บทที่ 3
การนำข้อมูลเข้าและออกในภาษา C
(Data input and output in C)

การนำข้อมูลเข้าในภาษา C จะประกอบด้วยฟังก์ชัน scanf, gets และ getchar ส่วนการนำข้อมูลออกในภาษา C จะประกอบด้วยฟังก์ชัน printf, puts และ putchar ซึ่งทุกฟังก์ชันนั้นได้ถูกกำหนดไว้ในไลบรารีฟังก์ชัน เมื่อเรานำฟังก์ชันเหล่านี้มาใช้ เราจะต้องกำหนด #include <stdio.h> ไว้ที่ตอนต้นของโปรแกรม ในบทนี้เราจะกล่าวถึงฟังก์ชันนำข้อมูลเข้า คือ scanf ซึ่งจะรับข้อมูลชนิดจำนวน, ตัวอักขระ และสายอักขระ และฟังก์ชันนำข้อมูลออก คือ printf ซึ่งจะพิมพ์ข้อมูลชนิดจำนวน, ตัวอักขระ และสายอักขระ

3.1 รูปแบบทั่วไปของฟังก์ชัน printf มีดังนี้

printf (format-control-string, other-arguments) ;

โดย format-control-string จะอธิบายถึงรูปแบบของแสดงผล ซึ่ง format-control-string จะประกอบด้วย ตัวระบุการแปลงผัน (conversion specifiers), แฟล็ก (flags), ความกว้างของฟิลด์ (field widths), ความเที่ยงตรง (precisions), สายอักขระ (string) และลำดับหลีก (escape sequence) ซึ่งจะต้องมีเครื่องหมาย “” ปิดล้อม ส่วน other-arguments จะเป็นส่วนที่ถูกนำมาแสดงผล ซึ่งอาจจะเป็นค่าคงตัว, ตัวแปร, นิพจน์ และฟังก์ชันที่ถูกเรียกใช้ ถ้ามีอาร์กิวเมนต์มากกว่า 1 ตัว ให้คั่นด้วยเครื่องหมาย commas (,) โดยจำนวนอาร์กิวเมนต์จะต้องสอดคล้องกับจำนวนตัวระบุและชนิดการแปลงผัน

หมายเหตุ

1. ส่วน other-arguments อาจจะไม่มีก็ได้
2. ตัวระบุแบบการแปลงผัน จะต้องมีเครื่องหมาย % นำหน้า

ตารางที่ 3.1 แสดงตัวระบุการแปลงผันที่ใช้ในการนำข้อมูลออกแสดงผล

ตัวระบุการแปลงผัน	ความหมาย
จำนวนเต็ม (integer)	
D	แสดงค่าจำนวนเต็มฐานสิบมีเครื่องหมาย
I	แสดงค่าจำนวนเต็มฐานสิบมีเครื่องหมาย
O	แสดงค่าจำนวนเต็มฐานสิบไม่มีเครื่องหมาย
U	แสดงค่าจำนวนเต็มฐานสิบไม่มีเครื่องหมาย
x	แสดงค่าจำนวนเต็มฐานสิบหกไม่มีเครื่องหมาย โดยใช้ภาษาอังกฤษตัวเล็ก จาก a ถึง f
X	แสดงค่าจำนวนเต็มฐานสิบหกไม่มีเครื่องหมาย โดยใช้ภาษาอังกฤษตัวใหญ่ จาก A ถึง F
จำนวนจุดลอยตัว (floating point)	
f	แสดงค่าเลขทศนิยมมีเครื่องหมาย
e	แสดงค่าเลขทศนิยมมีเครื่องหมายโดยใช้สัญลักษณ์ e
E	แสดงค่าเลขทศนิยมมีเครื่องหมายโดยใช้สัญลักษณ์ E
g	แสดงค่าจำนวนเต็มฐานสิบมีเครื่องหมายโดยใช้รูปแบบ e หรือ f ขึ้นอยู่ว่า แบบใดมีขนาดสั้นกว่า
G	แสดงค่าจำนวนเต็มฐานสิบมีเครื่องหมายโดยใช้รูปแบบ E หรือ f ขึ้นอยู่ว่า แบบใดมีขนาดสั้นกว่า
อักขระและสายอักขระ (character and string)	
c	แสดงอักขระ 1 ตัว
s	แสดงสายอักขระ
%	แสดงเครื่องหมาย %

หมายเหตุ 1. เราจะใช้ l เติมหน้า d, u, x, 0 เพื่อระบุว่าเป็น long integer ตัวอย่างเช่น %ld

2. เราจะใช้ h เติมหน้า d, u, x, 0 เพื่อระบุว่าเป็น short integer

ตารางที่ 3.2 แสดงความหมายของลำดับหลีก

ลำดับหลีก		ความหมาย
\'	(single quote)	แสดงอักขระ '
\"	(double quote)	แสดงอักขระ "
\?	(question mark)	แสดงอักขระ ?
\\	(backslash)	แสดงอักขระ \
\a	(alert or bell)	ทำให้เกิดเสียงกริ่งหรือระฆัง
\b	(backspace)	เลื่อนเคอร์เซอร์ถอยกลับไป 1 ตำแหน่ง
\f	(new page or form feed)	เลื่อนเคอร์เซอร์ไปหน้าถัดไป
\n	(newline)	เลื่อนเคอร์เซอร์ไปเริ่มต้นที่บรรทัดถัดไป
\r	(carriage return)	เลื่อนเคอร์เซอร์ไปที่จุดเริ่มต้นของบรรทัดในปัจจุบัน
\t	(horizontal tab)	เลื่อนเคอร์เซอร์ไปแนวนอน 1 tab
\v	(vertical tab)	เลื่อนเคอร์เซอร์ไปแนวตั้ง 1 tab

3.2 การพิมพ์จำนวนเต็ม (Printing Integers)

จำนวนเต็มจะประกอบด้วย

1. จำนวนเต็มบวก ได้แก่ 1, 2, 3, ...
2. จำนวนเต็มศูนย์ ได้แก่ 0
3. จำนวนเต็มลบ ได้แก่ –1, –2, –3, ...

หมายเหตุ จำนวนเต็มจะเป็นตัวเลขที่ไม่มีจุดทศนิยม ซึ่งในการแสดงค่าจำนวนเต็มจะมีหลายรูปแบบ เช่น %d, %i, %o, %u, %x, %X

ตัวอย่างโปรแกรมที่ 3.1 เป็นโปรแกรมแสดงการพิมพ์ค่าจำนวนเต็มโดยใช้ตัวระบุการแปลงผันจำนวนเต็ม (integer conversion specifiers)

โดยปกติ จำนวนเต็มที่มีค่าเป็นจำนวนบวก จะไม่แสดงเครื่องหมาย + ส่วนจำนวนเต็มที่มีค่าเป็นจำนวนลบ จะแสดงเครื่องหมาย –

```
/* Using the integer conversion specifiers */
#include <stdio.h>

int main ()
{
    printf ( “%d\n”, 455 ) ;
    printf ( “%i\n”, 455 ) ;  /* i same as d in printf */
    printf ( “%d\n”, +455 ) ;
    printf ( “%d\n”, –455 ) ;
    printf ( “%hd\n”, 32000 ) ;
    printf ( “%ld\n”, 2000000000 ) ;
    printf ( “%o\n”, 455 ) ;
    printf ( “%u\n”, 455 ) ;
    printf ( “%u\n”, –455 ) ;
    printf ( “%x\n”, 455 ) ;
    printf ( “%X\n”, 455 ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
455
455
455
–455
32000
2000000000
707
455
4294966841
1c7
1C7
```

3.3 การพิมพ์จำนวนจุดลอยตัว (Printing Floating–Point Numbers)

ค่าจุดลอยตัว จะเป็นจำนวนที่มีจุดทศนิยม เช่น 33.5, 0.0 หรือ –657.983 เป็นต้น ซึ่งในการแสดงค่าจุดลอยตัว จะมีหลายแบบ เช่น %e, %E, %g, %G

ซึ่งตัวระบุการแปลงผัน e และ E จะแสดงค่าจุดลอยตัวในรูปสัญลักษณ์เลขชี้กำลัง (exponential notation) โดยสัญลักษณ์เลขชี้กำลังในคอมพิวเตอร์ จะสมมูลกับสัญลักษณ์ทางวิทยาศาสตร์ (scientific notation) ที่ใช้ในทางคณิตศาสตร์ ตัวอย่างเช่น

ค่า 150.4582 เขียนเป็นสัญลักษณ์ทางวิทยาศาสตร์ได้เป็น 1.504582×10^2 แต่เมื่อเขียนเป็นสัญลักษณ์เลขชี้กำลัง จะเขียนได้เป็น 1.504582E+02 ซึ่งจะใช้ในคอมพิวเตอร์ โดย E มาจากคำว่า “exponent.”

ค่าที่พิมพ์ด้วยตัวระบุการแปลงผันแบบ e, E และ f ถ้าไม่มีการกำหนดความเที่ยงตรง ผลลัพธ์จะมีความเที่ยงตรงเท่ากับ 6 ซึ่งเป็นค่า default ตัวระบุการแปลงผัน แบบ f จะพิมพ์ตัวเลขอย่างน้อย 1 ตัว ทางด้านซ้ายของจุดทศนิยม

ตัวระบุการแปลงผันแบบ e และ E จะพิมพ์อักษร e และ E นำหน้าเลขชี้กำลัง และต้องพิมพ์ตัวเลข 1 หลัก แน่นอนทางซ้ายมือของจุดทศนิยม ตัวระบุการแปลงผันแบบ g หรือ G อาจจะพิมพ์ในแบบ e (E) หรือ f โดยไม่มีเลขศูนย์ต่อท้าย เช่น 1.234000 จะถูกพิมพ์เป็น 1.234

ถ้าค่าที่พิมพ์ด้วยรูปแบบ e (E) หลังจากเปลี่ยนค่าเป็นสัญลักษณ์เลขชี้กำลัง โดยถ้าค่าของเลขชี้กำลังน้อยกว่า –4 หรือเลขชี้กำลังมากกว่าหรือเท่ากับความเที่ยงตรงที่ได้กำหนดไว้ (ความเที่ยงตรงเท่ากับ 6 ซึ่งเป็นค่า default ของ g และ G) มิฉะนั้นแล้วตัวระบุการแปลงผันแบบ f จะถูกนำมาใช้พิมพ์ค่า และไม่มีเลขศูนย์มาต่อท้ายในส่วนที่เป็นเศษส่วน และต้องมีทศนิยมอย่างน้อย 1 ตำแหน่ง

ตัวอย่างที่ 3.1 ค่า 0.000087 จะใช้สัญลักษณ์ e เนื่องจาก $0.000087 = 8.7 \times 10^{-5}$ มีเลขชี้กำลังเป็น –5 ซึ่งน้อยกว่า –4 และเขียนเป็นสัญลักษณ์ของ e ได้คือ 8.75e – 05

ตัวอย่างที่ 3.2 8750000.0 จะใช้สัญลักษณ์ e เนื่องจาก $8750000.0 = 8.750000 \times 10^{6}$ มีเลขชี้กำลังเป็น 6 ซึ่งเท่ากับ ค่า default ของความเที่ยงตรงและเขียนเป็นสัญลักษณ์ของ e ได้คือ 8.75e + 06

ตัวอย่างที่ 3.3 8.75 จะใช้สัญลักษณ์ f เนื่องจาก $8.75 = 8.75 \times 10^{1}$ มีเลขชี้กำลังมากกว่า –4 และน้อยกว่า 6 และเขียนเป็นสัญลักษณ์ f ได้คือ 8.75

ตัวอย่างที่ 3.4 87.50 จะใช้สัญลักษณ์ f เนื่องจาก $87.50 = 8.750 \times 10^{1}$ มีเลขชี้กำลังมากกว่า –4 และน้อยกว่า –6 และเขียนเป็นสัญลักษณ์ f ได้คือ 87.5 (จะไม่มีเลข 0 ตามท้าย)

ความเที่ยงตรงของตัวระบุการแปลงผัน g และ G จะเป็นตัวกำหนดขนาดที่มากที่สุดของตัวเลขนัยสำคัญที่ใช้ในการพิมพ์ เช่น 1234567.0 จะพิมพ์เป็น 1.23457e + 06 เมื่อใช้ตัวระบุการแปลงผัน %g

ตัวอย่างโปรแกรมที่ 3.2 เป็นโปรแกรมแสดงการพิมพ์ค่าจุดลอยตัวโดยใช้ตัวระบุการแปลงผันจุดลอยตัว ตัวระบุการแปลงผันแบบ %e, %E, %g และ %G จะทำให้ค่าที่ได้มีการปัดเศษ แต่ถ้าใช้ตัวระบุการแปลงผันแบบ %f จะไม่มีการปัดเศษ

```
/* Printing floating–point numbers with
   floating–point conversion specifiers  */

#include <stdio.h>

int main ()
{
    printf ( “%e\n”, 1234567.89 ) ;
    printf ( “%e\n”, +1234567.89 ) ;
    printf ( “%e\n”, –1234567.89 ) ;
    printf ( “%E\n”, 1234567.89 ) ;
    printf ( “%f\n”, 1234567.89 ) ;
    printf ( “%g\n”, 1234567.89 ) ;
    printf ( “%G\n”, 1234567.89 ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
1.234568e+006
1.234568e+006
–1.234568e+006
1.234568E+006
1234567.890000
1.23457e+006
1.23457E+006
```

3.4 การพิมพ์สายอักขระและอักขระ (Printing String and Characters)

ตัวระบุการแปลงผัน c และ s ใช้ในการพิมพ์อักขระและสายอักขระ ตามลำดับ โดยตัวระบุการแปลงผัน c ต้องการอาร์กิวเมนต์แบบ char และตัวระบุการแปลงผัน s ต้องการอาร์กิวเมนต์ เป็นตัวชี้ (pointer) ไปยัง char ซึ่งเป็นอาร์กิวเมนต์ โดยตัวระบุการแปลงผัน s จะพิมพ์อักขระไปเรื่อย ๆ จนกระทั่งพบอักขระ terminating null (‘\0’)

ตัวอย่างโปรแกรมที่ 3.3 เป็นโปรแกรมแสดงการพิมพ์ โดยใช้ตัวระบุการแปลงผันแบบ c และ s

```
/* Printing strings and characters */
#include <stdio.h>

int main ()
{
    char character = 'A' ;  /* initialize char */
    char string [] = "This is a string" ;  /* intialize char array */
    conts char *stringPtr = "This is also a string" ;  / char pointer */

    printf ( "%c\n", character ) ;
    printf ( "%s\n", "This is a string" ) ;
    printf ( "%s\n", string ) ;
    printf ( "%s\n", stringPtr ) ;

    return 0 ;  /* indicates successful termination */
}   /* end main */
```

```
A
This is a string
This is a string
This is also a string
```

3.5 การพิมพ์เมื่อกำหนดความกว้างของฟิลด์ และความเที่ยงตรง

(Printing with Field Widths and Precision)

ขนาดของขอบเขตในการพิมพ์ข้อมูลที่แน่นอน เราจะระบุโดยความกว้างของฟิลด์ (Field width) ถ้าความกว้างของฟิลด์มากกว่าจำนวนหลักของข้อมูลที่จะพิมพ์ ข้อมูลจะถูกกำหนดให้พิมพ์ชิดขวาภายในฟิลด์ เราจะใช้เลขจำนวนเต็มเป็นการระบุความกว้างของฟิลด์ โดยจะใส่ตัวเลขนี้ระหว่างเครื่องหมาย percent (%) กับตัวระบุการแปลงผัน ตัวอย่างเช่น %4d หมายถึงจะกำหนดความกว้างให้กับข้อมูลชนิดจำนวนเต็มเท่ากับ 4 ถ้าความกว้างของฟิลด์ที่กำหนดมาให้น้อยกว่าจำนวนหลักของข้อมูลที่จะพิมพ์ ก็จะไม่ใช้ความกว้างของฟิลด์ โดยจะพิมพ์ตัวเลขจำนวนนั้นออกมาตามปกติ

ตัวอย่างโปรแกรมที่ 3.4 เป็นโปรแกรมแสดงการพิมพ์ค่าจำนวนเต็ม 2 กลุ่ม โดยแต่ละกลุ่มจะมี 5 จำนวน โดยจะกำหนดความกว้างของฟิลด์เท่ากับ 4 ข้อมูลจะถูกพิมพ์ชิดขวา ถ้าจำนวนหลักน้อยกว่าความกว้างของฟิลด์ และความกว้างของฟิลด์จะเพิ่มขึ้นเมื่อพิมพ์ค่าที่มีจำนวนหลักมากกว่าความกว้างของฟิลด์ และเครื่องหมาย (–) จะถือว่าเป็น 1 ตำแหน่งของความกว้างของฟิลด์ด้วย ความกว้างของฟิลด์สามารถใช้ได้กับทุก ๆ ตัวระบุการแปลงผัน

```
/* Printing integers right–justifield  */
#include <stdio.h>

int main ()
{
    printf ( “%4d\n”, 1 ) ;
    printf ( “%4d\n”, 12 ) ;
    printf ( “%4d\n”, 123 ) ;
    printf ( “%4d\n”, 1234 ) ;
    printf ( “%4d\n\n”, 12345 ) ;

    printf ( “%4d\n”, –1 ) ;
    printf ( “%4d\n”, –12 ) ;
    printf ( “%4d\n”, –123 ) ;
    printf ( “%4d\n”, –1234 ) ;
    printf ( “%4d\n”, –12345 ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
   1
  12
 123
1234
12345

  –1
 –12
–123
–1234
–12345
```

เราสามารถระบุความเที่ยงตรงให้กับข้อมูลที่เราจะพิมพ์ได้ โดยความเที่ยงตรงจะมีความแตกต่างกันขึ้นอยู่กับชนิดของข้อมูล กล่าวคือ

1. เมื่อใช้กับตัวระบุการแปลงผันจำนวนเต็ม ความเที่ยงตรงนี้จะเป็นตัวบอกถึงจำนวนน้อยที่สุดของตัวเลขที่จะพิมพ์ โดยค่าที่จะพิมพ์มีตัวเลขน้อยกว่าการกำหนดความเที่ยงตรง ก็จะมีการเติมศูนย์นำหน้าค่าที่จะพิมพ์ จนกระทั่งจำนวนตัวเลขทั้งหมดสมมูลกับความเที่ยงตรง แต่โดยปกติแล้วค่าที่กำหนดให้มาแล้ว (default) สำหรับจำนวนเต็มจะเท่ากับ 1

2. เมื่อใช้กับตัวระบุการแปลงผันจุดลอยตัวแบบ e, E และ f ความเที่ยงตรงจะเป็นจำนวนตัวเลขที่ปรากฏหลังจุดทศนิยม

3. เมื่อใช้กับตัวระบุการแปลงผันแบบ g และ G ความเที่ยงตรงจะเป็นขนาดที่มากที่สุดของตัวเลขนัยสำคัญที่จะพิมพ์

4. เมื่อใช้กับตัวระบุการแปลงผันแบบ s ความเที่ยงตรงจะเป็นขนาดที่มากที่สุดของจำนวนอักขระที่จะพิมพ์จากสายอักขระ การบ่งบอกความเที่ยงตรง เราจะใช้จุดทศนิยม (.) แล้วตามด้วยจำนวนเต็มที่ระบุความเที่ยงตรง และอยู่ระหว่างเครื่องหมาย % กับตัวระบุการแปลงผัน ตัวอย่างเช่น %.4d

ตัวอย่างโปรแกรมที่ 3.5 เป็นโปรแกรมแสดงการใช้ความเที่ยงตรงในรูปแบบการควบคุมสายอักขระ (format control string) โดยค่าจุดลอยตัวจะถูกพิมพ์ และมีการปัดเศษ เมื่อความเที่ยงตรงน้อยกว่าจำนวนจุดทศนิยมของจำนวนเดิม

```
/* Using precision while printing integers,
   floating–point numbers, and strings */

#include <stdio.h>

int main ()
{
    int i = 873 ;                    /* initialize int i */
    double f = 123.94536 ;           /* initialize double f */
    char s[] = “Happy Birthday” ;  /* initialize char array s */

    printf ( “Using precision for integers\n” ) ;
    printf ( “\t%.4d\n\t%. 9d\n\n” , i , i ) ;

    printf ( “Using precision for floating–point numbers\n” ) ;
    printf ( “\t%.3f\n\t%.3e\n\t%.3g\n\n” , f , f, f ) ;

    printf ( “Using precision for string\n” ) ;
    printf ( “\t%.11s\n” , s ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
Using precision for integers
        0873
        000000873
Using precision for floating–point numbers
        123.945
        1.239e+002
        124
Using precision for strings
        Happy Birth
```

ความกว้างของฟิลด์และความเที่ยงตรง เราสามารถนำมาผสมกันได้โดยการกำหนดความกว้างของฟิลด์ตามด้วยจุดทศนิยมและความเที่ยงตรงโดยจะอยู่ระหว่างเครื่องหมาย % และตัวระบุการแปลงผัน ตัวอย่างเช่น การใช้คำสั่ง printf (“%9.3f” , 123.456789) ; ซึ่งมีความหมายว่าต้องการความกว้างของฟิลด์เป็น 9 และความเที่ยงตรงเป็น 3 และสามารถแสดงผลลัพธ์ได้ดังรูป

ฟิลด์ที่ →	1	2	3	4	5	6	7	8	9
			1	2	3	.	4	5	7

หมายเหตุ เครื่องหมายจุดทศนิยม (.) จะถูกนับรวมเป็นความกว้างของฟิลด์ด้วยและข้อมูลจะถูกพิมพ์ชิดขวา

บางครั้งเราอาจจะระบุความกว้างของฟิลด์และความเที่ยงตรงโดยใช้นิพจน์จำนวนเต็มในรายการของอาร์กิวเมนต์ที่อยู่ตามรูปแบบการควบคุมสายอักขระ โดยเราจะต้องใช้เครื่องหมาย asterisk (*) แทนความกว้างของฟิลด์ หรือความเที่ยงตรงทั้งสองแบบ ตัวอย่างเช่น การใช้คำสั่ง

printf (“%*.*f”, 7, 2, 98.736) ;

โดยจะแทน 7 เป็นความกว้างของฟิลด์ และจะแทน 2 เป็น ความเที่ยงตรง และสามารถแสดงผลลัพธ์ได้ดังรูป

ฟิลด์ที่ →	1	2	3	4	5	6	7
			9	8	.	7	4

หมายเหตุ 1. ข้อมูลชนิดจำนวนเต็มและชนิดอักขระ ถ้ามีการกำหนดความเที่ยงตรงจะไม่มีผลต่อการแสดงผลลัพธ์ โดย

% 15.3d จะมีผลลัพธ์เหมือนกับ % 15d

และ % 15.3c จะมีผลลัพธ์เหมือนกับ % 15c

2. ข้อมูลชนิดสายอักขระ ถ้ามีการกำหนดตัวระบุการแปลงผันเป็น %10.3s จะเป็นการตัดอักขระออกจากสายอักขระ โดยจะแสดงเฉพาะอักขระ 3 ตัวแรกเท่านั้น ตัวอย่างเช่น การใช้คำสั่ง

printf (“%10.3s”, “abcdefghi”) ;

และสามารถแสดง ผลลัพธ์ได้ดังรูป

ฟิลด์ที่ →	1	2	3	4	5	6	7	8	9	10
								a	b	c

3.6 การใช้แฟล็กในคำสั่ง printf ในส่วนรูปแบบการควบคุมสายอักขระ (Using Flags in the printf Format Control String)

บางครั้งเราต้องการพิมพ์ข้อมูลให้ชิดซ้าย, มีเครื่องหมาย + นำหน้า เป็นต้น ในภาษา C จะมีแฟล็กอยู่ 5 ตัว ที่ใช้

ตารางที่ 3.3 แสดงเครื่องหมายแฟล็กที่ใช้ในภาษา C

แฟล็ก	ความหมาย
– (minus sign)	จัดผลลัพธ์ให้ชิดซ้ายภายในฟิลด์ที่กำหนดไว้
+ (plus sign)	จะแสดงเครื่องหมาย + นำหน้าจำนวนบวก และจะแสดงเครื่องหมาย – นำหน้าจำนวนลบ
ช่องว่าง (space)	จะแสดงอักขระว่างหน้าจำนวนบวก และจะไม่พิมพ์อักขระว่าง ถ้าใช้ร่วมกับ + flag
#	จะเติม 0 หน้าจำนวน เมื่อใช้ตัวระบุการแปลงผันฐานแปด (0) จะเติม Ox หรือ OX หน้าจำนวนเมื่อใช้ตัวระบุการแปลงผันฐานสิบหก (x หรือ X) จะบังคับให้มีจุดทศนิยมสำหรับจำนวนจุดลอยตัวที่พิมพ์ด้วย e, E, f, g หรือ G (โดยปกติจุดทศนิยมจะถูกพิมพ์ ถ้ามีตัวเลขตามหลัง) สำหรับตัวระบุ g และ G ถ้ามีเลขศูนย์ตามมาจะไม่ถูกกำจัดออก
0 (zero)	จะใส่เลขศูนย์ในฟิลด์ที่ว่างในส่วนข้างหน้าของจำนวน

หมายเหตุ เราจะใส่ flag ตามหลังเครื่องหมาย %

ตัวอย่างโปรแกรมที่ 3.6 เป็นโปรแกรมที่แสดงการพิมพ์ชิดขวาและชิดซ้ายของสายอักขระ, จำนวนเต็ม, ตัวอักขระ และจำนวนจุดลอยตัว

```
/* Right justifying and left justifying values */
#include <stdio.h>

int main ()
{
    printf ( "%10s%10d%10c%10f\n\n", "hello", 7, 'a', 1.23 ) ;
    printf ( "%–10s%–10d%–10c%–10f\n", "hello", 7, 'a', 1.23 ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
     hello         7         a   1.230000
hello      7       a          1.230000
```

ตัวอย่างโปรแกรมที่ 3.7 เป็นโปรแกรมที่แสดงการพิมพ์จำนวนบวกและจำนวนลบ โดยไม่ใช้ +flag โดยเครื่องหมาย + จะถูกแสดงเมื่อเราใช้ +flag

```
/* Printing numbers with and without the + flag */
#include <stdio.h>

int main ()
{
    printf ( "%d\n%d\n", 786, –786 ) ;
    printf ( "%+d\n%+d\n", 786, –786 ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
786
–786
+786
–786
```

ตัวอย่างโปรแกรมที่ 3.8 เป็นโปรแกรมที่แสดงการเติมหน้าด้วยช่องว่างให้กับจำนวนบวก โดยใช้ space flag จุดประสงค์เพื่อให้จำนวนบวกและจำนวนลบมีตำแหน่งของหลักตรงกัน โดยจำนวน –547 ไม่ต้องมีช่องว่างนำหน้าก็ได้ เนื่องจากมีเครื่องหมาย – อยู่แล้ว

```
/* Printing a space before signed values
   not preceded by + or – */
#include <stdio.h>

int main ()
{
    printf ( "%d \n% d\n", 547, –547 ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
 547
–547
```

ตัวอย่างโปรแกรมที่ 3.9 เป็นโปรแกรมที่แสดงการใช้ #flag โดยจะเติม 0 หน้าจำนวนเลขฐานแปด และเติม Ox หรือ OX หน้าจำนวนเลขฐานสิบหก และบังคับให้มีจุดทศนิยมบนจำนวนที่พิมพ์ด้วย g

```
/* Using the # flag with conversion specifiers
   o, x, X and any floating–point specifier */
#include <stdio.h>

int main ()
{
    int c = 1427 ;          /* initialize c */
    double p = 1427.0 ;   /* initialize p */

    printf ( "%#o\n", c ) ;
    printf ( "%#x\n", c ) ;
    printf ( "%#X\n", c ) ;
    printf ( "\n%g\n", p ) ;
    printf ( "%#g\n", p ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
02623
0x593
0X593

1427
1427.00
```

ตัวอย่างโปรแกรมที่ 3.10 เป็นโปรแกรมที่มีการใช้ flag ผสมระหว่าง +flag และ 0 (zero) flag เพื่อพิมพ์ค่า 452 โดยจะมีเครื่องหมาย + และเลขศูนย์เติมหน้าจำนวน 452 และพิมพ์ค่า 452 โดยใช้ 0 flag

```
/* Printing with the 0 ( zero ) flag fills in leading zeros */
#include <stdio.h>

int main ()
{
    printf ( "%+09d\n", 452 ) ;
    printf ( "%09d\n", 452 ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
+00000452
000000452
```

ตัวอย่างโปรแกรมที่ 3.11 เป็นโปรแกรมที่มีการใช้ตัวระบุแบบ %s, %d และ %f โดยจะเขียนตัวระบุรูปแบบไม่ติดกัน

```
#include <stdio.h>

int main ()
{
    char item [ 10 ] = "Bandid" ;
    int no = 12345 ;
    float cost = 0.05 ;
    printf ( "%s %d %f", item, no, cost ) ;

    return 0 ;

}
```

```
Bandid   12345   0.050000
```
← ผลลัพธ์จะมีช่องว่างระหว่างข้อมูล

แต่ถ้าโปรแกรมที่ 3.12 เราเขียนคำสั่ง printf เป็นดังนี้

```
printf ( "%s%d%f", item, no, cost ) ;
```

จะได้ผลลัพธ์ดังนี้

```
Bandid123450.050000
```
← ผลลัพธ์จะชิดกันหมด

3.7 การนำข้อมูลเข้าโดยใช้ฟังก์ชัน scanf

ในภาษา C ถ้าเราต้องการป้อนข้อมูลทางแป้นพิมพ์ เราสามารถใช้ฟังก์ชัน scanf โดยรูปแบบทั่วไปของฟังก์ชัน scanf มีดังนี้

scanf (format-control-string, other-arguments) ;

โดย format-control-string จะอธิบายถึงรูปแบบของข้อมูลในการนำเข้า ซึ่ง format-control-string อาจจะมีตัวระบุรูปแบบ 1 ตัว หรือมากกว่าก็ได้ ส่วน other-arguments จะประกอบด้วยที่อยู่ของตัวแปร 1 ตัว หรือมากกว่าก็ได้ ที่สมนัยกับตัวระบุรูปแบบใน format-control-string ถ้ามีตัวแปรมากกว่า 1 ตัว จะใช้เครื่องหมาย จุลภาค (,) คั่นระหว่าง ตัวแปรแต่ละตัว โดยตัวแปรแต่ละในส่วนของ other-arguments จะต้องมีอักขระ ampersand (&) เติมหน้า ซึ่งจะหมายถึง ตัวดำเนินการที่อยู่ (address operator) โดยถ้าตัวแปรมีข้อมูลพื้นฐานชนิด int, double และ char จะต้องมีการใช้ตัวดำเนินการที่อยู่ และจัดหาที่อยู่ให้กับตัวแปรในส่วน other-arguments ถ้าตัวแปรมีข้อมูลเป็นแบบสายอักขระ ไม่ต้องมีอักขระ ampersand นำหน้า เนื่องจากชื่อของตัวแปรสายอักขระจะชี้ไปยังตำแหน่งที่อยู่ในหน่วยความจำอยู่แล้ว

ตัวอย่างเช่น กำหนด int n ;

scanf (“%d”, &n) ;

เมื่อโปรแกรมประมวลผลคำสั่ง scanf โปรแกรมจะหยุดการประมวลผลชั่วคราว โดยจะรอให้ผู้ใช้ป้อนข้อมูลทางแป้นพิมพ์ให้กับตัวแปร n โดยเราจะต้องป้อนข้อมูลให้สอดคล้องกับรูปแบบชนิดของข้อมูลด้วย ในตัวอย่างนี้ ใช้รูปแบบ %d แสดงว่า เวลาป้อนข้อมูลเราจะต้องป้อนข้อมูลจำนวนเต็มด้วย โดยข้อมูลที่เราป้อนทางแป้นพิมพ์จะถูกส่งไปให้ตัวแปร n เมื่อเราได้กด enter ซึ่งจะทำให้โปรแกรมสามารถประมวลผลต่อไปได้ด้วย ถ้าเราป้อนข้อมูลไม่ตรงกับชนิดรูปแบบที่กำหนดไว้ เช่น ถ้าเราป้อนข้อมูลเป็น 125.56 ซึ่งเป็นข้อมูลแบบทศนิยม ก็จะถูกตัดออกเป็น 125 และกำหนดค่าให้กับตัวแปร n และถ้าเราป้อนข้อมูลแบบวิทยาศาสตร์ คือ $14.56e-2$ ก็จะถูกตัดออกเป็น 14

ตารางที่ 3.4 แสดงตัวระบุการแปลงผันที่ใช้ในการป้อนข้อมูล

ตัวระบุการแปลงผัน	ความหมาย
จำนวนเต็ม (integer)	
d	อ่านค่าจำนวนเต็มฐานสิบมีเครื่องหมาย ที่สมนัยกับอาร์กิวเมนต์ที่มีตัวชี้ (pointer) ไปยังจำนวนเต็ม
i	อ่านค่าจำนวนเต็มฐานสิบ, ฐานแปด หรือฐานสิบหก มีเครื่องหมายที่สมนัยกับอากิวเมนต์ ที่มีตัวชี้ไปยังจำนวนเต็ม
o	อ่านค่าจำนวนเต็มฐานแปด ที่สมนัยกับอากิวเมนต์ ที่มีตัวชี้ไปยังจำนวนเต็มไม่มีเครื่องหมาย
u	อ่านค่าจำนวนเต็มฐานสิบไม่มีเครื่องหมาย ที่สมนัยกับอากิวเมนต์ที่มีตัวชี้ไปยังจำนวนเต็มไม่มีเครื่องหมาย
x หรือ X	อ่านค่าจำนวนเต็มฐานสิบหก ที่สมนัยกับอากิวเมนต์ ที่มีตัวชี้ไปยังจำนวนเต็มไม่มีเครื่องหมาย
h หรือ l	อ่านค่าจำนวนเต็มแบบ short หรือ แบบ long
จำนวนจุดลอยตัว (floating–point number)	
e, E, f, g หรือ G	อ่านค่าจำนวนจุดลอยตัวที่สมนัยกับอาร์กิวเมนต์ ที่มีตัวชี้ไปยังค่าจำนวนจุดลอยตัว
l หรือ L	อ่านค่าจำนวนจุดลอยตัวแบบ double หรือ long double ที่สมนัยกับอาร์กิวเมนต์ ที่มีตัวชี้ไปยังตัวแปรแบบ double หรือ long double
ตัวอักขระและสายอักขระ (characters and strings)	
c	อ่านตัวอักขระที่สมนัยกับอาร์กิวเมนต์ ที่มีตัวชี้ไปยังตัวแปรแบบ char
s	อ่านสายอักขระที่สมนัยกับอาร์กิวเมนต์ ที่มีตัวชี้ไปยังแถวลำดับชนิด char
Scan set [scan characters]	scan สายอักขระสำหรับเซตของอักขระที่จะนำไปเก็บไว้ในแถวลำดับ

ตัวอย่างโปรแกรมที่ 3.12 เป็นโปรแกรมที่อ่านค่าจำนวนเต็มที่มีตัวระบุการแปลงผันจำนวนเต็มหลายแบบ และแสดงผลลัพธ์ โดย %i สามารถรับค่าข้อมูลจำนวนเต็มฐานสิบ, ฐานแปด และฐานสิบหก

```
/* Reading integers */
#include <stdio.h>

int main ()
{
    int a ;
    int b ;
    int c ;
    int d ;
    int e ;
    int f ;
    int g ;

    printf ( "Enter seven integers :" ) ;
    scanf ( "%d%i%i%i%o%u%x", &a, &b, &c, &d, &e, &f, &g ) ;

    printf ( "The input displayed as decimal integers is :\n" ) ;
    printf ( "%d %d %d %d %d %d\n", a, b, c, d, e, f, g ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
Enter seven integers ; –70  –70  070  0x70  70  70  70
The input displayed as decimal integers is :
–70  –70  56  112  56  70  112
```

ถ้าเราต้องการป้อนข้อมูลแบบจำนวนจุดลอยตัว เราจะต้องกำหนดตัวระบุการแปลงผันจุดลอยตัว เช่น e, E, f, g หรือ G นำมาใช้

ตัวอย่างโปรแกรมที่ 3.13 เป็นโปรแกรมที่อ่านค่าจำนวนจุดลอยตัวมา 3 จำนวน โดยการใช้ตัวระบุการแปลงผันจุดลอยตัว 3 แบบ และแสดงผลลัพธ์ด้วยตัวระบุการแปลงผันแบบ f

```
/* Reading floating–point numbers */
#include <stdio.h>

int main ()
{
    double a ;
    double b ;
    double c ;

    printf ( “Enter three floating–point numbers :\n” ) ;
    scanf ( “%1e%1f%1g”, &a, &b, &c ) ;

    printf ( “Here are the numbers entered in plain\n” ) ;
    printf ( “floating–point notation:\n” ) ;
    printf ( “%f\n%f\n%f\n”, a, b, c, ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
Enter three floating–point number :
1.27987  1.27987e+03  3.38476e–06
Here are the numbers entered in plain
floating–point notation :
1.279870
1279.870000
0.000003
```

ถ้าเราต้องการป้อนข้อมูลแบบตัวอักขระและสายอักขระ เราจะต้องกำหนดตัวระบุการแปลงผัน c และ s ตามลำดับ

ตัวอย่างโปรแกรมที่ 3.14 เป็นโปรแกรมที่จะมีข้อความพร้อมรับ (prompts) ให้ผู้ใช้ป้อนข้อมูลแบบสายอักขระทางแป้นพิมพ์ โดยใช้โปรแกรมนี้ จะมีการนำตัวอักขระตัวแรกไปไว้ในตัวแปรอักขระ x เนื่องจากได้กำหนดตัวระบุการแปลงผันเป็น %c และนำสายอักขระที่เหลือไปไว้ในตัวแปรแถวลำดับอักขระ y เนื่องจากได้กำหนดตัวระบุการแปลงผันเป็น %s

```
/* Reading characters and strings */
#include <stdio.h>

int main ()
{
    char x ;
    char y[ 9 ] ;

    printf ( "Enter a string : " ) ;
    scanf ( "%c%s", &x, y ) ;

    printf ( "The input was :\n" ) ;
    printf ( "the character \"%c\" ", x ) ;
    printf ( "and the string \"%s\"\n", y ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
Enter a string :  Sunday
The input was :
the character "S" and the string "unday"
```

ถ้าเราต้องการป้อนข้อมูลแบบตัวอักขระต่อเนื่องกันไป เราสามารถใช้ scan set โดย scan set คือเซตของตัวอักขระที่ปิดล้อมด้วยเครื่องหมาย [ ] และมีเครื่องหมาย % นำหน้า โดย scan set จะกราดตรวจ (scan) ตัวอักขระที่ถูกป้อนเข้ามาว่าอยู่ใน scan set หรือไม่ ถ้าอยู่ก็จะถูกเก็บไว้ในตัวแปรแถวลำดับอักขระ และถ้าตัวอักขระที่ถูกอ่านนั้นไม่อยู่ scan set ก็จะยุติการอ่านและเพิ่มตัวอักขระ null ('\o') ปิดท้ายแถวลำดับอักขระ

ตัวอย่างโปรแกรมที่ 3.15 เป็นโปรแกรมที่มีการใช้ scan set [aeiou] ในการกราดตรวจ (scan) ข้อมูลที่ถูกป้อนเข้ามา ถ้าตัวอักขระที่ถูกอ่านมาอยู่ใน scan set จะเก็บไว้ในตัวแปรแถวลำดับอักขระ ถ้าไม่อยู่ก็จะยุติการ scan

```
/* Using a scan set */
#include <stdio.h>

/* function main begins program execution */
int main ()
{
    char z[ 9 ] ; /* define array z */

    printf ( "Enter string : " ) ;
    scanf ( "%[aeiou]", z ) ;  /* search for set of characters */

    printf ( "The input was \"%s\"\n", z ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
Enter a string : ooeeooahah
The input was "ooeeooa"
```

เราสามารถใช้ในการกราดตรวจ (scan) ตัวอักขระที่ไม่อยู่ใน scan set โดยการใช้ inverted scan set การสร้าง inverted scan set จะแทนด้วยเครื่องหมาย caret (^) ในเครื่องหมาย [ ] ก่อนตัวอักขระที่ถูก scan

ตัวอย่างโปรแกรมที่ 3.16 เป็นโปรแกรมที่มีการใช้ inverted scan set [^aeiou]

```
/* Using an inverted scan set */
#include <stdio.h>

int main ()
{
    char z[ 9 ] ;

    printf ( "Enter a string : " ) ;
    scanf ( "%[^aeiou]", z ) ;  /* inverted scan set */

    printf ( "The input was \"%s\"\n", z ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
Enter a string : String
The input was "Str"
```

เราสามารถระบุความกว้างของฟิลด์ในฟังก์ชัน scanf ได้

ตัวอย่างโปรแกรมที่ 3.17 เป็นโปรแกรมที่ป้อนข้อมูลเข้ามาต่อเนื่องโดยตัวเลข 2 ตัวแรก จะกำหนดให้กับตัวแปร x และส่วนที่เหลือเป็นของตัวแปร y

```
/* inputting data with a field width */
#include <stdio.h>

int main ()
{
    int x ;
    int y ;

    printf ( "Enter a six digit integer : " ) ;
    scanf ( "%2d%d", &x, &y ) ;

    printf ( "The integers input were %d and %d\n", x, y ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
Enter a six digit integer :  123456
The integers input were 12 and 3456
```

บางครั้งเราอาจจะมีการป้อนข้อมูลในรูป เดือน–วัน–ปี เช่น 11–10–1999 โดยตัวเลขจะถูกเก็บไว้ในตัวแปร ส่วนเครื่องหมาย – จะถูกขจัดออก เราสามารถใช้คำสั่ง scanf ดังนี้

```
scanf ( " %d–%d–%d", &day, &year ) ;
```

ซึ่งคำสั่งดังกล่าวสามารถขจัดเครื่องหมาย – ออกไปได้ แต่บางครั้งอาจจะมีการป้อนข้อมูลในรูป เดือน/วัน/ปี เช่น 10/11/1999 คำสั่ง scanf ก่อนหน้านี้จะไม่สามารถขจัดเครื่องหมาย / ออกไปได้ ดังนั้นในภาษา C จะใช้เครื่องหมาย * แทน ซึ่งเราจะเรียกเครื่องหมาย * นี้ว่า assignment suppression character ซึ่งจะเป็นการอ่านข้อมูลแบบใด ๆ จากข้อมูลที่ถูกป้อนเข้ามา แล้วขจัดอักขระที่ไม่จำเป็นออกไป

ตัวอย่างโปรแกรมที่ 3.18 เป็นโปรแกรมที่มีการใช้ assignment suppression character ใน %C เพื่อใช้ขจัดเครื่องหมายที่ไม่จำเป็นออกไป เฉพาะค่าของ month, day และ year จะถูกเก็บไว้

```
/* Reading and discarding characters from the input stream */
#include <stdio.h>

int main ()
{
    int month1 ;
    int day1 ;
    int year1 ;
    int month2 ;
    int day2 ;
    int year2 ;

    printf ( "Enter a date in the form mm–dd–yyyy : " ) ;
    scanf ( "%d%*c%d%*c%d", &month1, &day1, &year1 ) ;

    printf ( "month = %d   day = %d   year = %d\n\n", month1, day1, yera1 ) ;

    printf ( "Enter a date in the form mm/dd/yyyy : " ) ;
    scanf ( "%d%*c%d%*c%d", &month2, &day2, &year2 ) ;

    printf ( "month = %d   day = %d   year = %d\n\n", month2, day2, yera2 ) ;

    return 0 ;  /* indicates successful termination */

}   /* end main */
```

```
Enter a date in the form mm–dd–yyyy : 11–18–2003
month = 11   day = 18   year = 2003

Enter a date in the form mm/dd/yyyy : 11–18–2003
month = 11   day = 18   year = 2003
```

การป้อนข้อมูลโดยใช้ฟังก์ชัน scanf มีข้อควรระวัง ดังตัวอย่างต่อไปนี้

ตัวอย่างโปรแกรมที่ 3.19

```
#include <stdio.h>
int main ()
{
    int a, b, c ;
        ⋮
    scanf ( "%3d %3d  %3d", &a, &b, &c ) ;
        ⋮
    return 0 ;
}
```

ถ้าเราป้อนข้อมูลดังนี้

1 2 3

จะได้ผลดังนี้ โดย

a = 1 , b = 2 , c = 3

แต่ถ้าเราป้อนข้อมูลดังนี้

123456789

จะได้ผลดังนี้ โดย

a = 123 , b = 456 , c = 789

เนื่องจากตัวแปรแต่ละตัวกำหนดความกว้างฟิลด์เท่ากับ 3 และถ้าเราป้อนข้อมูลดังนี้

1234 5678 9

จะได้ผลดังนี้ โดย

a = 123 , b = 4 , c = 567

เนื่องจากตัวแปรกำหนดความกว้างฟิลด์ เท่ากับ 3 ตัวเลข 8 และ 9 จะถูกเพิกเฉย

ตัวอย่างโปรแกรมที่ 3.20

```
#include <stdio.h>
int main ()
{
    int i ;
    flat x ;
    char c ;
       ⋮
    scanf ( “%3d %5f  %c”, &i, &x, &c ) ;
       ⋮
    return 0 ;
}
```

ถ้าเราป้อนข้อมูลดังนี้

10 256.875 T

จะได้ผลดังนี้

i = 10 , x = 256.8 , c = ‘7’ ส่วนที่เหลือคือ 5 และ T จะถูกเพิกเฉย

ตัวอย่างโปรแกรมที่ 3.21

```
#include <stdio.h>
int main ()
{
    char c1, c2, c3 ;
        ⋮
    scanf ( “%c%c%c”, &c1, &c2, &c3 ) ;
        ⋮
    return 0 ;
}
```

ถ้าเราป้อนข้อมูลดังนี้

a b c

จะได้ผลดังนี้ โดย

c1 = a , c2 = <blank space> , c3 = b

แต่ถ้าเราเขียนฟังก์ชัน scanf แบบนี้

scanf (“%c%1s%1s”, &c1, &c2, &c3) ;

โดยป้อนข้อมูลในลักษณะเช่นเดิม

จะได้ผลดังนี้ โดย

c1 = a , c2 = b , c3 = c

แต่ขอแนะนำว่า ควรเขียนคำสั่งฟังก์ชัน scanf ดังนี้

scanf (“%c %c %c”, &c1, &c2, &c3) ;

โดยมีช่องว่างระหว่าง %c ก็จะให้ผลเหมือนกัน