HW2-1

- 程式碼：

  A. version_1

  分別宣告 integer1, integer2, sum, dif, pro, quo, rem(line4, 10)，並列運算式(line 11~15)

```c
#include<stdio.h>
int main()
{
    int integer1, integer2;
    printf("please enter the first integer:\n");
    scanf("%d", &integer1);
    printf("please enter the second integer:\n");
    scanf("%d", &integer2);

    int sum, dif, pro, quo, rem;
    sum = integer1 + integer2;
    dif = integer1 - integer2;
    pro = integer1 * integer2;
    quo = integer1 / integer2;
    rem = integer1 % integer2;

    printf("first integer plus second one = %d\n", sum);
    printf("subtract second one from first one = %d\n", dif);
    printf("first integer multiply second one = %d\n", pro);
    printf("first integer divided second one = %d\n", quo);
    printf("the remainder when first one divided second one = %d\n",
rem);

    return 0;
}
```

  B. version_2

  宣告 sum, dif, pro, quo, rem 時，直接在後方加上運算式(line

10~14)

```
#include<stdio.h>
int main()
{
    int integer1, integer2;
    printf("please enter the first integer:\n");
    scanf("%d", &integer1);
    printf("please enter the second integer:\n");
    scanf("%d", &integer2);

    int sum = integer1 + integer2;
    int dif = integer1 - integer2;
    int pro = integer1 * integer2;
    int quo = integer1 / integer2;
    int rem = integer1 % integer2;

    printf("first integer plus second one = %d\n", sum);
    printf("subtract second one from first one = %d\n", dif);
    printf("first integer multiply second one = %d\n", pro);
    printf("first integer divided second one = %d\n", quo);
    printf("the remainder when first one divided second one = %d\n",
rem);

    return 0;
}
```

C. version_3

宣告 a 這個變數及其運算式後，利用同一個變數可以重複 assignment 的特性，將所求一一 printf 及以新的運算式覆蓋舊值 (line 10~19)

```
#include<stdio.h>
int main()
{
```

```c
    int integer1, integer2;
    printf("please enter the first integer:\n");
    scanf("%d", &integer1);
    printf("please enter the second integer:\n");
    scanf("%d", &integer2);

    int a = integer1 + integer2;
    printf("first integer plus second one = %d\n", a);
    a = integer1 - integer2;
    printf("subtract second one from first one = %d\n", a);
    a = integer1 * integer2;
    printf("first integer multiply second one = %d\n", a);
    a = integer1 / integer2;
    printf("first integer divided second one = %d\n", a);
    a = integer1 % integer2;
    printf("the remainder when first one divided second one = %d\n", a);

    return 0;
}
```

- 輸出結果：

```
enter the first integer:
77
enter the second integer:
88
first integer plus second one = 165
subtract second one from first one = -11
first integer multiply second one = 6776
first integer divided second one = 77
the remainder when first one divided second one = 0
```

- 延伸問題：

1. **為何不能以 printf（“Sum is sum\n”）; 的形式進行輸出？**

   這樣做可能造成 syntax error。因為 computer perform the actions 是以 string(“ ”)的形式，假如將變數名稱放入” ”中，電腦會視為字串的形式輸出，故在 printf 指令裡，不同型態的資料內容必須配合不

同的列印格式碼，如想印出整數變數的內容則使用%d 資料型態作為格式碼。

。

2. **如果程式改成要求使用者輸入 100 個整數，接著對 100 個整數做四則運算，你的程式碼會怎麼寫？**

因為宣告太多變數會導致 run out of memory，故我們可利用同一個變數可以重複付值(assignment)的特性，避免宣告太多變數及其名稱混亂。

另外因多個變數宣告可能造成程式碼冗長複雜，易讀性低，且變數名稱易搞混，故可利用陣列變數的形式，來省略宣告大量變數的時間，程式碼較為簡潔，也可節省效率。像是這個情況下可以使用 I-dimensional array 的方式宣告數個大量且具同性質的變數，如下圖格式

```
int number[100]; /*宣告整數陣列 number，可存放 100 個元素*/
```

或是

```
#define ARRAYSIZE 100 /*使用 define 將陣列的元素個數設為 100*/
int number[ARRAYSIZE]; /*number 是一個有 100 個元素的整數陣列*/
```

HW2-2

- 程式碼:

```
#include<stdio.h>
int main()
{
    double f, c;
    printf("Please enter a temperature in Fahrenheit:");
    scanf("%lf", &f);

    c = 5.0/9.0 * (f - 32);

    printf("%3.0f Fahrenheit = %5.3f Celsius", f, c);
    return 0;
}
```

- 輸出結果：

```
Please enter a temperature in Fahrenheit:0
  0 Fahrenheit = -17.778 Celsius
```

```
Please enter a temperature in Fahrenheit:32
 32 Fahrenheit = 0.000 Celsius
```

```
Please enter a temperature in Fahrenheit:203
203 Fahrenheit = 95.000 Celsius
```

- 延伸問題：

1. **浮點數為什麼可以比整數多出 4 倍的表示範圍？**

   因為浮點數的資料值較整數來的多(浮點數小數點後也須考慮，但整數無精準度及誤差的問題)，故表示範圍較整數多出 4 倍。

2. **為什麼整數與單精度浮點數的 data size 同樣都是 4 bytes？**

   電腦理解浮點數是以科學記號及有效數字的形式，Complier 在辨識浮點數時會以科學記號的形式來儲存(complier 不會儲存 10)，浮點數是存兩個整數(整數的型別)來完成目的，因為他們所使用的資料型別相同，故所占記憶體相同。

   Ex: 362.48(欲儲存的數字)；$3.62*10^2$(有效數字及科學記號形式) → 電腦會儲存 362 和 2 這兩個整數