -2, 4}; // Missing dimensions assumed 0s
    print( vi1 * vi2 ); // Inner product, output: "20"
    print( "\n" );
    print( vi1 + vi2 ); // Dimension-wise addition, output: "{7, 1, 8, 1, 2}"
    print( "\n" );
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_1d_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_1d_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int main() {
    double vi1[5] = { 5, 3, 4, 1, 2 };
    double vi2[5] = { 2, -2, 4, 0, 0 };
    printf("%g", inner_product_1d_double(5, vi1, vi2));
    printf("\n");
    print_1d_double(5, add_arrays_1d_double(5, vi1, vi2), 0);
    printf("\n");
}
```

Output

```
jovi@zhangruilindeMacBook-Air:~/Syntax-Analyzer ⌥⌘1
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ± make test TEST_INPUT=tes
t.qv
./syntax_analyzer < test.qv
Symbol Table Stack:
--- Symbol Table ---
-------------- Info ---------------
| Compile Complete.               |
| Generating C code...            |
| Please checkout output.c file. |
-----------------------------------
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main make testC
gcc -o checkResult checkResult.c
./checkResult
20
{ 7, 1, 8, 1, 2 }
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main
```

sample7.qv

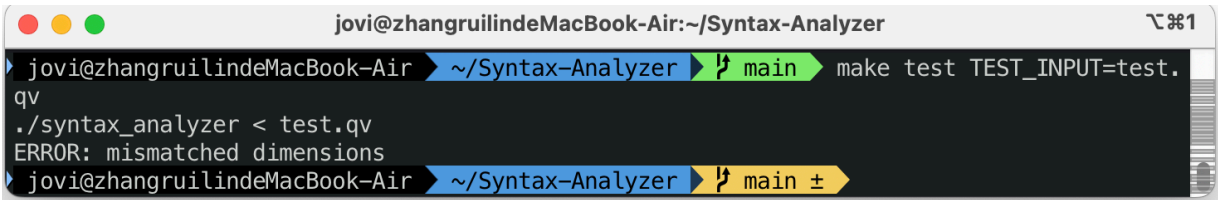
Vector declaration and failed arithmetic **(compulsory)**.

```
fun main () {
    var vi1: real[5] = {5, 3, 4, 1, 2}; // vi1[0]~vi1[4]
    var vi2: real[3] = {2, -2, 4}; // Missing dimensions assumed 0s
    print( vi1 * vi2 ); // yyerror("ERROR: mismatched dimensions")
  print( "\n" );
}
```

C code:

NULL

Output



sample8.qv

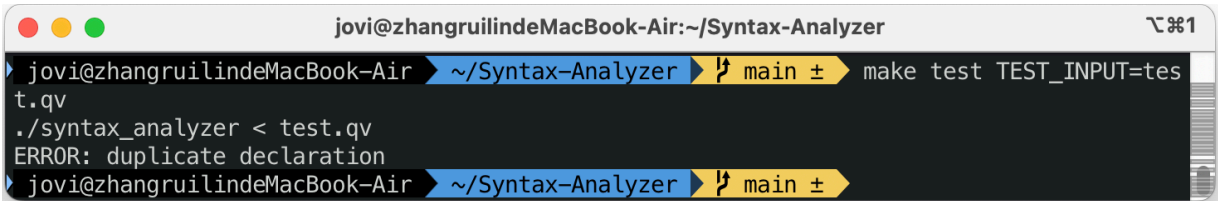
Failed variable declaration 1 **(compulsory)**

```
fun main () {
    var i: int;
    var i: real[3] = {2, -2}; // yyerror("ERROR: duplicate declaration")
}
```

C code:

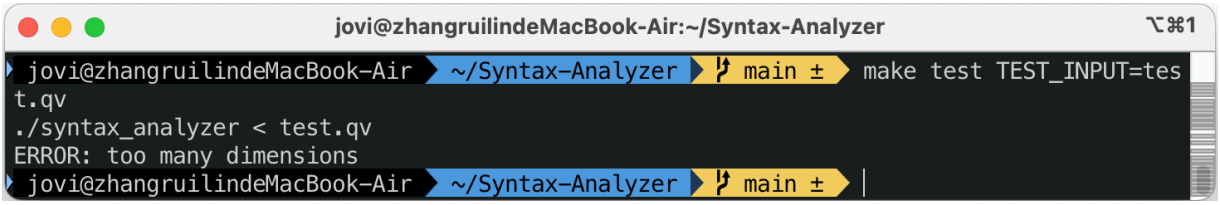
NULL

Output



`sample9.qv` Failed variable declaration 2 **(compulsory)**

`fun main () {`
`    var vi: real[2] = {2, -2, 5}; // yyerror("ERROR: too many dimensions")`
`}`

C code:
NULL
Output


b. 進階測資（老師提供的 optional 測資）

sample10.qv

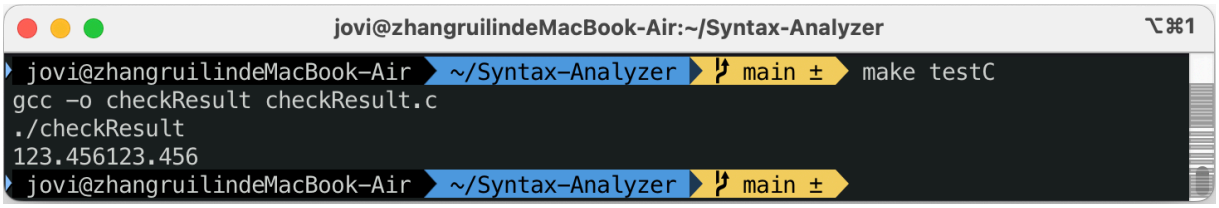
Print with newline (println): **(optional*)**

```
fun main () {
    print(123.456);
    println(123.456);
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_1d_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_1d_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int main() {
    printf("%g", 123.456);
    printf("%g\n", 123.456);
}
```

Output



sample11.qv

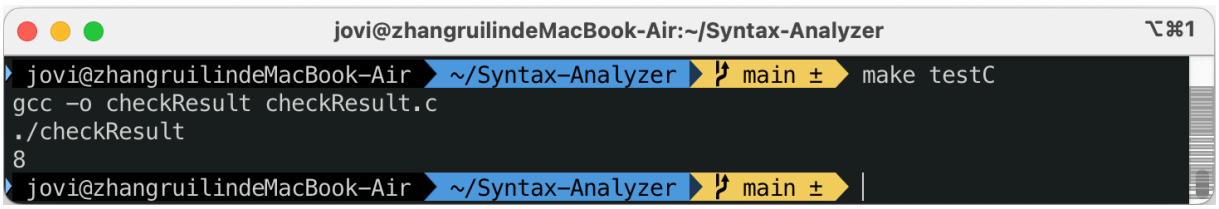
Functions (optional***)

```
fun add(i: int, j: int): int { // Add i and k, then return the result.
  ret i + j;
}

fun main() { // For simplicity, main() always returns 0
  print(add(3, 5));
  print("\n");
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_1d_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_1d_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int add(int i, int j) {
    return i + j;
}
int main() {
    printf("%d", add( 3, 5 ));
    printf("\n");
}
```

Output



sample12.qv

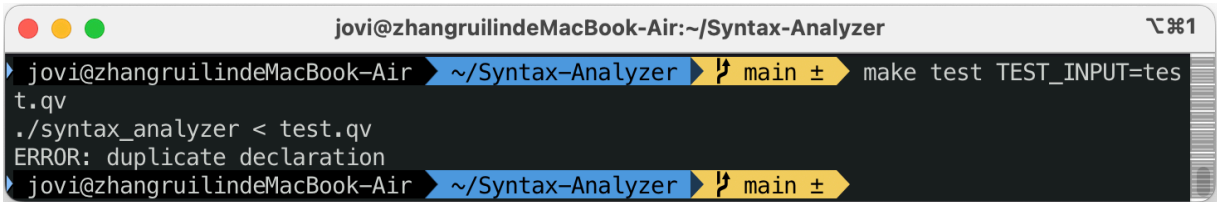
Name scoping **(optional**)**

```
fun main () {
    var i: real = 3.0;
    {
      // Allowed with name hiding (in the parenthesis only the inner i
      //   can be seen and accessed)
        var i: int = 2;
    }
    var i: real[2]; // yyerror("ERROR: duplicate declaration");
}
```

C code:

NULL

Output



c. 額外測資：

otherTestData1.qv

More assignment test data

測試是否可以宣告一個變數，並且在後續的程式碼中，賦值給該變數。

```
fun main(){
    var i: int = 10;
    var j: real = 15;
    var k: int;
    k = i + j;
    println(k);
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_1d_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_1d_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int main() {
    int i = 10;
    double j = 15;
    int k;
    k = i + j;
    printf("%d\n", k);
}
```

Output

```
jovi@zhangruilindeMacBook-Air:~/Syntax-Analyzer  ⌥⌘1
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ± make testC
gcc -o checkResult checkResult.c
./checkResult
25
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ±
```

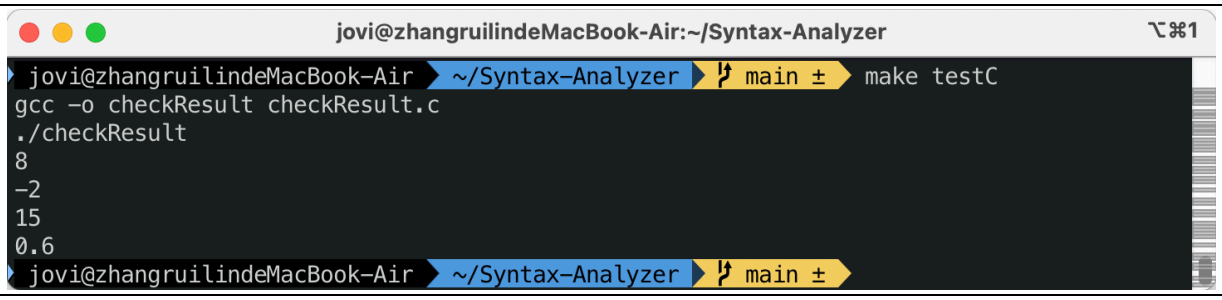
otherTestData2.qv
Function declaration
建立不同的 function，不同回傳型別及參數命名相同，確保在每個 function 中都是獨立的 symbol table，和 function 是否正常。

```
fun myAdd(i: int, j: int): int { // Add i and k, then return the result.
  ret i + j;
}

fun mySub(i: int, j: int): int {
  ret i - j;
}

fun myMul(i: int, j: int): real {
  ret i * j;
}

fun myDiv(i: real, j: real): real {
  ret i / j;
}

fun main() { // For simplicity, main() always returns 0
  println(myAdd(3, 5));
  println(mySub(3, 5));
  println(myMul(3, 5));
  println(myDiv(3, 5));
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_1d_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_1d_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int myAdd(int i, int j) {
    return i + j;
}
int mySub(int i, int j) {
    return i - j;
}
double myMul(int i, int j) {
    return i * j;
}
double myDiv(double i, double j) {
    return i / j;
}
int main() {
    printf("%d\n", myAdd( 3, 5 ));
    printf("%d\n", mySub( 3, 5 ));
    printf("%g\n", myMul( 3, 5 ));
    printf("%g\n", myDiv( 3, 5 ));
}
```

Output



otherTestData3.qv

Function declaration with identifier error

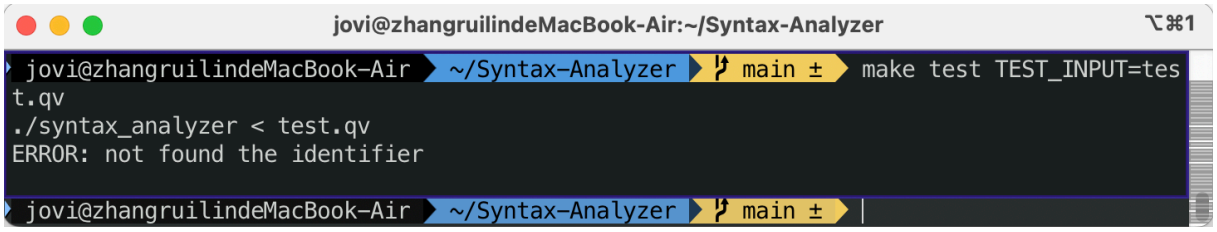
不可使用一個未存在的 Identifier

```
fun myAdd(i: int, j: int): int {
  ret i + j;
}

fun main() {
  println(addition(3, 5)); // ERROR: not found the identifier
}
```

C code:

NULL

Output



otherTestData4.qv

Val declaration and valid operation

使用 val 型別，將所宣告的 variable 作讀取

```
fun main(){
    val i : int = 10;
    println(i); // output: 10
    println(i+1); // output: 11
    println(i-1); // output: 9
    println(i*2); // output: 20
    println(i/2); // output: 5
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_1d_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_1d_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int main() {
    const int i = 10;
    printf("%d\n", i);
    printf("%d\n", i + 1);
    printf("%d\n", i - 1);
    printf("%d\n", i * 2);
    printf("%d\n", i / 2);
}
```

Output

```
jovi@zhangruilindeMacBook-Air:~/Syntax-Analyzer ⌥⌘1
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ± make testC
gcc -o checkResult checkResult.c
./checkResult
10
11
9
20
5
jovi@zhangruilindeMacBook-Air ~/Syntax-Analyzer main ±
```

otherTestData5.qv

Val declaration and invalid operation(assignment)

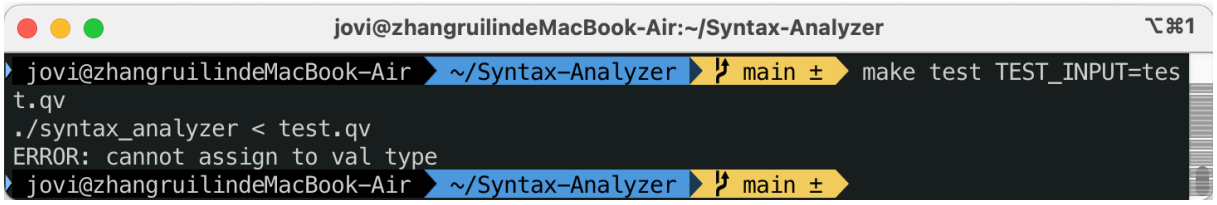
使用 val 型別，將所宣告的 variable 作 assignment 並報錯

```
fun main(){
    val i : int = 10;
    i = 1;  // ERROR: cannot assign to val type
}
```

C code:

NULL

Output



otherTestData6.qv

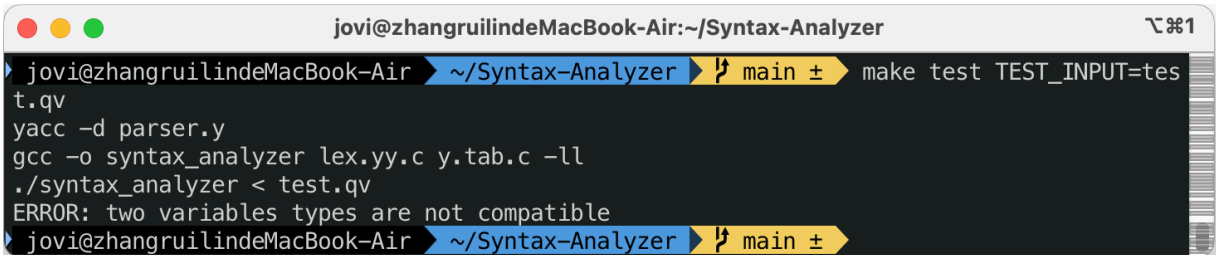
Type mismatch checking

```
fun main(){
    var i : int[5] = {1, 2, 3, 4, 5};
    var j : int = 10;
    println(i+j); // ERROR: two variables types are not compatible
}
```

C code:

NULL

Output



otherTestData7.qv

Name scoping and print its value

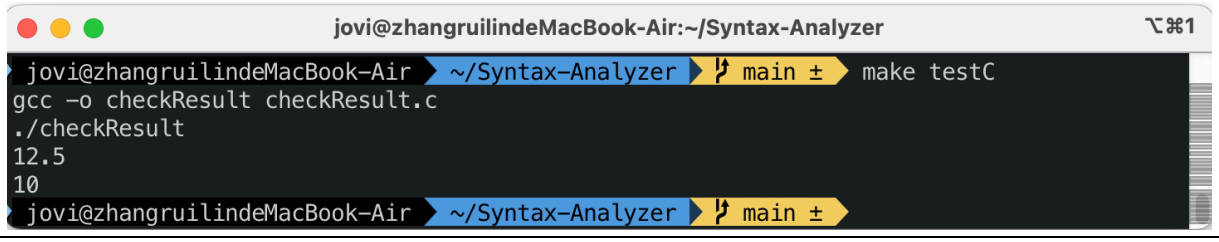
藉由 print(i)或是 println(i)確認不同的 scope 是否有正確取得該 scope 的 symbleTable

```
fun main(){
    var i : int = 10;
    {
        var i : real = 12.5;
        println(i); // output: 12.5
    }
    println(i); // output: 10
}
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
double inner_product_1d_double(int size, double *arr1, double *arr2) { double result = 0.0;for
(int i = 0; i < size; i++) {result += arr1[i] * arr2[i];}return result;}
int inner_product_1d_int(int size, int *arr1, int *arr2) {int result = 0.0;for (int i = 0; i <
size; i++) {result += arr1[i] * arr2[i];}return result;}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {int* result = (int*)malloc(size *
sizeof(int));if (result == NULL) {printf("Memory allocation failed\n");exit(1);}for (int i =
0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return result;}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {double* result =
(double*)malloc(size * sizeof(double));if (result == NULL) {printf("Memory allocation
failed\n");exit(1);}for (int i = 0; i < size; i++) {result[i] = arr1[i] + arr2[i];}return
result;}
void print_1d_int(int size, int* result, bool isNewline){printf("{ ");for (int i=0;i<size;i++)
{if (i == size-1) printf("%d }", result[i]);else printf("%d, ", result[i]);}if (isNewline)
printf("\n");}
void print_1d_double(int size, double* result, bool isNewline){printf("{ ");for (int
i=0;i<size;i++){if (i == size-1) printf("%g }", result[i]);else printf("%g, ", result[i]);}if
(isNewline) printf("\n");}
int main() {
    int i = 10;
    {
    double i = 12.5;
    printf("%g\n", i);
    }
    printf("%d\n", i);
}
```

Output



6. 參考資料

1. iThome 鐵人賽 - [Day8] Yacc - 基本介紹與原理：
   https://ithelp.ithome.com.tw/articles/10315876
2. iThome 鐵人賽 - [Day19] Yacc - 優先級(1) left & right
   https://ithelp.ithome.com.tw/articles/10322362
3. Github: https://github.com/sandunrajitha/Compiler-Design-and-Implementation-using-lex-and-yacc
4. Github: https://github.com/deshsidd/RPAL-Lexical-Analyzer-and-Syntax-Analyzer
5. Github: https://github.com/AnjaneyaTripathi/c-compiler/tree/master/Part%203%20-%20Generating%20the%20Symbol%20Table
6. LexAndYaccTutorial:
   https://cse.iitkgp.ac.in/~bivasm/notes/LexAndYaccTutorial.pdf
7. Youtube: https://www.youtube.com/watch?v=__-wUHG2rfM&t=1s

7. 附件

a. 已宣告函式（格式化）

```
double inner_product_1d_double(int size, double *arr1, double *arr2) {
    double result = 0.0;
    for (int i = 0; i < size; i++) {
        result += arr1[i] * arr2[i];
    }
    return result;
}
int inner_product_1d_int(int size, int *arr1, int *arr2) {
    int result = 0.0;
    for (int i = 0; i < size; i++) {
        result += arr1[i] * arr2[i];
    }
    return result;
}
int* add_arrays_1d_int(int size, int* arr1, int* arr2) {
    int* result = (int*)malloc(size * sizeof(int));
    if (result == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    for (int i = 0; i < size; i++) {
```

```
        result[i] = arr1[i] + arr2[i];
    }
    return result;
}
double* add_arrays_1d_double(int size, double* arr1, double* arr2) {
    double* result = (double*)malloc(size * sizeof(double));
    if (result == NULL) {
        printf("Memory allocation failed\n");exit(1);
    }
    for (int i = 0; i < size; i++) {
        result[i] = arr1[i] + arr2[i];
    }
    return result;
}
void print_1d_int(int size, int* result, bool isNewline){
    printf("{ ");
    for (int i=0;i<size;i++){
        if (i == size-1) printf("%d }", result[i]);
        else printf("%d, ", result[i]);
    }
    if (isNewline) printf("\n");
}
void print_1d_double(int size, double* result, bool isNewline){
    printf("{ ");
    for (int i=0;i<size;i++){
        if (i == size-1) printf("%g }", result[i]);
        else printf("%g, ", result[i]);
    }
    if (isNewline) printf("\n");
}
```