Chapter 04

變數與常數

學習綱要

- 4-1　識別字與保留字
- 4-2　變數和常數宣告與應用
- 4-3　資料型態轉換
- 4-4　資料型態應用實例

學習目標

- 1. 認識識別字與保留字。
- 2. 能宣告變數與常數。
- 3. 能實作資料型態轉換。
- 4. 能實作資料型態應用。

4-1 識別字與保留字

識別字(Identifiers)

眞實的世界裏，每個人、事及物都有一個名稱，程式設計亦不例外，於程式設計時我們必須爲每一個變數、常數、函式、結構及類別命名，以上所有變數、常數、函式、結構及類別等名稱，統稱爲程式語言的識別字。C 語言的識別字命名規則如下：

1. 識別字必須是以字母（大小寫的 A 至 Z）或底線 (_) 開頭。例如，以下是一些合法的識別字。

```
a
i
sum
_sum
Income
```

以下是一些非法的識別字。

```
7eleven      /*  不能由數字開頭  */
%as          /*  不能由符號開頭  */
```

2. 識別字由字母開頭後，僅可由字母、底線及數字組合而成，但不得包含空白。例如，以下是一些合法的識別字。

```
a123
a123b
_a_b
```

以下是一些非法的識別字。

```
A=           /*  不能含有 = 號 */
sum!         /*  不能含有 ! 號 */
Age#3        /*  不能含有 # 號 */
a c      /*  不能含空白     */
c+3          /*  不得含有加號  */
```

3. 識別字的長度不可以超過 32 個字元。
4. 識別字的大小寫均視爲不同，例如 Score、score 及 SCORE 皆代表不同的識別字。
5. 識別字不得使用保留字，如 if、for 等。但是，if1、fora 等則可使用。
6. 識別字亦可使用有意義的單字的組合，這樣更能傳達變數所代表的意義。例如：StudentNumber 或 AverageIncome。除非有效範圍很小（或稱生命週期極短）的變數才用 x、i 或 a 等當識別字，也千萬不要用 k23erp 等這種沒意義又難記的識別字。
7. 識別字有多個單字時，中間可以加上底線（_），例如上例的 StudentNumber 可寫成 Student_Number，若擔心單字拼錯亦可寫成 Stu_Num、stu_num、StuNum 或 stumum，其中 StuNum 又稱駝峰表示法，因爲大寫字母看起來像駱駝駝峰一樣，可以避免鍵入底線的困擾、且提昇閱讀效率。
8. 不要單獨使用 l,o,O 等字母當作變數名稱，因爲這些字母與阿拉伯數字 1,0 很相近，很容易讓人誤判。

保留字(Keywords)

保留字（又稱關鍵字）是任一程式語言已事先賦予某一識別字（可識別的文字或字串，稱爲識別字）一個特別意義，所以程式設計者不得再重複賦予不同的用途。例如 if 已賦予決策敘述，程式設計者當然不得再定義 if 爲另外的用途，表 4-1 是 C 語言的保留字，請於 https://en.cppreference.com/w 點選『C Keywords』。

▶ 表 4-1　C 語言保留字

auto break case char const continue default do double else enum extern	float for goto if inline (since C99) int long register restrict (since C99) return short	signed sizeof static struct switch typedef union unsigned void volatile while	_Alignas (since C11) _Alignof (since C11) _Atomic (since C11) _Bool (since C99) _Complex (since C99) _Generic (since C11) _Imaginary (since C99) _Noreturn (since C11) _Static_assert (since C11) _Thread_local (since C11)

4-2 變數與常數宣告與應用

變數的宣告

變數的功能是用來輸入、處理及儲存外界的資料，而變數在使用以前均要事先宣告才可使用。C 語言的變數宣告語法如下：

```
資料型態 變數名稱[=初值];
```

例如：

```
int a;
```

即是宣告變數 a 爲 int 型態。又例如：

```
char b,c;
```

則是宣告變數 b,c 爲 char 型態。變數的宣告亦可連同初值一起設定，如下所示：

```
 double d = 30.2;
```

宣告 d 是 double 型態，並設其初值是 30.2。以下程式，宣告 e 是 char 型態，且其初值是 'a'。

```
char e='a';
```

以下敘述，同時宣告兩個整數，且指派其初值。

```
int f1=3,f2=4;
```

以下敘述可宣告變數爲字元陣列，字元陣列就能代表字串。(陣列於 9-3 節有更詳細說明)

```
char g[]="Gwosheng";
```

變數宣告應用

變數經過宣告之後，編譯器即會根據該變數的資料型態配置適當的記憶體儲存此變數，所以若要提高程式的執行效率，則應儘量依照資料性質，選擇佔用記憶體較小的資料型態。在一些舊式的 Basic 語言中，變數使用前並不需要事先宣告，卻也帶來極大的困擾。以下敘述即爲變數未宣告的結果，編譯器便無法回應使用者在拼字上的錯誤，而造成除錯上的困難。

```
student=studend+1;
```

上式若事先宣告 student 如下：

```
int student;
```

則編譯器遇到 studend 時，便會出現 studend 未宣告的錯誤訊息，提醒使用者補宣告或注意拼字錯誤。其次，變數宣告的優點是可配置恰當的記憶體而提高資料的處理效率。例如，有些變數的值域僅爲整數，則不用宣告爲 float 或 double。此即爲小東西用小箱子裝，大東西用大箱子裝，才能有效運用空間與提升搬運效率。

自我練習

1. 請鍵入以下程式，編譯、除錯與執行程式。

題號	程式	結果

1	`#include <stdio.h>`
	`#include <stdlib.h>`
	`int main() {`
	`    int 7eleven;      //  不能由數字開頭`
	`    int %as;          //  不能由符號開頭`
	`    int _a;`
	`    int A=;           //  不能含有 = 號`
	`    int sum! ;        //  不能含有 ! 號`
	`    int Age#3;        //  不能含有 # 號`
	`    int a c ;         //  不能含空白`
	`    int c+3 ;         //  不得含有加號`
	`    int if;           // 不得使用保留字`
	`    return 0;`
	`}`	

2	#include <stdio.h>
	#include <stdlib.h>
	int main() {
	int student=0;
	student=studend+1;
	printf("%d",student);
	int a=3.4;
	char b,c;
	b="aa"
	c="a";
	int d='a';
	double e=3;
	return 0;
	}	

常數的宣告

程式設計有兩種表示常數的方式，一種是文字式 (Literal)，例如直接以 15 或 3.14159 表示某一常數；另一種是本單元的常數符號式 (Symbolic) 。因爲有些數字在程式中會不斷的重複出現，爲了增加程式的可讀性及減少程式鍵入的麻煩，此時即可用一個有意義的符號代替，我們可以利用 const 或 define 來定義常數，則該符號的值將永遠保持在你所宣告的符號，程式中任何位置均不可能改變其值，此稱爲常數符號，簡稱常數。const 宣告常數，語法如下：

```
const 型態 常數名稱=常數值;
```

例如，以下程式令 PI=3.14，常數符號通常均用大寫表示。

```
const double PI=3.14;
```

則每次要使用 3.14 時，只要填入 PI 即可。例如，以下程式可計算圓面積。

```
a=3*3*PI；
```

define 定義常數語法如下：

```
#define 常數名稱 常數值
```

例如：

```
#define PI 3.14;
```

常數宣告的應用

又例如，於銀行利息的計算，利率亦應以常數符號表示，程式中任何位置若需使用此利率時，均應以此常數符號代替，當要調整此利率時，只要在程式的最前面修改此常數符號的值即可。若未使用常數符號統一此值，則因此常數散落程式各地，而必須到處修改，萬一有些地方漏掉未修改，則無法確保此值的一致性。

自我練習

1. 請鍵入以下程式，並寫出執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    const double PI=3.14;
    double a;
    a=3*3*PI;
    printf("%f\n",a);
    const int b=4;
    b=b+1;//常數不能改變其值
    printf("%d",b);
    return 0;
}
```

2. 請鍵入以下程式，並寫出執行結果。

```
#include <stdio.h>
#include <stdlib.h>
#define PI 3.14
int main() {
    double a;
    a=3*3*PI;
    printf("%lf\n",a);
    PI=PI+1;//常數不能改變其值
    printf("%lf",PI);
    return 0;
}
```

3. 關於 C 程式語言中，使用 define 建立常數的方式，下列何者正確？（111 年統測）
   (A) define PI = 3.14;
   (B) define PI 3.14;
   (C) #define PI = 3.14
   (D) #define PI 3.14

4-3 資料型態轉換

每一個變數宣告之後，即有屬於自己的型態，往後此變數均只能指派給相同或相近型態的變數儲存，若執行階段欲指定給相近型態的變數儲存，則稱此爲型態轉換。在 C 語言中型態轉換又可分爲隱含轉換（Implicit Conversion）與強制轉換（Explicit Conversion），分別說明如下：

隱含轉換（Implicit Conversion）

將值域小的型態轉爲值域大的型態，稱爲自動轉換或轉型（Convert）。此種轉換，系統可自動處理並確保資料不會流失。例如，將 int 轉爲 long，則因後者的值域比前者大，所以可順利的轉換。以下敘述可將型態爲 int 的變數 a 指派給型態爲 long 的變數 b，且原值不會改變。

```
int a=23;
long b;
b=a;
printf("%d",b);
```

結果是 23。

強制轉換（Emplicite Conversion）

將值域大的變數轉爲值域小的變數（如 long 轉爲 int），則稱此爲強制轉換或稱爲鑄型（cast）。強制轉換的語法如下：

```
變數2 （值域小） = （變數2的型態） 變數1（值域大） ;
```

以上可將變數 1（值域大）的型態強制轉爲變數 2（值域小）。例如，以下敘述可將型態是 long 的 c 變數指定給型態是 int 的 d 變數。

```
long c=23;
int d;
d=(int) c;
printf("%d\n",d);
```

結果是 23。強制轉換的風險比較大，有可能資料流失或溢位。例如，以下敘述將 float 型態強制轉換爲 int 型態，將造成小數部份的流失。

```
float e=3.4f;
   //浮點數預設爲double，加上f,表示此3.4爲float，請複習3-2節浮點數
int k;
k=(int) e;
printf("%d\n",k);//3
```

結果 3。其次，關於數值與字串互轉，則要看 11-3 的字串函式。

自我練習

1. 下列 C 語言程式碼片段執行後，x 與 y 的結果爲何？（統測 111）

```
int x, a = 7, b = 2;
float y;
x = a/b;
y = (float)a/b;
```

4-4　資料型態應用實例

範例4-4a

假如有資料如下，請問如何以電腦儲存？

座號	姓名	國文成績	數學成績	總分	平均
1	洪煖維	98	95		

程式設計步驟

1. 資料數位化且以變數儲存。
   (1) 座號取 int 型態，變數名稱取 no。
   (2) 姓名取 char 型態，變數名稱取 name。
   (3) 國文成績取 int 型態，變數名稱取 chi。
   (4) 數學成績取 int 型態，變數名稱取 math。
   (5) 總分成績取 int 型態，變數名稱取 total。
   (6) 平均取 double 型態，變數名稱取 avg。

2. 程式設計如下：

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int no=1;
    char name[]="洪焌維";
    int chi=98;
    int math=95;
    int total;
    double avg;
    return 0;
}
```

自我練習

1. 假設有資料如下，請問如何以電腦儲存？

單字	詞性	中文
good	adj	美好的