由片語學習C程式設計

台灣大學資訊工程系劉邦鋒著

台灣大學劉邦鋒老師講授

August 19, 2016

第三單元

運算

指定

片語 1: 整數變數指定為 0

```
1  i = 0;
```

- 等號 = 是**指定** (assignment) 的動作。
- = 右邊的 0 指定給 = 左邊的變數 i 當成 i 的新值。
- 和初始化不同，指定可以出現很多次，每次指定不同的值。
- 變數 i 在指定之前的舊值從此消失。

要點

特殊字元

= 代表將 = 右邊的值指定給 = 左邊的變數當成新值。

要點

學習要點

在 C 程式語言 中 = 代表指定，而非相等。

要點

風格要點

通常習慣在 = 左右各加一個空白，這樣可以清楚顯示這是一個指定的動作。

設定變數

片語 2: 將變數設定為另一變數

```
1  i = j;
```

- 將 = 右邊變數 j 的值指定給 = 左邊的變數 i 當成 i 的新值。

設定變數

範例程式 3: (set-i-j.c) 設定 i 為 j

```
1  #include <stdio.h>
2  main()
3  {
4     int i;
5     int j;
6     scanf("%d", &i);
7     scanf("%d", &j);
8     printf("%d\n", i);
9     i = j;
10    printf("%d\n", i);
11 }
```

輸入

```
1 3
2 5
```

輸出

```
1 3
2 5
```

- i 的值會由 3 變成 5，而舊值 3 就消失了。

交換變數值

片語 4: 交換兩個變數的值

```
1 temp = i;
2 i = j;
3 j = temp;
```

- 先將 i 的值複製一份到另一個變數 temp 中， 再將 j 的值指定給 i。 再將 temp 中 i 的舊值指定給 j。
- 必須宣告另一個變數 temp，作為暫存之用。

範例程式 5: (swap.c) 交換 i j 的值

```
1  #include <stdio.h>
2  main()
3  {
4     int i;
5     int j;
6     int temp;
7     scanf("%d", &i);
8     scanf("%d", &j);
9     temp = i;
10    i = j;
11    j = temp;
12    printf("%d\n", i);
13    printf("%d\n", j);
14 }
```

輸入

```
1 3
2 5
```

輸出

```
1 5
2 3
```

指定算式

片語 6: 使用指定當算式

```
1 i = j = 0;
```

- 指定句本身也可以當一個算式，而值就是 = 右邊的值。
- 由於 = 規定由右邊先算，這時就等於 `i = (j = 0);`。 意思是先將 0 指定給 j 當新值， 然後再將 j 的新值 (此時為 0) 指定給 i 當新值。

範例程式 7: (multiple-assign.c) 使用指定當算式

```
1  #include <stdio.h>
2  main()
3  {
4     int i = 1;
5     int j = 2;
6     int k = 3;
7
8     scanf("%d", &i);
9     k = j = i;
10    printf("%d\n", i);
11    printf("%d\n", j);
12    printf("%d\n", k);
13 }
```

輸入

```
1 10
```

輸出

```
1 10
2 10
3 10
```

算術運算

意義	數學表示法	C 語言表示法
加	$+$	+
減	$-$	-
乘	$\times$	*
除	$/$	/
求餘數	mod	%
求負數	$-$	-

Table： 算術運算子

運算子與運算元

- 算數算式是由運算子 (operator) 及運算元 (operand) 所構成。
- 在 C 程式語言 中，算數運算子包括加 (+)，減 (-)， 乘 (*)，除 (/)，求餘數 (%)，以及求負數 (-)。

要點

特殊字元

* 代表乘法。 % 代表求餘數。

片語 8: 計算和、差、積、商、或是餘數

```
1  k = i + j;
2  k = i - j;
3  k = i * j;
4  k = i / j;
5  k = i % j;
6  k = -i;
```

- 以上列舉 c 程式語言 的算術算式運算子，並將一個整數變數設定為另兩個整數變數的和、 差、 積、 商、 餘數、 或是負數。

運算子

- 和、 差、 積、 商，餘數等運算子稱為二元運算子 (binary operator)， 因為他們需要兩個運算元。
- 求負數的 - 只需要一個運算元，稱為一元運算子 (unary operator)。

要點

風格要點

在二元運算子前後各加一個空白可以使程式易於閱讀，因為可以清楚看到運算子。

加減運算

範例程式 9: (set-i-j-sum.c) 設定 k 為 i 及 j 的和或差

```
1  #include <stdio.h>
2  main()
3  {
4     int i, j, k;
5
6     scanf("%d", &i);
7     scanf("%d", &j);
8     k = i + j;
9     printf("%d\n", k);
10    k = i - j;
11    printf("%d\n", k);
12 }
```

輸入

```
1 3
2 9
```

輸出

```
1 12
2 -6
```

加減運算

範例程式 10: (set-i-j-divide.c) 設定 k 為 i 及 j 的商

```
1  #include <stdio.h>
2  main()
3  {
4     int i, j, k;
5     scanf("%d", &i);
6     scanf("%d", &j);
7     k = i / j;
8     printf("%d\n", k);
9     scanf("%d", &i);
10    scanf("%d", &j);
11    k = i / j;
12    printf("%d\n", k);
13 }
```

輸入

```
1  13
2  5
3  -13
4  5
```

輸出

```
1  2
2  -2
```

- 整數的除法是沒有小數的，所以 13 / 5 = 2，-13 / 5 = -2。

要點

學習要點

整數的除法是沒有小數的。 答案如果有小數就一律捨去。

餘數運算

範例程式 11: (set-i-j-mod.c) 設定 k 為 i 除以 j 的餘數

```
1  #include <stdio.h>
2  main()
3  {
4     int i, j, k;
5     scanf("%d", &i);
6     scanf("%d", &j);
7     k = i % j;
8     printf("%d\n", k);
9     scanf("%d", &i);
10    scanf("%d", &j);
11    k = i % j;
12    printf("%d\n", k);
13 }
```

輸入

```
13
5
-13
5
```

輸出

```
3
-3
```

- 當 i 為負數時，餘數也為負數。

應用

範例程式 12: (two-digit.c) 計算一個二位數的十位數及個位數

```
1  #include <stdio.h>
2  main()
3  {
4     int i;
5     int k;
6     scanf("%d", &i);
7     k = i / 10;
8     printf("%d\n", k);
9     k = i % 10;
10    printf("%d\n", k);
11 }
```

輸入

```
1 47
```

輸出

```
1 4
2 7
```

- 使用 / 運算子求 i 除以 10 的商數，也就是十位數。
- 再使用 % 運算子求 i 除以 10 的餘數，也就是個位數。

負數運算

範例程式 13: (negative.c) 計算一個數的負數

```
1  #include <stdio.h>
2  main()
3  {
4     int i;
5     int j;
6     scanf("%d", &i);
7     j = -i;
8     printf("%d\n", j);
9  }
```

輸入

```
1 -10
```

輸出

```
1 10
```

調整變數值

片語 **14:** 把 `i` 的值加/減/乘以/除以/ `j` 的簡潔表示

```
1 i  +=  j;
2 i  -=  j;
3 i  *=  j;
4 i  /=  j;
5 i  %=  j;
```

- 在計算中經常需要根據變數的舊值設定新值，
- `i += j` 意義等同 `i = i + j`。

片語 15: 把 i 的值加/減 1

```
1 i = i + 1;
2 i = i - 1;
```

- 將一個變數加一或減一是常用的片語。
- 我們可以用一個變數來記錄某件事發生的次數，也就是做為**計數器**。 或是做為**倒數計數器**，每次減一直到零為止。

片語 16: 把 i 的值加/減 1 的簡潔表示

```
1 i++;
2 i--;
```

意義	原來的表示法	簡潔的表示法
i 加上 j	i = i + j;	i += j;
i 減去 j	i = i - j;	i -= j;
i 乘以 j	i = i * j;	i *= j;
i 除以 j	i = i / j;	i /= j;
i 對 j 求餘數	i = i % j;	i %= j;
i 加 1	i = i + 1;	i++;
i 減 1	i = i - 1;	i--;

Table： 算術運算的簡潔表示法

片語 17: 印出變數所佔記憶體的位元組數

```
1 printf("%d\n", sizeof(variable));
```

- 到目前為止變數均為整數 `int`。
- 計算機使用固定數目的位元組代表一個整數， 所以 `int` 所能存的整數大小是有一定範圍的。 這個範圍與一個 `int` 在記憶體中佔幾個位元組有關。
- 可以使用 `sizeof` 察知變數所占的位元組。

範例程式 18: (sizeof-int.c) 計算 int 所佔位元組數

```
1  #include <stdio.h>
2  main()
3  {
4     int i;
5     printf("%d\n", sizeof(i));
6     scanf("%d", &i);
7     printf("%d\n", i);
8     i++;
9     printf("%d\n", i);
10 }
```

輸入

```
1 2147483647
```

輸出

```
1 4
2 2147483647
3 -2147483648
```

學習要點

如果計算的結果超過一個變數所能記錄的範圍就是溢位。 溢位會使計算的結果不正確。

範例程式 19: (set-arith-order.c) 數學運算的先後順序

```
1  #include <stdio.h>
2  main()
3  {
4     int i, j, k;
5     scanf("%d", &i);
6     scanf("%d", &j);
7     k = i + 4 * j;
8     printf("%d\n", k);
9     k = (i + 4) * j;
10    printf("%d\n", k);
11 }
```

輸入

```
1 3
2 5
```

輸出

```
1 23
2 35
```

比較運算

意義	數學表示法	C 語言表示法
大於等於	$\geq$	>=
大於	$>$	>
小於等於	$\leq$	<=
小於	$<$	<
等於	$=$	==
不等於	$\neq$	!=

Table： 整數比較

片語 20: 比較一個整數變數及常數

```
1 i > 0
2 i >= 0
3 i < 0
4 i <= 0
5 i == 0
6 i != 0
```

片語 21: 比較兩個整數變數

```
1 i > j
2 i >= j
3 i < j
4 i <= j
5 i == j
6 i != j
```

學習要點

C 程式語言 中比較兩個變數是用 ==，而非 =。 這是初學者常犯的錯誤。

特殊字元

! 在 C 程式語言 中是 非 的意思，所以 != 就是不等於。

片語 22: 比較一個整數變數及一個整數算術算式

```
1 i > (j + n)
2 i >= (j + n)
3 i < (j + n)
4 i <= (j + n)
5 i == (j + n)
6 i != (j + n)
```

學習要點

算術運算先做，比較運算後做。

風格要點

加上小括號（ ）强調運算的順序可增加程式的可讀性。

片語 23: 偶數判斷

```
1 (i % 2) == 0
```

片語 24: 直角三角形判斷

```
1 (a * a) + (b * b) == (c * c)
```

範例程式 25: (print-bool.c) 印出比較運算的結果

```
1  #include <stdio.h>
2  main()
3  {
4     int i, j, k;
5     scanf("%d", &i);
6     scanf("%d", &j);
7     k = (i == 3);
8     printf("%d\n", k);
9     k = (i == j);
10    printf("%d\n", k);
11 }
```

輸入

```
1 3
2 5
```

輸出

```
1 1
2 0
```

學習要點

C 程式語言 用一個非零的整數來對應對邏輯中的 **真**， 用零來對應對邏輯中的 **偽**。

邏輯運算

1. 且
   邏輯的 i and j 意思是 i 和 j 兩者都要為 **真** 答案才為 **真**。 這在 C 中 寫成 i && j。
2. 或
   邏輯的 i or j 意思是 i 或 j 有一個是 **真** 答案即為 **真**。 這在 C 中 寫成 i || j。
3. 反
   邏輯的 not i 意思是 i 為 **真** 時答案為 **偽**。 i 為 **偽** 時答案為 **真**。 這在 C 中 寫成 !i。

範例程式 26: (strange-logic.c) 整數使用邏輯運算

```
1  #include <stdio.h>
2  main()
3  {
4     int i = 3;
5     int j = 0;
6     int k;
7     k = (i && j);
8     printf("%d\n", k);
9     k = (i || j);
10    printf("%d\n", k);
11    k = !i;
12    printf("%d\n", k);
13 }
```

輸入

```
1 3
2 0
```

輸出

```
1 0
2 1
3 0
```

- i 為 3 非 0，被當成 **真**
- j 為 0，被當成 **偽**。

運算順序

- C 程式語言 是先算一元運算子，再算算術運算，再算比較運算，再算邏輯運算， 最後算調整運算。
- 一元運算子包括加一 ++，減一 --，求負數 -， 非 !。
- 算術運算是先乘除後加減。

意義	C 語言表示法
加一、 減一	++、 --
求負數	-
非	!
乘、 除、 求餘數	*、 /、 %
加、 減	+、 -
等於、 不等於、 大於、 小於、 大於等於、 小於等於	==、 !=、 >、 <、 >=、 <=
且、 或	&&、 ||
指定及調整變數值	+=、 -=、 *=、 /=、 %=

Table： 計算順序越上面越先算

範例程式 27: (leap-year.c) 判斷閏年

```
1  #include <stdio.h>
2  main()
3  {
4     int year;
5     int k;
6     scanf("%d", &year);
7     k = (year % 400 == 0) ||
8        ((year % 4 == 0) && (year % 100 != 0));
9     printf("%d\n", k);
10 }
```

輸入

```
1 2000
```

輸出

```
1 1
```

- 一個閏年必須是 400 的倍數，或是 4 的倍數但非 100 的倍數。
- 一個句子如果太長，可以接到下一行寫。
- 用小括號（ ）將運算的順序强調清楚，

快捷運算

- 如果一個邏輯算式經由部分算式已經可以確認真偽，則剩下的部分不會執行。
- 例如 (a > 0) || (b > 0)。 如果 a 的值為 7，則 (a > 0) 為真，我們就可以不用檢查 (b > 0) 是否為真，而直接判定 (a > 0) || (b > 0) 為真。

範例程式 28: (short-circuit.c) 快捷運算造成意外執行結果

```
1  #include <stdio.h>
2  main()
3  {
4     int i, j, k = 3, l = 4;
5     scanf("%d", &i);
6     scanf("%d", &j);
7     if ((k = i) > 0 || (l = j) > 0) {
8        printf("%d\n", k);
9        printf("%d\n", l);
10    }
11 }
```

輸入

```
1 1
2 2
```

輸出

```
1 1
2 4
```

範例程式 29: (no-short-circuit.c) 避免快捷運算的意外執行結果

```
1  #include <stdio.h>
2  main()
3  {
4     int i, j, k = 3, l = 4;
5     scanf("%d", &i);
6     scanf("%d", &j);
7     k = i;
8     l = j;
9     if (k > 0 || l > 0) {
10       printf("%d\n", k);
11       printf("%d\n", l);
12    }
13 }
```

輸入

```
1 1
2 2
```

輸出

```
1 1
2 2
```

學習要點

避免在複雜的邏輯算式改變變數值， 以免快捷運算造成意料之外的執行結果。