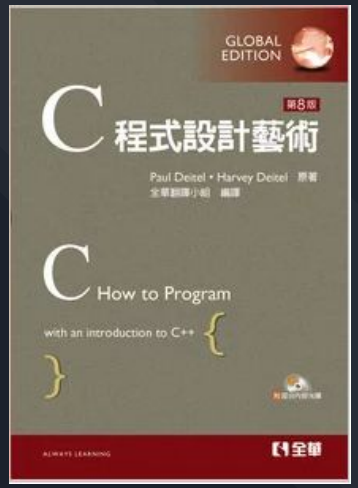
基礎程式設計
C How to Program
Structured Program Development in C



Yung-Chen Chou Ph.D.
iSchool, Feng Chia University
Aug. 15, 2021

演算法 (Algorithm)

- 任何計算的問題均可歸納成以特定的順序來執行一系列的動作
- 解決問題的**程序** (procedure) 如下
  a. 將要執行的**動作** (actions)
  b. 執行這些動作的**順序** (order)
- 這就稱為是**演算法** (algorithm)

虛擬程式碼 (Pseudocode)

- 虛擬程式碼 (Pseudocode) 是一種給人看的非正規的語言，用來幫助發展演算法
- 這裡介紹虛擬程式碼，特別有助於發展那種將轉換成結構化C程式的演算法
- 虛擬程式碼只包含了動作敘述式——就是那些當程式由虛擬碼轉換成C的時候被執行的部分
- 宣告部分並不是可執行的敘述式-他們是給編譯器看的簡易訊息

控制結構 (Control Structures)

- 一般說來，程式中的敘述式是以他們在程式中的順序一個接一個地被執行。這叫做循序式的執行 (sequential execution)
- 有些C敘述式能夠讓你指定下一個執行的敘述式 (非循序式的)。這叫做控制權的移轉 (transfer of control)
- 程式可由三種控制結構 (control structure) 寫成
  - 循序結構 (sequence structure):
    - 循序結構是C內建的特性，除非改變程式執行的流程，否則電腦會自動地按照你所寫的C敘述式的順序，一行一行地執行
  - 選擇結構 (selection structure)
  - 重複結構 (repetition structure)

流程圖 (flowchart)

- 流程圖是整個演算法或是演算法的一部分的圖形表示法
- 流程圖使用具有特殊涵義的標誌來繪製，像是矩形、菱形、圓角矩形以及圓形等等；這些標誌用稱為流向(flowline)的箭頭連接起來

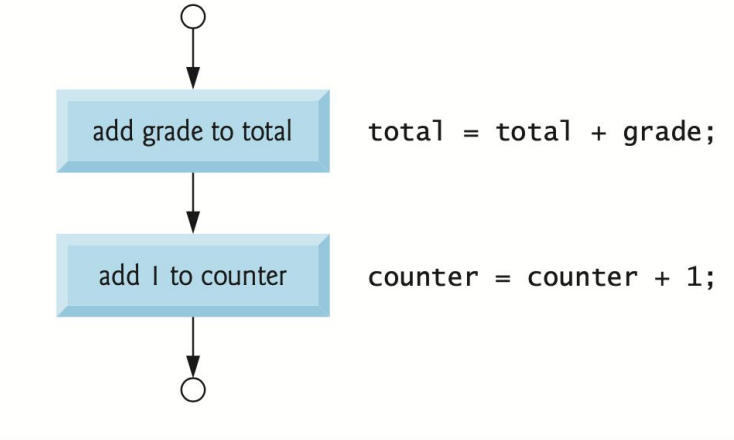


Fig. 3.1 | Flowcharting C's sequence structure.

流程圖 (flowchart)

- 流程圖中最重要的也許是**菱形符號** (diamond symbol)，它也被稱為判斷符號(decision symbol)
- 這種符號表示某項判斷將在此進行
- 選擇結構
  - C語言以敘述式的形式提供了三種選擇結構。if選擇結構在條件式為真時選擇 (執行) 某項動作，而當條件為偽時則跳過這項動作
  - if...else選擇敘述式在條件為真時執行某項動作，而當條件為偽時則執行另一項動作
  - switch選擇敘述式會依運算式的不同，選擇執行許多動作中的一項

重複敘述 (Iteration Statements in C)

- C以敘述式的形式提供了三種重複結構，分別是while、do...while和for
- 以上所介紹的便是C的所有控制結構。C只有七種控制敘述式：循序、三種選擇和三種重複
- 每個C程式都是依據其演算法的需要，組合這七種結構所形成的

if 選擇敘述式

- 選擇敘述可用來選取不同功能的動作
- 例如，假設考試中及格的成績為60分，以下的虛擬碼敘述式

```
If student's grade is greater than or equal to 60
    Print "Passed"
```

- 會判斷條件「student's grade is greater than or equal to 60」是真或偽
- 若為真的話則印出「Passed」，然後繼續「執行」下一個虛擬碼敘述式
- 如果條件式為偽，則列印的動作不會執行，然後執行下一個虛擬碼敘述式

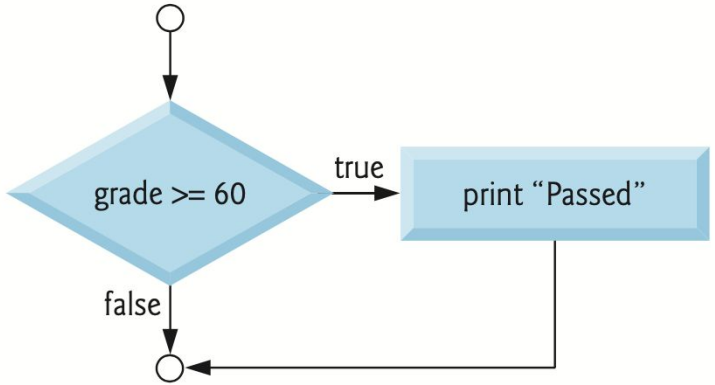


Fig. 3.2 | Flowcharting the single-selection `if` statement.

if...else選擇敘述式

- if...else選擇敘述式則讓你可依據條件的真偽，來執行不同的動作。舉例來說，下列的虛擬碼敘述式

```
If student's grade is greater than or equal to 60
    Print "Passed"
else
    Print "Failed"
```

- 會在學生的成績大於等於60的時候印出Passed，而在小於60的時候印出Failed。在這種情況下，當列印完成後接下來的虛擬碼敘述式都將會被「執行」

```
if ( grade >= 60 ) {
   puts( "Passed" );
} // end if
else {
   puts( "Failed" );
} // end else
```

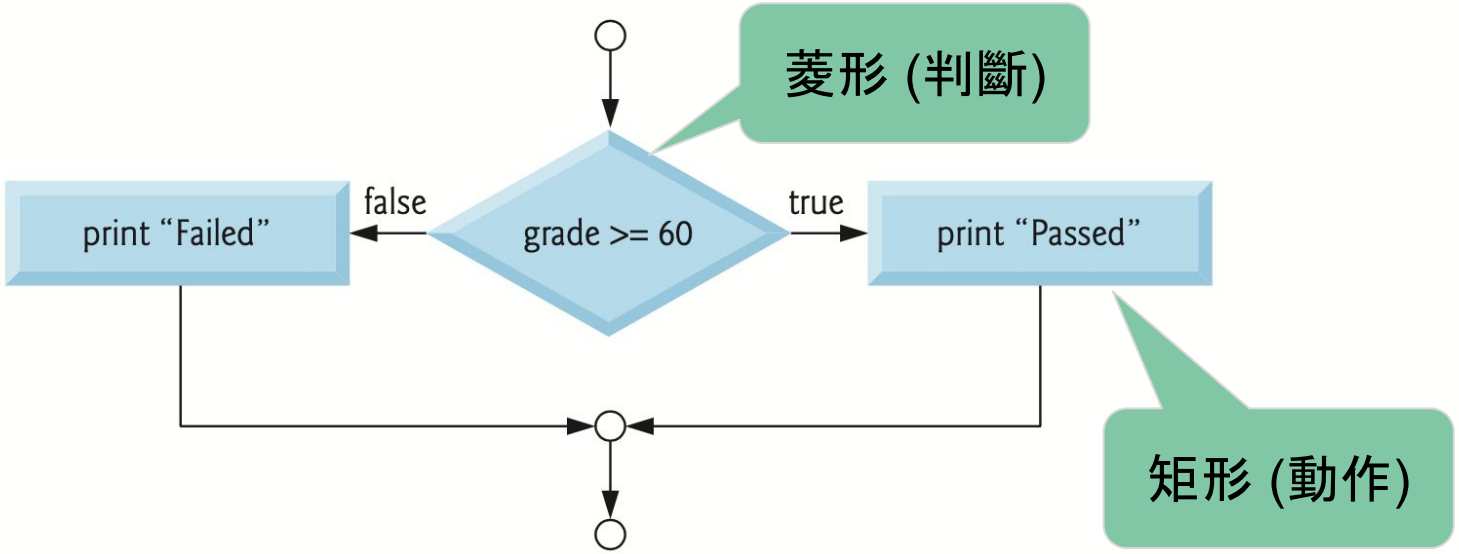


Fig. 3.3 | Flowcharting the double-selection `if...else` statement.

conditional operator (?:)

- closely related to the if...else statement
- The conditional operator is C's only *ternary* operator—it takes three operands.
- These together with the conditional operator form a **conditional expression**.

```
puts( grade >= 60 ? "Passed" : "Failed" );
```

```
grade >= 60 ? puts( "Passed" ) : puts( "Failed" );
```

巢狀的 if...else 敘述式 (Nested if...else Statements)

- 將 if...else 敘述式放到另一個 if...else 敘述式裡，構成巢狀的 if...else 敘述式 (nested if...else statements)，用以檢測多重的狀況
- 舉例來說，下列的虛擬碼敘述式會在考試成績大於等於90的時候印出A，大於等於80（但小於90）時印出B，大於等於70（但小於80）時印出C，大於等於60（但小於70）時印出D，而其他的成績則印出F

```
If student's grade is greater than or equal to 90
    Print "A"
else
    If student's grade is greater than or equal to 80
        Print "B"
    else
        If student's grade is greater than or equal to 70
            Print "C"
        else
            If student's grade is greater than or equal to 60
                Print "D"
            else
                Print "F"
```

```
if ( grade >= 90 ) {
   puts( "A" );
} // end if
else {
   if ( grade >= 80 ) {
      puts( "B" );
   } // end if
   else {
      if ( grade >= 70 ) {
         puts( "C" );
      } // end if
      else {
         if ( grade >= 60 ) {
            puts( "D" );
         } // end if
         else {
            puts( "F" );
         } // end else
      } // end else
   } // end else
} // end else
```

巢狀的if...else敘述式 (Nested if...else Statements)

- 將如果變數grade大於等於90的話，所有四個條件都將為真，但只有第一個測試的puts敘述式會執行
- 在這個puts執行之後，最「外層」之if...else敘述式的else部分將會被跳過
- 可將上述的if敘述式寫成

```
if ( grade >= 90 ) {
   puts( "A" );
} // end if
else if ( grade >= 80 ) {
   puts( "B" );
} // end else if
else if ( grade >= 70 ) {
   puts( "C" );
} // end else if
else if ( grade >= 60 ) {
   puts( "D" );
} // end else if
else {
   puts( "F" );
} // end else
```

while重複敘述式

- 重複敘述式 (repetition statement)，或也稱為循環敘述 (iteration statement)可讓你指定在某種條件持續為真時，重複執行同一項動作。下面這個虛擬碼敘述式

While there are more items on my shopping list
Purchase next item and cross it off my list

- 在購物行程中的重複動作，其中的條件「there are more items on my shopping list」可能是真也可能是偽。如果為真的話，那麼動作「Purchase next item and cross it off my list」就會被執行。而只要此條件持續為真，這項動作就會重複地執行

while重複敘述式

```
product = 3;
while ( product <= 100 ) {
    product = 3 * product;
}
```

- 當條件變成偽時 (當購買了shopping list中的最後一項並將之刪除後)，重複動作便停止，而接下來執行的是重複結構之後的第一個虛擬碼敘述式

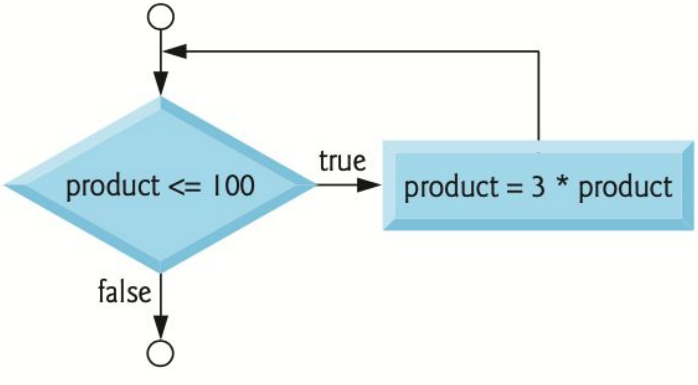


Fig. 3.4 | Flowcharting the `while` iteration statement.

動動腦 1: 求出班上同學的平均成績

- 情境:
  - 10個學生的班級進行一次測驗
  - 已有這次測驗的成績 (分數為 0~100的整數)
- 全班平均即等於所有成績的總和除以學生人數
- 在電腦上解決此問題的演算法必須輸入每個學生的成績，執行求平均值的計算，然後再將結果印出來

A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Determine the class average on the quiz.

- **請用虛擬碼 (pseudocode)，將上述的情境的執行步驟列示下來**

動動腦 1: 求出班上同學的平均成績

- 利用計數器控制的重複結構 (counter-controlled repetition)，一次一個地輸入成績
- 在這項技巧裡，我們用了一個稱為counter (計數器) 的變數，來指定某一組陳述句應被執行的次數
- 在本例中，當counter超過10時，重複動作便告結束

動動腦 1: 求出班上同學的平均成績

```
1   Set total to zero
2   Set grade counter to one
3
4   While grade counter is less than or equal to ten
5       Input the next grade
6       Add the grade into the total
7       Add one to the grade counter
8
9   Set the class average to the total divided by ten
10  Print the class average
```

Fig. 3.5 | Pseudocode algorithm that uses counter-controlled iteration to solve the class-average problem.

C Demo3-1.c > ...

```c
// counter-control iteration example
# include <stdio.h>

int main( void ){
    unsigned int counter;  // number of grades to be entered next
    int grade;     // grade value
    int total;     // sum of grades entered by user
    int average;   // average of grades

    // initialization phase
    total = 0;     // initialize total
    counter = 1;   // initialize loop counter

    // processing phase
    while ( counter <= 10 ){   // loop 10 times
        printf("%s", "Enter grade: "); // prompt for input
        scanf("%d", &grade);    // read grade from user
        total = total + grade; // add grade to total
        counter = counter + 1; // incrment counter
    } // end while
    // termination phase
    average = total / 10; // integer division

    printf("Class average is %d\n", average); // display result
} // end of function main()
```

```
➜  C_Workspace gcc Demo3-1.c -o Demo3-1
➜  C_Workspace ./Demo3-1
Enter grade: 89
Enter grade: 99
Enter grade: 89
Enter grade: 99
Enter grade: 96
Enter grade: 90
Enter grade: 94
Enter grade: 92
Enter grade: 90
Enter grade: 96
Class average is 93
```

動動腦 2: 不同課程修課人數不一的情況

- 情境:
  - 班級修課人數不同測驗，設計一個程式可隨時因應不同人數的成績輸入與平均計算
  - 已有這次測驗的成績 (分數為 0~100的整數)
- 全班平均即等於所有成績的總和除以學生人數
- 在電腦上解決此問題的演算法必須輸入每個學生的成績，執行求平均值的計算，然後再將結果印出來
- 從上而下逐步改進的技術 (top-down, stepwise refinement)，來發展上述的全班平均程式
- **請用虛擬碼 (pseudocode)，將上述的情境的執行步驟列示下來**

```
1   Initialize total to zero
2   Initialize counter to zero
3
4   Input the first grade (possibly the sentinel)
5   While the user has not as yet entered the sentinel
6       Add this grade into the running total
7       Add one to the grade counter
8       Input the next grade (possibly the sentinel)
9
10  If the counter is not equal to zero
11      Set the average to the total divided by the counter
12      Print the average
13  else
14      Print "No grades were entered"
```

Fig. 3.7 | Pseudocode algorithm that uses sentinel-controlled iteration to solve the class-average problem.

C Demo3-2.c > ...

```c
// Class-average program with sentinel-controlled iteration.
# include <stdio.h>

int main( void ){ // function main begins program execution
    unsigned int counter;  // number of grades to be entered next
    int grade;      // grade value
    int total;      // sum of grades entered by user
    float average; // number with decimal point for average
    total = 0;     // initialize total
    counter = 1;   // initialize loop counter
    printf("%s", "Enter grade, -1 to end: "); // prompt for input
    scanf("%d", &grade);    // read grade from user
    while ( grade != -1 ){
        total = total + grade; // add grade to total
        counter = counter + 1; // incrment counter
        printf("%s", "Enter grade, -1 to end: "); // prompt for input
        scanf("%d", &grade);    // read grade from user
    } // end while
    if( counter != 0){
        // calculate average of all grades entered
        average = ( float ) total / counter; // avoid truncation

        // display average with two digits of precision
        printf( "Class average is %.2f\n", average );
    } else { // if no grades were entered, output message
        puts( "No grades were entered" );
    }// end else
} // end of function main()
```

Initialization phase

```
➜  C_Workspace gcc Demo3-2.c -o Demo3-2
➜  C_Workspace ./Demo3-2
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 73.33
```

Get first grade from user

Loop while sentinel value not yet read from user

If user enter at least one grade

含有 (**float**) 這個強制型別轉換運算子, 它會為它的運算元**total**產生一個暫時的浮點數拷貝。而存放在**total**的值仍然是個整數。以這種方式來使用強制型別轉換運算子稱為**明確地轉換 (explicit conversion)**

C Demo3-2.c > ...

```c
// Class-average program with sentinel-controlled iteration.
# include <stdio.h>

int main( void ){ // function main begins program execution
    unsigned int counter;  // number of grades to be entered next
    int grade;      // grade value
    int total;      // sum of grades entered by user
    float average; // number with decimal point for average
    total = 0;     // initialize total
    counter = 1;  // initialize loop counter
    printf("%s", "Enter grade, -1 to end: "); // prompt for input
    scanf("%d", &grade);    // read grade from user
    while ( grade != -1 ){
        total = total + grade; // add grade to total
        counter = counter + 1; // incrment counter
        printf("%s", "Enter grade, -1 to end: "); // prompt for i
        scanf("%d", &grade);    // read grade from user
    } // end while
    if( counter != 0){
        // calculate average of all grades entered
        average = ( float ) total / counter;  // avoid tr cation

        // display average with two digits of precision
        printf( "Class average is %.2f\n", average );
    } else { // if no grades were entered, output message
        puts( "No grades were entered" );
    }// end else
} // end of function main()
```

- **格式化浮點數**
- 使用**printf**的轉換指定詞 **%.2f** 來印出**average**的值
- 其中**f**表示將會有個浮點數要被列印，而 **2**則指定了此值被列印時的**精準度(precision)**
- 如果用的是 **%f**這個轉換指定詞(沒有指定精準度)，那麼列印時將使用**預設精準度 (default precision)** 6－－就如同我們使用 **% 6f**這樣的轉換指定詞

當列印浮點數時，其值會**被四捨五入 (rounded)** 成所指定的精準度。在記憶體內的值未被改變

浮點數注意事項

- 浮點數並不一定「**100%的精確**」，但他們還是可以應用在許多地方
- 例如當我們說「正常」的體溫為98.6度 (華氏) 時，我們並不計較精準到何種程度
- 當我們從體溫計上讀出98.6度時，真正的值可能是98.5999473210643。但將此值看成98.6卻是較實用的
- 相除也會產生浮點數。當我們用10除以3時，得到的是一個無窮小數3.3333333...。電腦只配置了固定大小的空間來存放這種數值，因此電腦所存的值只是個近似值

動動腦 3: 巢狀控制敘述

- 某學校開授一門課程，專門教授有關房地產經紀人執照考試的知識。去年有數位學生修完這門課程，並參加了執照考試。很自然的，這所學校會想要知道她的學生們在這次考試的表現如何。假設你被授意寫個程式來統計考試結果。你手上有這10位學生的名單，以及他們的考試結果 (1 代表過關，2代表失敗)
- 要求
  a. 輸入每個考試結果 (即1或2)。在程式每次要求輸入下一個考試結果時，提示「**Enter result**」這個訊息
  b. 計算每一種考試結果的個數
  c. 列出考試結果的統計，告知有多少學生過關，有多少學生失敗
  d. 如果超過8位學生通過這項考試的話，印出「**Raise tuition**」這個訊息
- **請用虛擬碼 (pseudocode)，將上述的情境的執行步驟列示下來**

```
1   Initialize passes to zero
2   Initialize failures to zero
3   Initialize student to one
4
5   While student counter is less than or equal to ten
6           Input the next exam result
7
8           If the student passed
9               Add one to passes
10          else
11              Add one to failures
12
13          Add one to student counter
14
15  Print the number of passes
16  Print the number of failures
17  If more than eight students passed
18      Print "Bonus to instructor!"
```

Fig. 3.9 | Pseudocode for examination-results problem.

觀察報告

1. 此程式必須處理10個考試結果。可用計數器控制式的迴圈
2. 每一個考試結果是一個數字：不是 1 便是 2。每次程式讀進考試成績時，必須判斷此數為1或2。在我們的演算法裡是判斷它是否為1，若不是1的話，則假設它是2。
3. 用到兩個計數器：
   一個計算過關的學生人數，一個計算失敗的學生人數。
4. 在程式處理完所有的考試結果後，它必須判斷是否有8位以上的學生通過這項考試

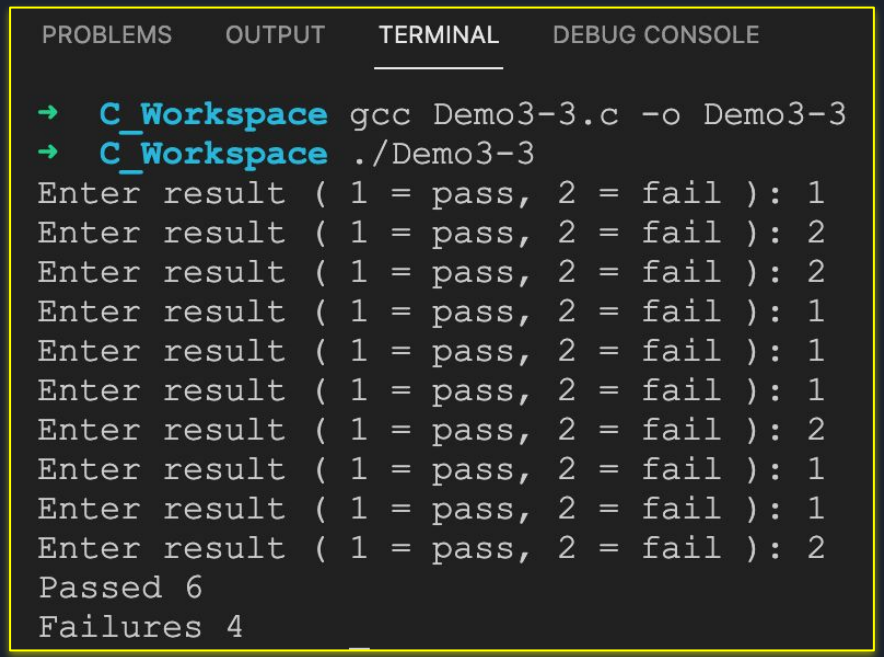
Demo3-3.c > ...

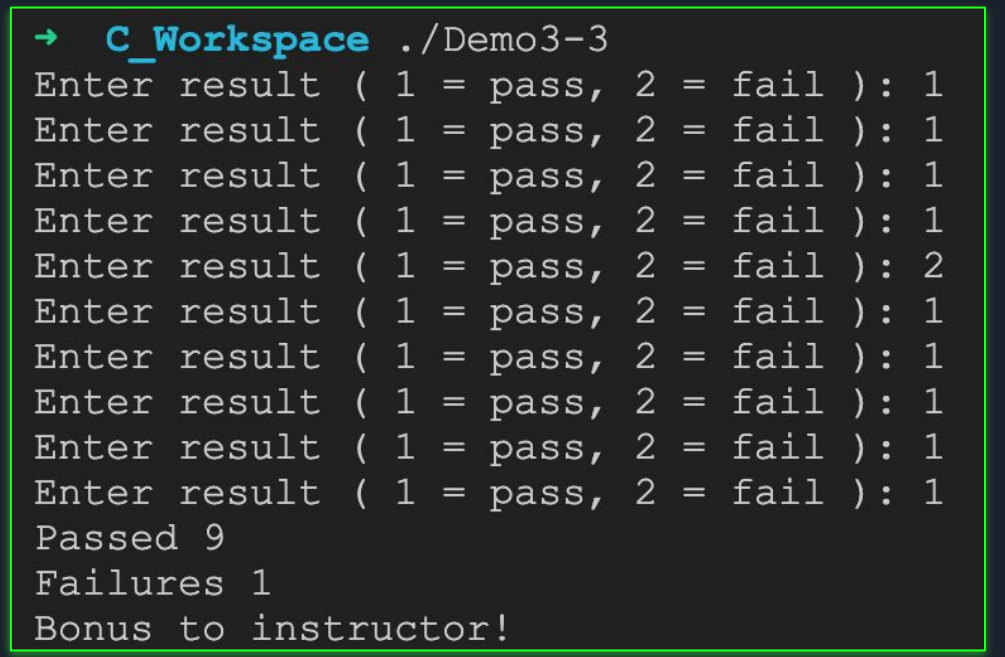
```c
// Analysis of examination results.
#include <stdio.h>

// function main begins program execution
int main( void ){
    unsigned int passes = 0;    // number of passes
    unsigned int failures = 0; // number of failures
    unsigned int student = 1; // student counter
    int result; // one exam result
    while ( student <= 10 ){
        // prompt user for input and obtain value from user
        printf( "%s", "Enter result ( 1 = pass, 2 = fail ): ");
        scanf( "%d", &result );
        if ( result == 1){ // if result 1, increment passes
            passes = passes + 1;
        }else{ // otherwise, increment failures
            failures = failures + 1;
        } // end else
        student = student + 1; // increment student counter
    } // end while
    // termination phase; display number of passes and failures
    printf( "Passed %u\n", passes );
    printf( "Failures %u\n", failures );
    if ( passes > 8 ){
        puts("Bonus to instructor!");
    } // end if
} // end main function
```

```
1  Initialize passes to zero
2  Initialize failures to zero
3  Initialize student to one
4
5  While student counter is less than or equal to ten
6      Input the next exam result
7
8      If the student passed
9          Add one to passes
10     else
11         Add one to failures
12
13     Add one to student counter
14
15 Print the number of passes
16 Print the number of failures
17 If more than eight students passed
18     Print "Bonus to instructor!"
```

Fig. 3.9 | Pseudocode for examination-results problem.


PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
➜ C_Workspace gcc Demo3-3.c -o Demo3-3
➜ C_Workspace ./Demo3-3
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Passed 6
Failures 4


➜ C_Workspace ./Demo3-3
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Passed 9
Failures 1
Bonus to instructor!

指定運算子（Assignment Operators）

- C提供了數種指定運算子，使得指定運算式可以縮寫
- 可利用加法指定運算子 **+=** (addition assignment operator **+=**)
- Ex: **c = c + 3;** ⇒ 縮寫成 **c += 3;**
- **+=** 運算子會把在此運算子右邊的運算式的值，加上此運算子左邊變數的值，然後將結果存到運算子左邊的變數

Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
`+=`	`c += 7`	`c = c + 7`	10 to c
`-=`	`d -= 4`	`d = d - 4`	1 to d
`*=`	`e *= 5`	`e = e * 5`	20 to e
`/=`	`f /= 3`	`f = f / 3`	2 to f
`%=`	`g %= 9`	`g = g % 9`	3 to g

Fig. 3.11 | Arithmetic assignment operators.

遞增和遞減運算子
(Increment and Decrement Operators)

- C還提供了單元遞增運算子++ (increment operator) 和單元遞減運算子-- (**decrement operator**)

Operator	Sample expression	Explanation
++	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

Fig. 3.12 | Increment and decrement operators

C demo20210917.c > main(void)

```c
#include <stdio.h>

int main( void ){
    int c = 5;

    printf("%d\n", c);
    printf("%d\n", c++);
    printf("%d\n\n", c);

    c = 5;
    printf("%d\n", c);
    printf("%d\n", ++c);
    printf("%d\n", c);
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  C_Workspace gcc demo20210917.c -o demo20210917
→  C_Workspace ./demo20210917
5
5
6

5
6
6
```

Operators	Associativity	Type
++ *(postfix)* -- *(postfix)*	right to left	postfix
+ - *(type)* ++ *(prefix)* -- *(prefix)*	right to left	unary
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment

Fig. 3.14 | Precedence and associativity of the operators encountered so far in the text.

安全程式設計 (Secure C Programming)

- 計算兩個整數總和
  - sum = integer1 + integer2;
- 即使這樣的簡單敘述有著潛在的問題-於所儲存的整數變數空間, 兩個整數相加會產生過大的值對。這是所知的算數溢位(arithmetic overflow)並造成不可預知的情況, 且有可能導致系統開放受攻擊
- 最大最小值可以儲存於整數變數以常數INT_MAX和INT_MIN來表示, 定義於標頭檔<limits.h>

C demo20210917.c > ...

```c
#include <stdio.h>
#include <limits.h>

int main( void ){
    printf("max integer type value is %d\n", INT_MAX);
    printf("min integer type value is %d\n", INT_MIN);
    printf("max unsign integer type value is %u\n", UINT_MAX);
    int a = INT_MAX;
    int b = INT_MAX;
    int c = a + a;
    printf("c value is %d\n", c);
    int d = b + b;
    printf("d value is %d\n", d);
    unsigned int e = a + a;
    printf("e value is %u\n", e);
}
```

```
→  C_Workspace gcc demo20210917.c -o demo20210917
→  C_Workspace ./demo20210917
max integer type value is 2147483647
min integer type value is -2147483648
max unsign integer type value is 4294967295
c value is -2
d value is -2
e value is 4294967294
```

無號整數 (Unsigned Integer)

- 無號數宣告在整數型態之前
- 無號類型所表示的範圍從0到一般有號的兩倍範圍
- 在<limits.h>中UINT_MAX確認所使用的平台的最大無號整數值