計算機程式設計
C語言 Unit Test

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

單元測試 (Unit Test)

❑ 測試

  ○ 一個程式如何算完成,如何證明程式沒有問題。

  ○ 應有測試程式(test driver)與測試案例(test case)驗證,程式通過這些test case。

  ○ 單元測試用簡單明確方法,驗證某功能在測試案例如預期運作。

❑ 計算BMI公式: BMI = 體重(公斤) / 身高^2(公尺平方)

  ○ 測試案例:52公斤的人,身高155公分,BMI為 : 52(公斤)/(1.55*1.55)(公尺平方)= 21.64412。兩位小數 21.64

```
double computeBMI(int kg, int height) {
    double M = height/100.0;
    double BMI = 0.0;
    if (kg<=0 || height<=0)
        return -1;
    BMI = round(kg/(M*M),2) ; //四捨五入取兩位小數
    return BMI;
}
```

單元測試 (Unit Test)

❑ #include <assert.h>，assert(int expression)，錯誤會中斷程式。

```c
#include <stdio.h>              // main.c
#include <math.h>
#include <assert.h>
double computeBMI(int kg, int height) {
    double BMI = 0.0, M = height/100.0;
    if (kg<=0 || height<=0)
        return -1;
    BMI = round(100*kg/(M*M))/100;  //四捨五入取兩位小數
    return BMI;
}
int main() {
    int kg = 52, height = 155;
    double expectedResult = 21.64f;
    double result = computeBMI(kg, height);
    assert(fabs(result-expectedResult)<0.0001);
    printf("Hi\n");
    return 0;
}
```

程式碼涵蓋度 (Code Coverage)

❑ 程式碼涵蓋度 (Code Coverage) – 指令涵蓋與分支涵蓋
  ○ 每條指令是否執行過? 每個分支判斷 True/False是否執行過?
❑ gcov (GCC Coverage)，測試程式碼覆蓋率的工具
  ○ 分析哪幾行程式被執行過，統計每一行程式的執行次數
❑ gprof 程式分析工具(profiling tool)，分析程式耗費時間
 ○ 藉以改善程式執行效率

程式碼涵蓋度 (Code Coverage)

- 指令行執行步驟
  - 編輯main.c
  - 在指令行執行gcc -fprofile-arcs -ftest-coverage -o main main.c
    - 在目標檔中加入gcov所需extra profiling information
    - 產生main.exe、main.gcno(gcov 所需檔案)
  - 在指令行執行main.exe
    - 產生test.gcda檔案(gcov 所需data檔案)
  - 在指令行執行gcov -b -c main.c
    - 顯示code coverage資訊
    - 產生main.c.gcda，紀錄每行程式碼執行的次數
  - 使用記事本打開main.c.gcov，查看每行程式碼執行的次數

程式碼涵蓋度 (Code Coverage)

❑ 指令行執行步驟

  ○ 在e:\Test目錄，編輯main.c
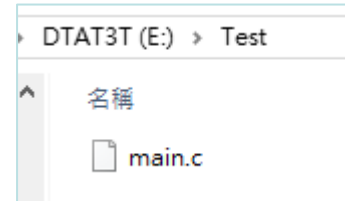
```
main.c - 記事本
檔案(F)  編輯(E)  格式(O)  檢視(V)  說明(H)
#include <stdio.h>
#include <math.h>
#include <assert.h>
double computeBMI(int kg, int height) {
    double M = height/100.0;
    double BMI = 0.0;
    if (kg<=0 || height<=0)
        return -1;
    BMI = round(100*kg/(M*M))/100;   //四捨五入取兩位小數
    return BMI;
}
int main() {
    int kg = 52, height = 155;
    double expectedResult = 21.64f;
    double result = computeBMI(kg, height);
    assert(fabs(result-expectedResult)<0.0001);
    printf("Hi\n");
    return 0;
}
```

程式碼涵蓋度 (Code Coverage)

○ 在指令行執行gcc -fprofile-arcs -ftest-coverage -o main main.c
  ➤ 開啟命令提示字元
  ➤ e:
  ➤ cd e:\Test
  ➤ dir main.c
  ➤ path=%path%;C:\Program Files (x86)\CodeBlocks\MinGW\bin
    – C:\Program Files (x86)\CodeBlocks\MinGW\bin 安裝codeblock的
      地方，或gcc的地方
  ➤ gcc -fprofile-arcs -ftest-coverage -o main main.c
  ➤ dir
    – 確認產生main.exe、main.gcno

程式碼涵蓋度 (Code Coverage)

○ 在指令行執行gcc -fprofile-arcs -ftest-coverage -o main main.c

程式碼涵蓋度 (Code Coverage)

- 在指令行執行main.exe
  - 產生main.gcda檔案(gcov 所需data檔案)

```
E:\Test>main.exe
Hi

E:\Test>dir
 磁碟區 E 中的磁碟是 DTAT3T
 磁碟區序號:  B613-6B32

 E:\Test 的目錄

2020/02/16  上午 11:27    <DIR>          .
2020/02/16  上午 11:27    <DIR>          ..
2020/02/16  上午 10:42                499 main.c
2020/02/16  上午 11:22            132,620 main.exe
2020/02/16  上午 11:27                244 main.gcda
2020/02/16  上午 11:22                796 main.gcno
              4 個檔案            134,159 位元組
              2 個目錄  845,378,142,208 位元組可用
```

程式碼涵蓋度 (Code Coverage)

○ 在指令行執行gcov -b -c main.c
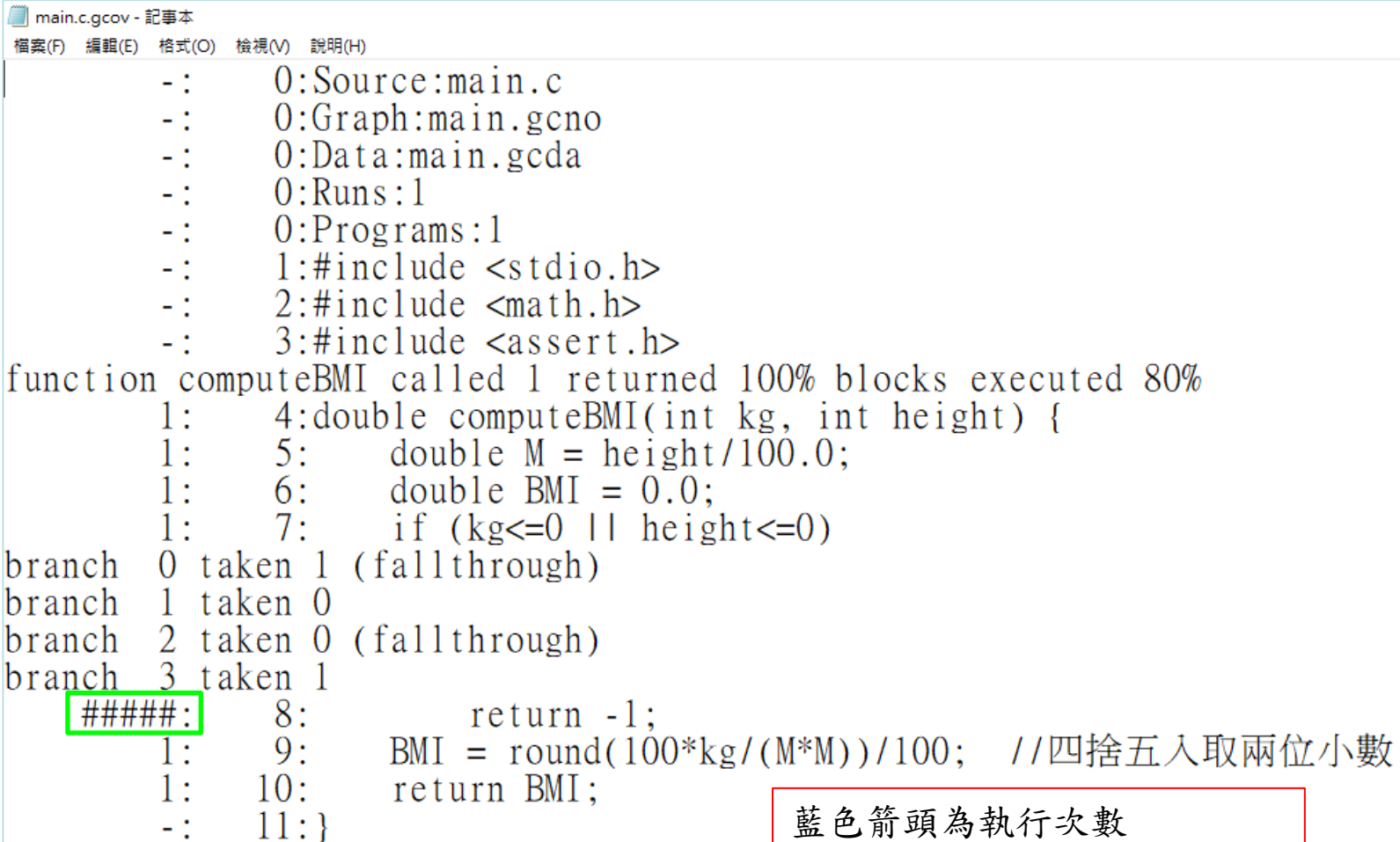
> 顯示code coverage資訊

> 產生main.c.gcov，紀錄每行程式碼執行的次數

```
E:\Test>gcov -b -c main.c
File 'main.c'
Lines executed:92.86% of 14
Branches executed:100.00% of 6
Taken at least once:50.00% of 6
Calls executed:66.67% of 3
Creating 'main.c.gcov'


E:\Test>dir
 磁碟區 E 中的磁碟是 DTAT3T
 磁碟區序號:  B613-6B32

 E:\Test 的目錄

2020/02/16  上午 11:27    <DIR>          .
2020/02/16  上午 11:27    <DIR>          ..
2020/02/16  上午 10:42               499 main.c
2020/02/16  上午 11:27             1,315 main.c.gcov
2020/02/16  上午 11:22           132,620 main.exe
2020/02/16  上午 11:27               244 main.gcda
2020/02/16  上午 11:22               796 main.gcno
               5 個檔案         135,474 位元組
               2 個目錄  845,378,138,112 位元組可用
```

程式碼涵蓋度 (Code Coverage)

○ 使用記事本打開main.c.gcov，查看每行程式碼執行的次數

```
        -:        0:Source:main.c
        -:        0:Graph:main.gcno
        -:        0:Data:main.gcda
        -:        0:Runs:1
        -:        0:Programs:1
        -:        1:#include <stdio.h>
        -:        2:#include <math.h>
        -:        3:#include <assert.h>
function computeBMI called 1 returned 100% blocks executed 80%
        1:        4:double computeBMI(int kg, int height) {
        1:        5:    double M = height/100.0;
        1:        6:    double BMI = 0.0;
        1:        7:    if (kg<=0 || height<=0)
branch  0 taken 1 (fallthrough)
branch  1 taken 0
branch  2 taken 0 (fallthrough)
branch  3 taken 1
    #####:        8:        return -1;
        1:        9:    BMI = round(100*kg/(M*M))/100;   //四捨五入取兩位小數
        1:       10:    return BMI;
        -:       11:}
```

藍色箭頭為執行次數
紅色箭頭為行數
綠色框框的####表示沒執行到

程式碼涵蓋度 (Code Coverage)

○ 使用記事本打開main.c.gcov，查看每行程式碼執行的次數

```
function main called 1 returned 100% blocks executed 83%
        1:    12:int main() {
        1:    13:      int kg = 52, height = 155;
        1:    14:      double expectedResult = 21.64f;
        1:    15:      double result = computeBMI(kg, height);
call      0 returned 1
        1:    16:      assert(fabs(result-expectedResult)<0.0001);
branch    0 taken 0 (fallthrough)
branch    1 taken 1
call      2 never executed
        1:    17:      printf("Hi\n");
call      0 returned 1
        1:    18:      return 0;
        -:    19:}
        -:    20:
```
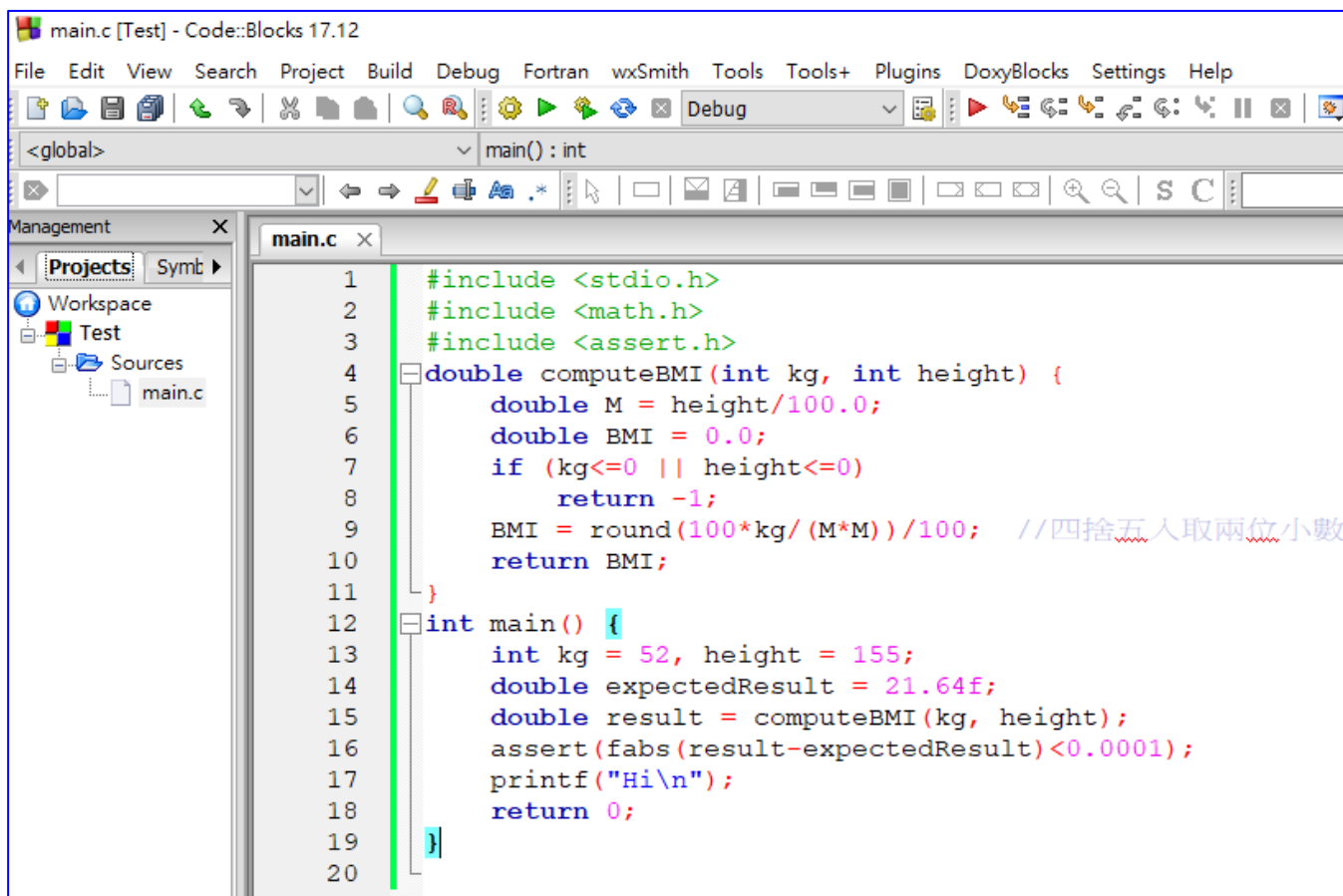
Exercise

❑ 如何使得涵蓋度 100%

程式碼涵蓋度 (Code Coverage)
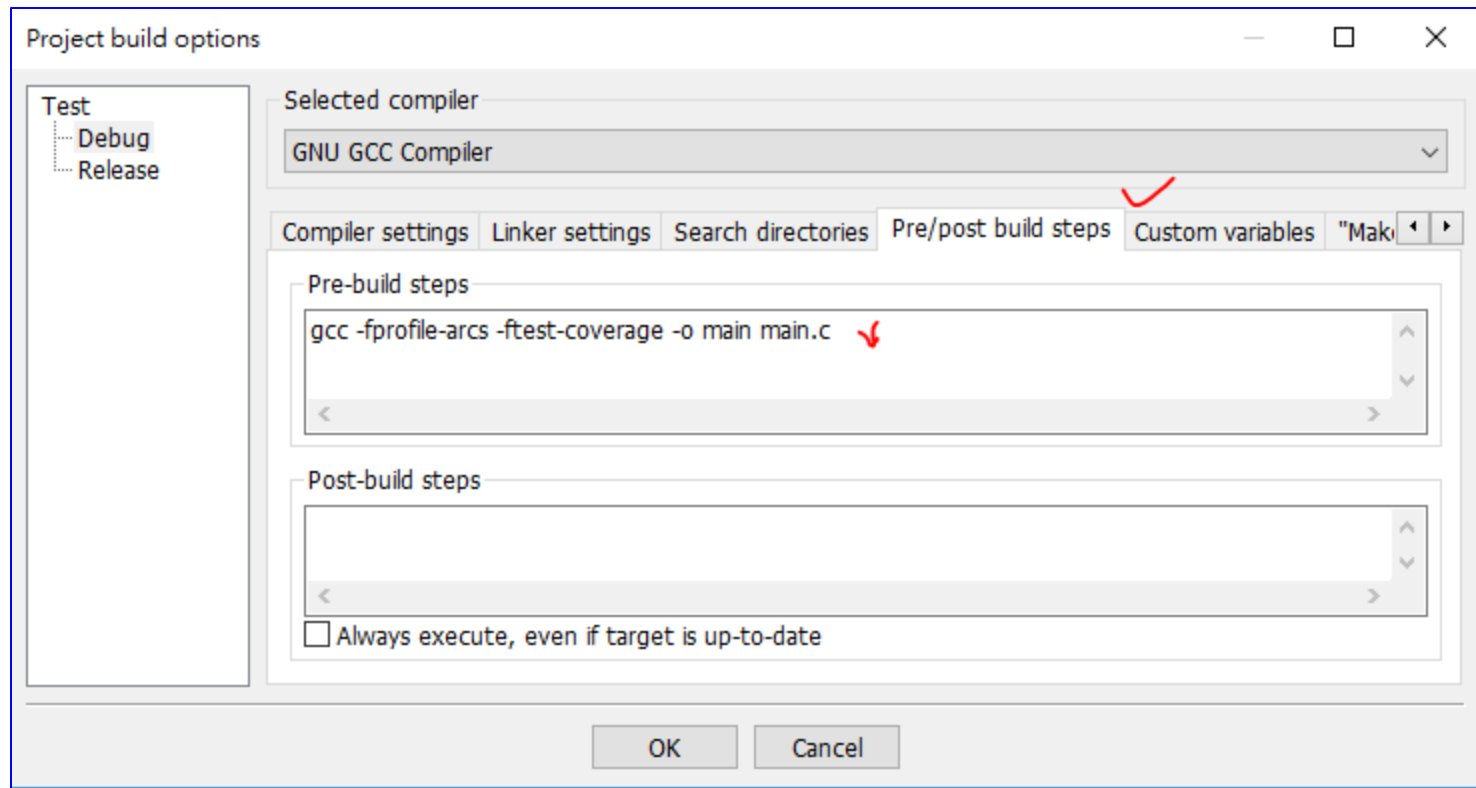
❑ 使用Code:Blocks

　○ 新增一個 Console Application 專案，編輯main.c

程式碼涵蓋度 (Code Coverage)

○ 選單 project – build options – pre/post build steps，輸入

➢ gcc -fprofile-arcs -ftest-coverage -o main main.c

➢ 按 OK

➢ 選單 File – save project

程式碼涵蓋度 (Code Coverage)

○ 選單 Build – Build and run

○ 查看檔案總管，產生main.gcno
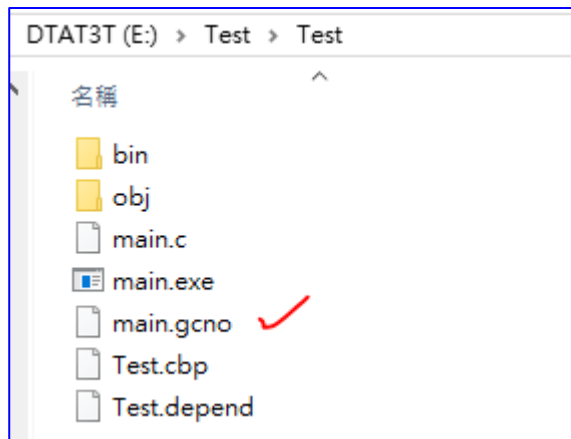
Exercise

❑ 測試計算BMI值的function

○ BMI值計算公式: BMI = 體重(公斤) / 身高^2(公尺平方)

○ 例如：一個52公斤的人，身高是155公分，則BMI為:

○ 52(公斤)/(1.55*1.55)(公尺平方)= 21.64412

○ 正常範圍為 BMI=18.5～24

○ 請設計一個 function，傳入身高(公分)、體重(公斤)，回傳BMI，取兩位小數四捨五入。

○ 當BMI<18.5，輸出-1。

○ 當BMI>24，輸出-2。

○ 當身高或體重<0，輸出0。

Exercise

❑ 測試計算BMI值的function

```c
#include <stdio.h>          // main.c
#include <math.h>
#include <assert.h>
double computeBMI(int kg, int height) {
    double BMI = 0.0, M = height/100.0;
    if (kg<=0 || height<=0)
        return 0;
    BMI = round(100*kg/(M*M))/100;  //四捨五入取兩位小數
    if (BMI <18.5)
        return -1;
    if (BMI >24)
        return -2;
    return BMI;
}
```

```c
int main() {
    int kg = 52, height = 155;
    double expectedResult = 21.64f;
    double result = computeBMI(kg, height);
    assert(fabs(result-expectedResult)<0.0001);
    printf("Hi\n");
    return 0;
}
```

Exercise

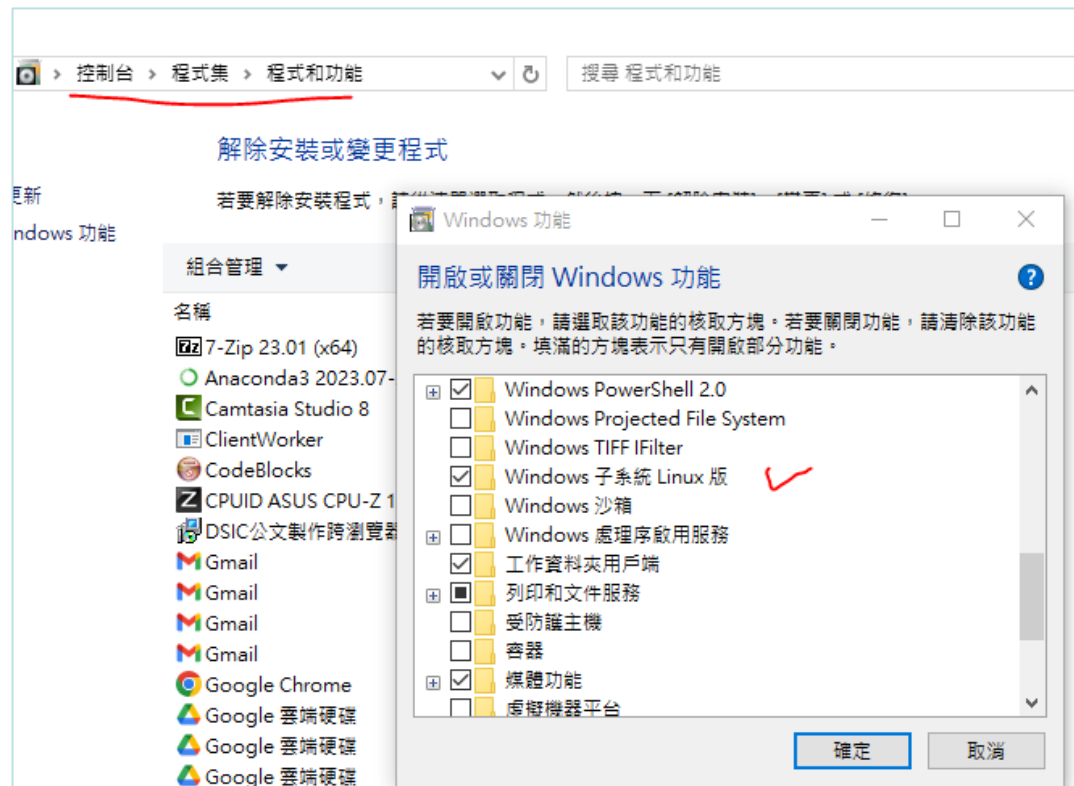❑ 給予一組學生名單，包括名字、學號以及其三科成績，計算每位學生的平均分數，並將最高分與最低分的學生姓名分數印出。

Google Test

❑ Windows 10 安裝 Ubuntu bash

　○ 控制台 -> 應用程式與功能 -> 程式和功能 -> 開啟或關閉 Windows 功能

　○ 勾選『適用於 Linux 的 Windows 子系統 (立刻重新啟動)

Google Test

❑ 搜尋 Microsoft Store 安裝 ubuntu ，安裝完 成後按「啟動」
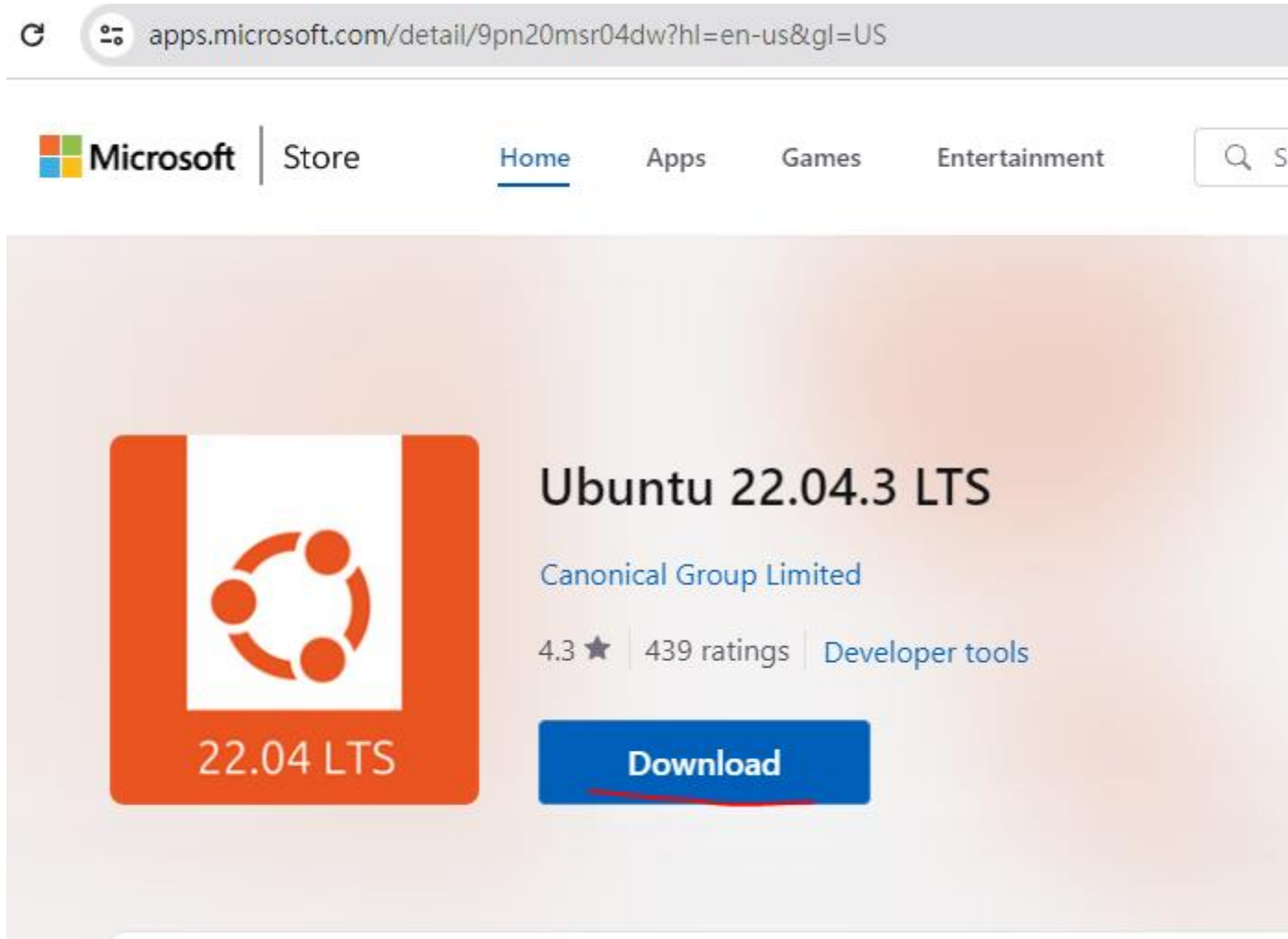
Google Test

❑ 若Windowd 10是LTSC版,無法在MicroSoft Store安裝
  ○ 手動下載 Windows 子系統 Linux 版散發版本套件
  ○ https://docs.microsoft.com/zh-tw/windows/wsl/install-manual

  ## 下載發行版本

  如果無法使用 Microsoft Store 應用程式,您可以按一下下列連結,下載並手動安裝 Linux 散發版本:

  * Ubuntu 20.04
  * Ubuntu 20.04 ARM
  * Ubuntu 18.04

  ○ 使用 PowerShell安裝散發版本
  ○ Add-AppxPackage .\Ubuntu_1804.2019.522.0_x64.appx

```
PS D:\desk> Add-AppxPackage .\Ubuntu_1804.2019.522.0_x64.appx
PS D:\desk>
```

Google Test

❑ 在 Ubuntu (win10-Ubuntu)設定 Google test

　○ 左下角程式集，開啟 Ubuntu bash

　○ 一開始設定帳號、密碼

Google Test

❑ 在 Ubuntu bash 一行一行 輸入以下指令(需要十幾分鐘)

- sudo apt-get update
- sudo apt-get install g++
- sudo apt-get install make
- sudo apt-get install libgtest-dev
- sudo apt-get install cmake
- cd /usr/src/gtest
- sudo cmake CMakeLists.txt
- sudo make
- cd lib
- sudo cp *.a /usr/lib

Google Test

❑ 在 Windows 10 桌面新增googleTest目錄

❑ 在目錄內編輯 ut.cpp
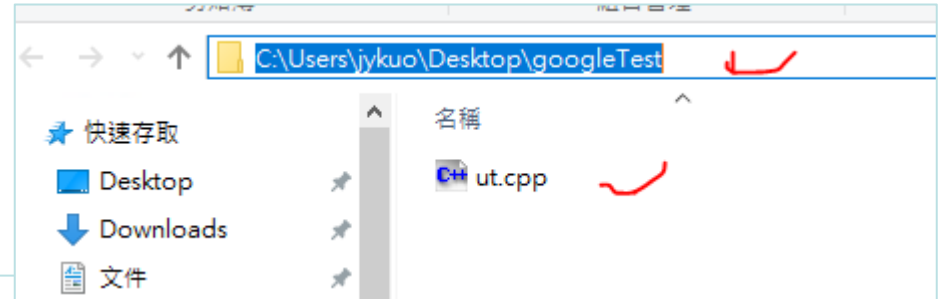

```cpp
#include <cstdlib>
#include <gtest/gtest.h>
int mul( int a , int b ) {
    return a * b ;
}
TEST( multest , HandleNoneZeroInput ) {
    ASSERT_EQ( 21 , mul( 3 , 7 ) ) ;
    ASSERT_EQ( -24 , mul( -6 , 4 ) ) ;
}
int main( int argc , char **argv ){
    testing :: InitGoogleTest( &argc , argv ) ;
    return RUN_ALL_TESTS( ) ;
}
```

Google Test

❑ 在 桌面googleTest目錄，按shift + 滑鼠右鍵
  ○ 選開啟powerShell，在指令行輸入bash，進入Linux Ubuntu。
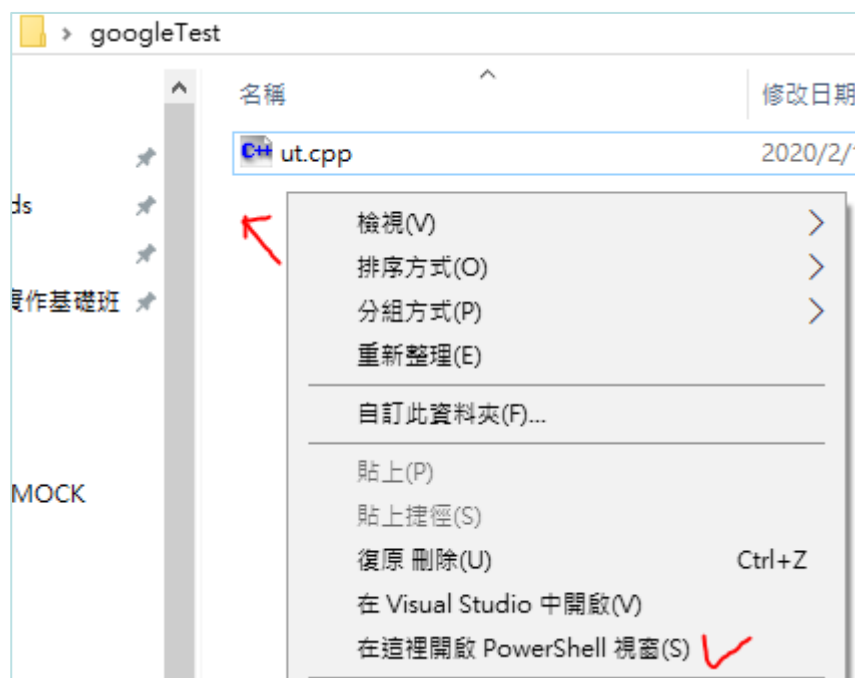
Google Test

❑ 輸入指令，編譯、執行、看code coverage報表

○ g++ -pg -fprofile-arcs -ftest-coverage ut.cpp -o ut -lgtest -lpthread

○ ./ut

```
jykuo@DESKTOP-CE8JI9C: /mnt/c/Users/jykuo/Desktop/googleTest                                    —      □      ×
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ g++ -pg -fprofile-arcs -ftest-coverage ut.cpp -o ut -lgtest
 -lpthread
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ ./ut
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from multest
[ RUN      ] multest.HandleNoneZeroInput
[       OK ] multest.HandleNoneZeroInput (0 ms)
[----------] 1 test from multest (1 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (2 ms total)
[  PASSED  ] 1 test.
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$
```

○ gcov -c -b ut.cpp

```
jykuo@DESKTOP-CE8JI9C: /mnt/c/Users/jykuo/Desktop/googleTest
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ gcov -c -b ut.cpp
File 'ut.cpp'
Lines executed:100.00% of 9
Branches executed:57.89% of 38
Taken at least once:28.95% of 38
Calls executed:53.19% of 47
Creating 'ut.cpp.gcov'
```

Google Test - 安裝html報表

❑ 安裝 lcov，sudo apt-get install -y lcov

```
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ sudo apt-get install -y lcov
[sudo] password for jykuo:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

❑ 產生html報表，

○ lcov -c -o ut.info -d . --rc lcov_branch_coverage=1

```
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ lcov -c -o ut.info -d . --rc lcov_branch_coverage=1
Capturing coverage data from .
Found gcov version: 7.4.0
Scanning . for .gcda files ...
Found 1 data files in .
Processing ut.gcda
Finished .info-file creation
```

○ genhtml ut.info -o report --branch-coverage

```
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ genhtml ut.info -o report --branch-coverage
Reading data file ut.info
Found 8 entries.
Found common filename prefix "/usr/include"
Writing .css and .png files.
Generating output.
Processing file /mnt/c/Users/jykuo/Desktop/googleTest/ut.cpp
Processing file c++/7/iostream
```

Google Test - 安裝html報表

❑ 在googleTest\report目錄點選index.html

❑ 產生html報表

Google Test - 分開測試檔案

❑ 編輯三個檔案

```
// bmi.h
double computeBMI(int kg, int height);
```

```
//bmi.cpp
#include <math.h>
double computeBMI(int kg, int height) {
    double BMI = 0.0, M = height/100.0;
    if (kg<=0 || height<=0)
        return 0;
    BMI = round(100*kg/(M*M))/100;  //四捨五入取兩位小數
    if (BMI <18.5)
        return -1;
    if (BMI >24)
        return -2;
    return BMI;
}
```

Google Test - 分開測試檔案

❑ 編輯三個檔案

```cpp
//ut.cpp
//#include <cstdlib>
#include <gtest/gtest.h>
#include "bmi.h"
TEST( multest , HandleNoneZeroInput ) {
    ASSERT_EQ( 0 , computeBMI( 0 ,0 ) ) ;
    ASSERT_EQ( 0 , computeBMI( 100 , 0) ) ;
    ASSERT_EQ( -2 , computeBMI( 52 , 100) ) ;
    ASSERT_EQ( -1 , computeBMI( 42 , 155) ) ;
    ASSERT_EQ( 21.64 , computeBMI( 52 , 155) ) ;
}
int main( int argc , char **argv ){
    testing :: InitGoogleTest( &argc , argv ) ;
    return RUN_ALL_TESTS( ) ;
}
```

Google Test - 分開測試檔案

○ g++ -pg -fprofile-arcs -ftest-coverage bmi.cpp ut.cpp -o ut -lgtest -lpthread

○ ./ut

```
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ g++ -pg -fprofile-arcs
-ftest-coverage bmi.cpp ut.cpp -o ut -lgtest -lpthread
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ ./ut
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from multest
[ RUN      ] multest.HandleNoneZeroInput
[       OK ] multest.HandleNoneZeroInput (0 ms)
[----------] 1 test from multest (5 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (9 ms total)
[  PASSED  ] 1 test.
```

○ gcov -c -b ut.cpp

○ gcov -c -b bmi.cpp

```
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ gcov -c -b bmi.cpp
File 'bmi.cpp'
Lines executed:100.00% of 10
Branches executed:100.00% of 8
Taken at least once:100.00% of 8
No calls
Creating 'bmi.cpp.gcov'
```

Google Test - 分開測試檔案

○ 在googleTest目錄，檢視bmi.cpp.gcov
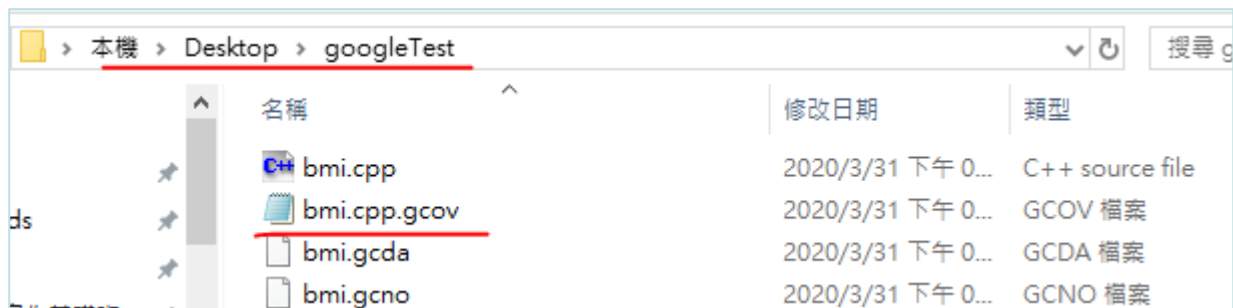

```
bmi.cpp.gcov
 1         -:     0:Source:bmi.cpp
 2         -:     0:Graph:bmi.gcno
 3         -:     0:Data:bmi.gcda
 4         -:     0:Runs:1
 5         -:     0:Programs:1
 6         -:     1:#include <math.h>
 7 function _Z10computeBMIii called 5 returned 100% blocks executed 100%
 8         5:     2:double computeBMI(int kg, int height) {
 9         5:     3:    double BMI = 0.0, M = height/100.0;
10         5:     4:    if (kg<=0 || height<=0)
11 branch  0 taken 4 (fallthrough)
12 branch  1 taken 1
13 branch  2 taken 1 (fallthrough)
14 branch  3 taken 3
15         2:     5:        return 0;
16         3:     6:    BMI = round(100*kg/(M*M))/100;   //四捨五入取兩位小數
17         3:     7:    if (BMI <18.5)
18 branch  0 taken 1 (fallthrough)
19 branch  1 taken 2
20         1:     8:        return -1;
21         2:     9:    if (BMI >24)
22 branch  0 taken 1 (fallthrough)
23 branch  1 taken 1
24         1:    10:        return -2;
25         1:    11:    return BMI;
26         -:    12:}
```

Google Test - 分開測試檔案

❑ 產生html報表

   ○ lcov -c -o ut.info -d . --rc lcov_branch_coverage=1

   ○ genhtml ut.info -o report --branch-coverage

   ○ 在googleTest\report目錄點選index.html

Google Test - 分開測試檔案

❑ 產生html報表

設計測試案例(Test case)方法

❑ 白箱測試(white box testing)
  ○ 檢視程式碼，設計測試案例，執行所有程式碼可能路徑
  ○ 評估測試案例的涵蓋度準則(coverage criteria)
    ➢ Line coverage (statement coverage)，測試案例能執行所有程式碼行
    ➢ branch coverage，測試案例能執行所有條件判斷狀況，一個條件判斷有二種狀況(True, False)，兩個條件組合有四種狀況(True/True, True/False, False/True, False False)

❑ 黑箱測試(black box testing)
  ○ 檢視題目需求，設計測試案例，執行題目需求可能狀況、功能
  ○ 設計測試案例方法
    ➢ 分類功能、規則、輸入條件，例如BMI太高或太低情況
    ➢ 邊界值分析，考慮邊界情況，例如BMI太高或太低剛好邊界情況
    ➢ 負向，不合規範的情況，例如負數體重、身高

黑箱測試方法 Homework

□ 題目需求

　○ 辨識輸入符號，Identifier是C語言的變數

　○ Input: rate R2D2 -2 time 555666 0.23 -1.2 #

　○ Output:

　　➢ rate - Identifier

　　➢ R2D2 - Identifier

　　➢ -22 - Negative Integer

　　➢ 555666 – Positive Integer

　　➢ 0.23 - Positive Floating

　　➢ -1.2 - Negative Floating

　○ Exercise: 考慮負向情況的測試案例

黑箱測試方法 Homework

□ 使用有限狀態機解題

○ **簡化題目**

➤ 只處理identifier, Number

Digit(1)	1, .., 9
Pound	#
Blank	empty
Letter	A,.., Z, a, .., z
Digit(2)	0, 1, .., 9
Underscore	_
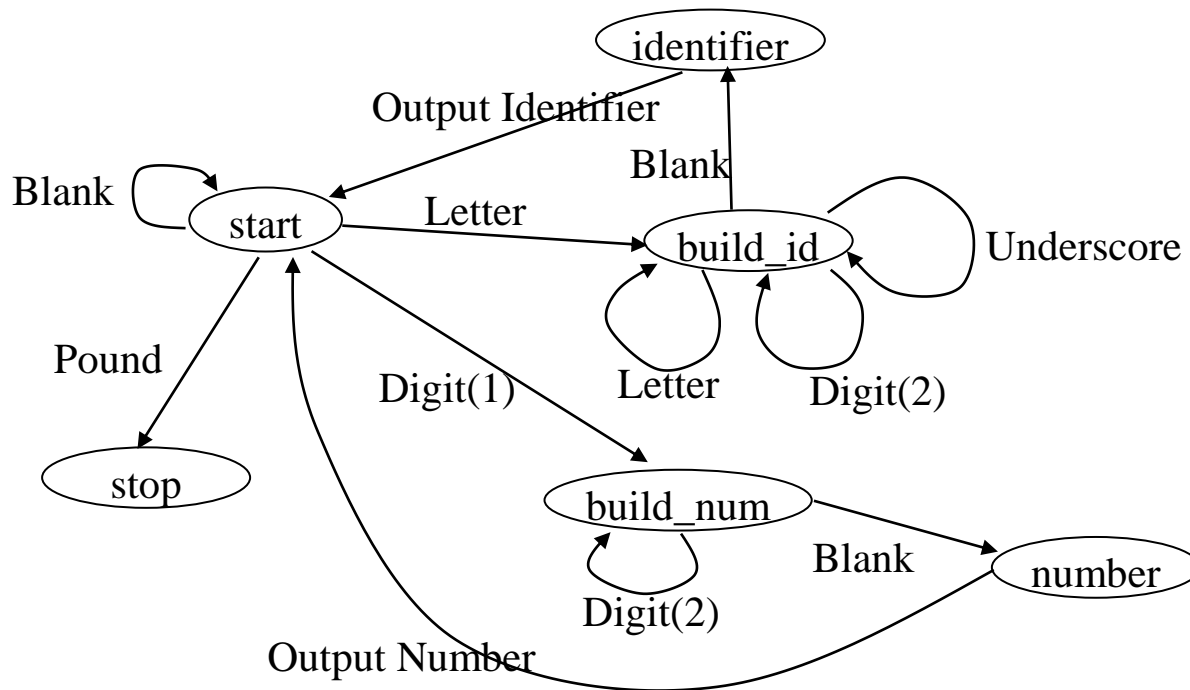
黑箱測試方法 Homework

❑ typedef 自訂資料型別 data type

　○ 自訂資料型別，命名以_t結束，可以一此宣告變數

```
typedef int myInt_t;
myInt x1=0, x2=3;
x1 = x2 + 4;
```

❑ 列舉型別 (enumeration)，使程式易於閱讀、了解、修改

　○ 列舉所宣告的變數，有哪些值，第一個符號其值預設為0，後面依序加1

　○ state_t，是自訂資料型別，值有 start, build_id, build_num，其值分別為 0, 1, 2。

```
typedef enum {start, build_id, build_num} state_t;
state_t x1, x2;
x1 = start;
x2 = build_num;
printf("%d, %d\", x1, x2);  //印出 0, 1
```

黑箱測試方法 Homework

```c
#include <stdio.h>          // FMS.c
#include <ctype.h>
typedef enum {start, build_id, build_num, build_invalid, identifier, number, invalid, stop} state_t;
state_t getNextState(state_t current_state, char ch) {
  if (current_state == start) {
    if (ch == ' ') return start;
    else if (isalpha(ch)) return build_id;
    else if (isdigit(ch)) return build_num;
    else if (ch=='#') return stop;
  }
  if (current_state == build_id) {
    if (isalpha(ch)||isdigit(ch)||(ch=='_')) return build_id;
    else if (ch==' ') return identifier;
  }
  if (current_state == build_num) {
    if (isdigit(ch)) return build_num;
    else if (ch==' ') return number;
  }
  return stop;
}
```

黑箱測試方法 Homework

❑ 使用有限狀態機解題，只處理identifier, Number

```
void FMS() {
  char input_char;
  state_t current_state;
  current_state = start;
  do {
    if (current_state==identifier) {
      printf(" - identifier\n");
      current_state = start;
    }
    else if (current_state==number) {
      printf(" - number\n");
      current_state = start;
    }
    scanf("%c", &input_char);
    if (input_char != ' ') printf("%c", input_char);
    current_state = getNextState(current_state, input_char);
  } while (current_state!=stop);
}
int main() {    FMS();    return 0; }
```

黑箱測試方法 Homework

❑ 設計 test.txt

> rate R2D2 2 time 55566 23 12 #

❑ 編譯 fms.c

❑ 在 cmd 模式，執行 fms.c 測試

```
命令提示字元

D:\Test>fms.exe<test.txt
rate - identifier
R2D2 - identifier
2 - number
time - identifier
55566 - number
23 - number
12 - number
#
D:\Test>
```

APPENDIX Google Test白箱測試

```cpp
#include <stdio.h>                    // fms.cpp
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
typedef enum {start, build_id, build_num, build_invalid, identifier, number, invalid, stop} state_t;
state_t getNextState(state_t current_state, char ch) {
  if (current_state == start) {
    if (ch == ' ')
      return start;
    else if (isalpha(ch))
      return build_id;
    else if (isdigit(ch)) return build_num;
    else if (ch=='#') return stop;
  }
  if (current_state == build_id) {
    if (isalpha(ch)||isdigit(ch)||(ch=='_'))
      return build_id;
    else if (ch==' ')
      return identifier;
  }
  if (current_state == build_num) {
    if (isdigit(ch))
      return build_num;
    else if (ch==' ')
      return number;
  }
  return stop;
}
```

APPENDIX Google Test白箱測試

```cpp
char * FMS(char *s) {                          // fms.cpp
    char out[80];
    char *output = (char*) malloc(sizeof(char)*500);
    char input_char;
    int index=0;
    state_t current_state;
    current_state = start;
    strcpy(output,"");
    do {
       if (current_state==identifier) {
          current_state = start;
          sprintf(out," - identifier\n");
          strcat(output, out);
       }
       else if (current_state==number) {
          current_state = start;
          sprintf(out," - number\n");
          strcat(output, out);
       }
       //scanf("%c", &input_char);
       input_char = s[index++];
       if ((input_char != ' ')&&(input_char != '#')) {
           sprintf(out, "%c", input_char);
           strcat(output, out);
       }
       current_state = getNextState(current_state, input_char);
    } while (current_state!=stop);
    return output;
}
```

44

黑箱測試方法 Homework

❑ 編輯 run.sh

```
#!/bin/bash
g++ -pg -fprofile-arcs -ftest-coverage fms.cpp ut.cpp -o ut -lgtest -lpthread
./ut
gcov -c -b fms.cpp
lcov -c -o ut.info -d . --rc lcov_branch_coverage=1
genhtml ut.info -o report --branch-coverage
```

```
//fms.h
char * FMS(char *s);
```

```
// ut.cpp
#include <cstdlib>
#include <gtest/gtest.h>
#include "fms.h"
TEST( fmsTest , HandleInput ) {
    char s[]="rate R2D2 2 time 55566 23  12 #";
    char expOutput[] = "rate - identifier\nR2D2 - identifier\n2 - number\ntime - identifier\n55566 - number\n23 - number\n12 - number\n";
    ASSERT_STREQ(expOutput, FMS(s));
}
int main( int argc , char **argv ){
    testing :: InitGoogleTest( &argc , argv ) ;
    return RUN_ALL_TESTS( ) ;
}
```

黑箱測試方法 Homework

❑ 編輯 run.sh

　　○ vi run.sh 或 vim run.sh 或gedit run.sh

❑ chmod +x run.sh

❑ ./run.sh

```
jykuo@DESKTOP-CE8JI9C:/mnt/d/Test/googleTest$ ./run.sh
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from fmsTest
[ RUN      ] fmsTest.HandleInput
[       OK ] fmsTest.HandleInput (0 ms)
[----------] 1 test from fmsTest (14 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (381 ms total)
[  PASSED  ] 1 test.
File 'fms.cpp'
Lines executed:95.12% of 41
Branches executed:100.00% of 40
Taken at least once:87.50% of 40
Calls executed:100.00% of 1
```

APPENDIX Google Test白箱測試

LCOV - code coverage report

				Hit	Total	Coverage
Current view: top level - mnt/d/Test/googleTest - fms.cpp (source / functions)			Lines:	39	41	95.1 %
Test: ut.info			Functions:	2	2	100.0 %
Date: 2020-04-02 17:09:09			Branches:	35	40	87.5 %

```
        Branch data      Line data    Source code
    1                 :            : #include <stdio.h>
    2                 :            : #include <stdlib.h>
    3                 :            : #include <ctype.h>
    4                 :            : #include <string.h>
    5                 :            : typedef enum
    6                 :            : {start, build_id, build_num, build_invalid,
    7                 :            : identifier, number, invalid, stop}
    8                 :            : state_t;
    9                 :         31 : state_t getNextState(state_t current_state, char ch) {
   10        [ +  + ]:         31 :     if (current_state == start) {
   11        [ +  + ]:          9 :         if (ch == ' ')
   12                 :          1 :             return start;
   13        [ +  + ]:          8 :         else if (isalpha(ch))
   14                 :          3 :             return build_id;
   15 [ +  + ][ +  + ]:          5 :         else if (isdigit(ch)) return build_num;
   16        [ +  - ]:          1 :         else if (ch=='#') return stop;
   17                 :            :     }
   18        [ +  + ]:         22 :     if (current_state == build_id) {
   19 [ +  + ][ +  + ]:         12 :         if (isalpha(ch)||isdigit(ch)||(ch=='_'))
        [ -  + ]
   20                 :          9 :             return build_id;
   21        [ +  - ]:          3 :         else if (ch==' ')
   22                 :          3 :             return identifier;
   23                 :            :     }
   24        [ +  - ]:         10 :     if (current_state == build_num) {
   25 [ +  + ][ +  + ]:         10 :         if (isdigit(ch))
   26                 :          6 :             return build_num;
   27        [ +  - ]:          4 :         else if (ch==' ')
   28                 :          4 :             return number;
   29                 :            :     }
   30                 :          0 :     else return stop;
   31                 :          0 : }
   32                 :          1 : char * FMS(char *s) {
   33                 :            :     char out[80];
   34                 :          1 :     char *output = (char*) malloc(sizeof(char)*500);
   35                 :            :     char input_char;
   36                 :          1 :     int index=0;
   37                 :            :     state_t current_state;
   38                 :          1 :     current_state = start;
   39                 :          1 :     strcpy(output,"");
   40                 :         30 :     do {
   41        [ +  + ]:         31 :         if (current_state==identifier) {
   42                 :          3 :             current_state = start;
   43                 :          3 :             sprintf(out," - identifier\n");
   44                 :          3 :             strcat(output, out);
   45                 :            :         }
   46        [ +  + ]:         28 :         else if (current_state==number) {
   47                 :          4 :             current_state = start;
   48                 :          4 :             sprintf(out," - number\n");
   49                 :          4 :             strcat(output, out);
   50                 :            :         }
```

Exercise

```
int main() {
    int kg = 52, height = 155;
    double expectedResult = 21.64f;
    double result = computeBMI(kg, height);
    assert(fabs(result-expectedResult)<0.0001);
    assert(computeBMI(0,0)==0);
    assert(computeBMI(100,0)==0);
    assert(computeBMI(52,100)==-2);
    assert(computeBMI(42,155)==-1);
    printf("Hi\n");
    return 0;
}
```

6rte - Invalid
r_yg - Identifier
t#ee - Invalid