บทที่ 2 ข้อมูลพื้นฐาน (Basic Data Types)

ตัวแปรทุกตัวที่มีการกำหนดใช้ในการเขียนโปรแกรมคอมพิวเตอร์ มีลักษณะเบื้องต้นเหมือนกันสองประการ กล่าวคือ ชื่อของตัวแปร เพื่อบอกถึงชนิดของข้อมูลที่จะนำมาเก็บในตัวแปร เนื้อหาภายในบทนี้จะกล่าวถึง การตั้งชื่อตัวระบุ(identifier)ในภาษา C ประเภทของข้อมูล ได้แก่ จำนวนเต็ม จำนวนจริง และอักขระ นอกจากนี้จะกล่าวถึงขอบเขตของค่าข้อมูลที่เก็บไว้ในตัวแปรแต่ละประเภท

2.1 ตัวระบุและกฎการตั้งชื่อ

ตัวระบุในที่นี้จะหมายถึง ชื่อตัวแปรและชื่อของฟังก์ชัน ซึ่งมีกฎการตั้งชื่อดังต่อไปนี้

1) ชื่อที่ตั้งประกอบด้วยตัวอักษรภาษาอังกฤษ ซึ่งอาจเป็นตัวอักษรพิมพ์เล็ก ตัวอักษรพิมพ์ใหญ่ ตัวเลข และ _(underscore) โดยมีเงื่อนไขว่าตัวแรกจะต้องเป็นตัวอักษรเท่านั้น
2) การตั้งชื่อตัวระบุมีความยาวเท่าไรก็ได้ แต่ภาษา C จะนับตัวอักขระแปดตัวแรกเป็นสำคัญ โดยกฎนี้ทำให้ emp_name1 และ emp_name2 ถือเป็นตัวระบุตัวเดียวกัน
3) ห้ามนำศัพท์สงวน (ศัพท์สงวนทุกตัวจะใช้ตัวพิมพ์เล็กเท่านั้น) มาตั้งเป็นชื่อตัวระบุ
4) การตั้งชื่อตัวระบุจะให้ความสำคัญกับชนิดของตัวอักษรว่าเป็นตัวพิมพ์ใหญ่หรือเล็ก ตัวอย่างเช่น emp_name Emp_name และ Emp_Name ตัวระบุทั้งสามถือเป็นตัวระบุคนละตัว

ข้อแนะนำ การตั้งชื่อตัวระบุควรตั้งชื่อให้สื่อกับวัตถุประสงค์ของการทำงานของตัวแปร เพื่อสื่อความหมายที่ตรงกัน

2.2 ประเภทข้อมูล

ในการประกาศตัวแปร นอกจากการระบุด้วยชื่อเป็นส่วนแรก จะต้องประกอบด้วยประเภทข้อมูลของตัวระบุ ข้อมูลพื้นฐานในภาษา C มีดังนี้

- จำนวนเต็ม(integer) เช่น 2,-5,18
- จำนวนจริง(float) เช่น 4.5,-8.3
- ตัวอักขระ(character) เช่น "a","A"

2.2.1 จำนวนเต็ม (integer)

จำนวนเต็มในภาษา C แบ่งได้ดังนี้

ประเภทของข้อมูล	จำนวนบิตที่เก็บข้อมูล	ขอบเขตของค่าของข้อมูล
int	16 หรือ 32	-32,768 ถึง 32,767 หรือ –2,147,483,648 ถึง 2,147,483,647
short	16	-32,768 ถึง 32,767
long	32	-2,147,483,648 ถึง 2,147,483,647
unsigned int	16 หรือ 32	0 ถึง 65,535 หรือ 0 ถึง 4,294,967,295
unsigned short	16 หรือ 32	0 ถึง 65,535 หรือ 0 ถึง 4,294,967,295
unsigned long	32	0 ถึง 4,294,967,295

ตารางที่ 2.1 ขนาดและขอบเขตของค่าของข้อมูลประเภทจำนวนเต็ม

โดยปกติ short และ long จะเก็บค่าจำนวนเต็ม ตามที่ผู้ใช้กำหนด แต่ int จะมีขนาด 16 หรือ 32 บิต ขึ้นอยู่กับชนิดของเครื่องคอมพิวเตอร์

เมื่อกำหนดประเภทข้อมูลเป็น int, short และ long จะหมายถึงจำนวนเต็มที่มีเครื่องหมายกำหนด (signed integer) ในการเก็บจำนวนเต็มเหล่านี้จะต้องใช้บิตแรกสำหรับเก็บเครื่องหมายบวกหรือลบ ส่วนบิตที่เหลือใช้เก็บค่าของข้อมูล สำหรับจำนวนเต็มที่ไม่ระบุเครื่องหมาย (unsigned integer)ได้แก่ unsigned int, unsigned short และ unsigned long จะเก็บข้อมูลที่มีค่าเป็นบวกเท่านั้น ในกรณีนี้ทำให้สามารถใช้เนื้อที่ทั้งหมดในการเก็บค่าของข้อมูล ซึ่งทำให้เก็บข้อมูลที่มีค่ามากขึ้นได้

2.2.2 จำนวนจริง (real)

จำนวนจริงแบ่งออกเป็น 2 ประเภท คือ float และ double การจัดเก็บจำนวนจริงจะต้องแปลงจำนวนจริงให้อยู่ในรูปของ $Ax10^n$ โดยที่ A <=1 ซึ่ง A เป็นจำนวนเต็ม เราเรียก A ว่า fraction และ เรียก n ว่า exponent

ลักษณะการจัดเก็บจำนวนจริง float ใช้เนื้อที่ทั้งหมด 32 บิต โดยแบ่งออกเป็นสามส่วน คือส่วนที่เป็น fraction ใช้เนื้อที่ 23 บิต ส่วนที่เป็น exponent ใช้เนื้อที่ 9 บิต และ เครื่องหมายและค่าของ exponent 1 บิต

โดยแสดงการเก็บข้อมูลจำนวนจริง แสดงได้ดังนี้

บิตที่ 0	บิตที่ 1…8	บิตที่ 9…31
เครื่องหมาย	exponent	fraction

ยกตัวอย่างเช่น

ลำดับที่	Floating Point	สัญลักษณ์ทางวิทยาศาสตร์	บิตเครื่องหมาย	exponent	fraction
1	12.45	1.245×10^{1}	0	1	1.245
2	-211.0	-2.110×10^{2}	1	2	2.11
3	0.0056	5.600×10^{-3}	0	-3	5.6

สำหรับการจัดเก็บจำนวนจริง double จะใช้เนื้อที่ในการจัดเก็บทั้งหมด 64 บิต ทำให้สามารถจัดเก็บจำนวนจริงที่มีค่ามากขึ้น และมีค่านัยสำคัญประมาณ 14 ตำแหน่ง ในขณะที่จำนวนจริง float มีค่านัยสำคัญ 7 ตำแหน่ง

2.2.3 การอ่านและการแสดงผลจำนวนในภาษา C

ในการรับรูปแบบของตัวแปรที่มีประเภทข้อมูลต่างๆ กัน เช่น จำนวนเต็มจะระบุด้วย %d และจำนวนจริงจะระบุด้วย %f โดยมีรายละเอียดตามตารางที่ 2.2

ประเภทข้อมูล	Format String สำหรับคำสั่ง scanf	Format String สำหรับคำสั่ง printf
short	%hd	%d
int	%d	%d
long	%ld	%ld
unsigned short	%hu	%u
unsigned int	%u	%u
unsigned long	%lu	%lu
octal short	%ho	%o
octal int	%o	%o
octal long	%lo	%lo
hex short	%hx	%x
hex int	%x	%x
hex long	%lx	%lx
float	%f	%f
double	%lf	%f

ตารางที่ 2.2 รูปแบบประเภทข้อมูลในการอ่านและแสดงผล

ซึ่ง %d จะสามารถแทนจำนวนเต็มทั้งจำนวน และ %f สามารถพิมพ์ค่าทศนิยมที่มีเลขหลังจุดได้ถึง 6 ตำแหน่ง และเลขหน้าจุดทศนิยมอีกจำนวนหนึ่ง แต่บางครั้งเราพบการระบุจำนวนระหว่างรูปแบบ เช่น %9.2f หมายความว่า เรากำลังบอกกับเครื่องคอมพิวเตอร์ว่าเราต้องการผลลัพธ์เป็นเลขทศนิยม ซึ่งมีขนาด 9 หลัก เป็นเลขหลังจุดทศนิยม 2 ตำแหน่ง

ตัวอย่างที่ 2.1 โปรแกรมสำหรับการแสดงผล ค่า x และ y ที่เป็นข้อมูลประเภทจำนวนเต็ม

```
#include <stdio.h>

main()
{
        int x,y;
        x=5;
        y=x;
        x=8;
        printf ("The value of x = \t%d\n" , x);
        printf ("The value of y = \t%d\n" , y);
        return 0;
} /* End main */
```

ผลลัพธ์ที่ได้

```
The value of x =        8
The value of y =        5
```

ตัวอย่างที่ 2.2 โปรแกรมคำนวณพื้นที่สี่เหลี่ยมผืนผ้า

```
#include <stdio.h>

main()
{
        double width,length,area;
        printf("Enter width \t=");
        scanf("%lf",&width);
        printf("Enter length \t=");
        scanf("%lf",&length);
        area = width * length;
        printf ("Area of this rectangle : \t=%f\n", area);
        printf ("Area of this rectangle in exponential notation : \t=%e\n", area);
        return 0;
}/* End main */
```

ผลลัพธ์ที่ได้

```
Enter width     =75.2
Enter length    =128.9
Area of this rectangle :        =9693.280000
Area of this rectangle in exponential notation :        =9.69328e+03
```

2.2.4 อักขระ (character)

อักขระ หมายถึง ตัวอักษร ตัวเลข และตัวอักษรพิเศษ การจัดเก็บตัวอักขระในเครื่องหมายคอมพิวเตอร์จะจัดเก็บในรูปของรหัสแอสกี (ASCII)และเอ็บซิดิก (EBCDIC) ขึ้นอยู่กับชนิดของคอมพิวเตอร์ รูปแบบข้อมูลประเภทตัวอักขระสำหรับคำสั่ง scanf และ printf คือ %c

2.2.5 อักขระหลีก (escape character)

อักขระหลีกใช้สำหรับกำหนดรูปแบบการแสดงผล ซึ่งประกอบไปด้วยเครื่องหมาย \ (blackslash) แล้วตามด้วยตัวอักขระที่แสดงในตารางที่ 2.3 วิธีใช้อักขระหลีกคือจะแทนอักขระหลีกที่ต้องการไว้เป็นส่วนหนึ่งของ format string ในคำสั่ง prinf

อักขระ	ความหมาย
\a	แสดงเสียง
\b	ถอยหลัง
\f	Form feed
\n	ขึ้นบรรทัดใหม่
\r	ปัดแคร่
\t	ตั้งระยะแนวนอน
\v	ตั้งระยะแนวตั้ง
\\	Blackslash
\?	เครื่องหมายคำถาม
\’	ฝนทอง
\”	ฟันหนู
\0	ค่าว่าง(null)
\ddd	เลขฐานแปด
\xddd	เลขฐานสิบหก

ตารางที่ 2.3 อักขระหลีกที่ใช้กับคำสั่งแสดงผล

เนื่องจากคำสั่ง printf ไม่ทำการขึ้นบรรทัดใหม่ให้อย่างอัตโนมัติ การทำงานในการจัดรูปแบบการพิมพ์จึงต้องใช้อักขระหลีก (escape character) เช่น \t (tab) เพื่อทำการเลื่อนไป 1 ช่วงกั้น หรือ \n (newline) เพื่อส่งให้ข้อความถัดไปขึ้นบรรทัดใหม่ เป็นต้น

อนึ่ง จะเห็นว่าอักขระใน control string มีบางตัวที่เราอาจจำเป็นต้องใช้ เช่นเครื่องหมาย % ในการนี้เราสามารถระบุให้คอมไพเลอร์ มีการเข้าใจว่าอักขระดังกล่าวว่าไม่ใช่อักขระที่ระบุประเภทข้อมูลดังที่กล่าวไว้ข้างต้น ด้วยการระบุเครื่องหมาย " \ " (Backslash) นำหน้าอักขระ % เช่น

```
printf("Interest Rate \t:\t%4.2f\%\n",INTEREST);
```

ด้วยวิธีการนี้จะทำให้ \% มีความหมายเป็นการสั่งพิมพ์อักขระ % ในผลลัพธ์

ตัวอย่างที่ 2.3 โปรแกรมแสดงการเก็บค่าของตัวอักขระและจำนวนเต็ม

```
#include <stdio.h>

#define NUMCHAR 75
main()
{
        unsigned char CHAR;
        printf("\ndecimal\t|\toctal\t|\tcharacter");
        for (CHAR=60; CHAR <= NUMCHAR ; CHAR=CHAR+1)
        printf("\n%5d\t|\t%5o\t|\t%c", CHAR,CHAR,CHAR);
        return 0;
}/* End main */
```

ผลลัพธ์ที่ได้

```
decimal |       octal |       character
   60   |        74   |       <
   61   |        75   |       =
   62   |        76   |       >
   63   |        77   |       ?
   64   |       100   |       @
   65   |       101   |       A
   66   |       102   |       B
   67   |       103   |       C
   68   |       104   |       D
   69   |       105   |       E
   70   |       106   |       F
   71   |       107   |       G
   72   |       110   |       H
   73   |       111   |       I
   74   |       112   |       J
   75   |       113   |       K
```

จากตัวอย่างจะเห็นว่า ข้อมูลประเภทตัวอักขระได้ถูกจัดเก็บเป็นจำนวนเต็มซึ่งสามารถนำมาดำเนินการทางคณิตศาสตร์ได้

2.3 การอ่านและการแสดงผลข้อมูลประเภทอักขระ

ลักษณะของข้อความ (text) โดยทั่ว ๆ ไปจะประกอบด้วยตัวอักษรมาเรียงกัน ในภาษา C มีฟังก์ชันที่ทำการสนับสนุนการจัดการกับตัวอักขระทีละตัว ได้แก่ getchar และ putchar

โดยที่ getchar จะทำหน้าที่อ่านอักขระตัวถัดไปในสายอักขระ จนกระทั่งพบค่า '-1' ซึ่งถือเป็นจุดสิ้นสุดของแฟ้มข้อมูล (End Of File:EOF)

putchar ทำหน้าที่แสดงผลอักขระในข้อความทีละตัว

ตัวอย่างที่ 2.4 โปรแกรมแสดงข้อความที่รับเข้า

```
#include <stdio.h>

main()
{
      int Char;
      Char = getchar();
      while (Char != EOF) /* != หมายถึง ไม่เท่ากับ */
      {
            putchar(Char);
            Char = getchar();
      }/* End while */
      return 0;
}/* End main */
```

ผลลัพธ์ที่ได้

```
It is better to ask some of the questions then
It is better to ask some of the questions then
to know all the answer. -James Thurber
to know all the answer. -James Thurber
```

ตัวอย่างที่ 2.5 โปรแกรมแสดงข้อความที่รับเข้าโดยแสดงเลขที่บรรทัด

```
#include <stdio.h>
main()
{
        int Char,LastChar,LineNo;
        char e;
        e= '9';
        printf("\n\tLineNo\n");
        LineNo=0;
        for(LastChar = '\n'; ((Char = getchar()) != e) ; LastChar = Char)
        {
                if (LastChar == '\n')
                {
                        LineNo = LineNo + 1;
                        printf ("\t%6d\t ", LineNo);
                }/* End if */
                putchar(Char);
        }/* End for */
        return 0;
}/* End main */
```

ผลลัพธ์ที่ได้

```
        LineNo
It is better to ask some of the questions
             1   It is better to ask some of the questions
than to know all the answer. -James Thurber
             2   than to know all the answer. -James Thurber
9
        Number of Lines =      2
```

2.4 ฟังก์ชันในการทดสอบตัวอักขระ

ภาษา C มีฟังก์ชันเพื่อใช้ในการทดสอบตัวอักขระ ซึ่งจัดเก็บในคลังคำสั่งในแฟ้ม ctype.h ซึ่งจะต้องเรียกใช้ด้วยคำสั่ง #include <ctype.h> ไว้ที่ตอนต้นของโปรแกรมรายละเอียดของฟังก์ชัน ดังตารางที่ 2.4

ชื่อฟังก์ชัน	ความหมาย
isalpha	ทดสอบว่าเป็นอักษรภาษาอังกฤษ (a-z, A-Z)หรือไม่
islower	ทดสอบว่าเป็นอักษรภาษาอังกฤษตัวพิมพ์เล็ก (a-z) หรือไม่
isupper	ทดสอบว่าเป็นอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ (A-Z) หรือไม่
isalnum	ทดสอบว่าเป็นอักษรภาษาอังกฤษ ตัวเลข (a-z, A-Z,0-9) หรือไม่
isdigit	ทดสอบว่าเป็น ตัวเลข (0-9) หรือไม่
isxdigit	ทดสอบว่าเป็น ตัวเลขฐานสิบหก (0-9,a-f,A-F) หรือไม่
isspace	ทดสอบว่าเป็นที่ว่าง,\t,\v,\f ,\r หรือ \n หรือไม่
iscntrl	ทดสอบว่าเป็นอักขระควบคุม (0x00-0x1F) หรืออักขระการลบ(0x7F)หรือไม่
ispunct	ทดสอบว่าเป็นเครื่องหมายวรรคตอน หรือไม่
isgraph	ทดสอบว่าเป็นอักขระที่สามารถแสดงผลบนหน้าจอได้หรือไม่
Isascii	ทดสอบว่าเป็นอักขระแอสกี หรือไม่
Isprint	ทดสอบว่าเป็นอักขระที่สามารถพิมพ์ได้หรือไม่

ตารางที่ 2.4 รายละเอียดของฟังก์ชัน #include <ctype.h>

ในการใช้งานแต่ละฟังก์ชันจะคืนค่าใด ๆ ที่ไม่ใช่เลขศูนย์ หากการทดสอบเป็นจริง และจะคืนค่าศูนย์หากการทดสอบเป็นเท็จ เช่น

ถ้าต้องการทดสอบว่าตัวอักขระนั้นเป็นตัวพิมพ์เล็กหรือไม่อาจทำได้ดังนี้

```
if (islower(Char) != 0)
        printf (“It is a lowercase letter\n”);
```

หรือ

```
if (islower(Char))
        printf (“It is a lowercase letter\n”);
```

ตัวอย่างที่ 2.6 โปรแกรมแสดงการนับจำนวนคำของข้อมูลนำเข้า

```
#include <stdio.h>
#include <ctype.h>
/* print only Alphabetic */
#define ON 1
#define OFF 0

main()
{
int Char, Word, nc;
nc = 0;
Word = OFF;
printf("\n");
while ((Char = getchar()) != EOF)
        if (isalpha(Char) || (Word && isdigit(Char)))
        {
                putchar(Char);
                Word = ON;
        }/* End if */
        else
                if (Word)
                {
                     putchar ('\t');
                     ++nc;
                     Word = OFF;
                }/* End if */
     printf ("Number of words in this line = %d",nc);
return 0;
}/* End main */
```

ผลลัพธ์ที่ได้

```
The quick brown fox jumps over the lazy dog. 0123456789
The     quick   brown   fox     jumps   over    the     lazy    dog     ^Z
Number of words in this line = 9
```

นอกจากนี้ยังมีฟังก์ชันที่ทำการแปลงตัวอักษรภาษาอังกฤษจากตัวพิมพ์เล็กเป็นตัวพิมพ์ใหญ่หรือกลับกัน และฟังก์ชันแปลงตัวอักษรเป็นรหัสแอสกีดังนี้

ชื่อฟังก์ชัน	ความหมาย
tolower	เปลี่ยนอักษรภาษาอังกฤษ จากตัวพิมพ์ใหญ่เป็นตัวพิมพ์เล็ก (นอกนั้นไม่เปลี่ยนแปลง)
toupper	เปลี่ยนอักษรภาษาอังกฤษ จากตัวพิมพ์เล็กเป็นตัวพิมพ์ใหญ่ (นอกนั้นไม่เปลี่ยนแปลง)
toascii	เปลี่ยนจากตัวอักษรเป็นรหัสแอสกี

ตารางที่ 2.5 ฟังก์ชันในการแปลงตัวอักษรภาษาอังกฤษ

ตัวอย่างที่ 2.7 การใช้ฟังก์ชัน tolower เพื่อเปลี่ยนตัวอักษรพิมพ์ใหญ่ให้เป็นพิมพ์เล็ก (ตัวอักษรอื่น ๆ ที่ไม่ใช่ตัวพิมพ์ใหญ่จะไม่เปลี่ยนแปลง)

```
#include <stdio.h>
#include <ctype.h>
main()
{
        int Char,LastChar,LineNo,Numofline;
        char e;
        e= '9';
        printf("\n\tLineNo\n");
        LineNo=0;
        Numofline=0;
        for(LastChar = '\n'; ((Char = getchar()) != e) ; LastChar = Char)
        {
                if (LastChar == '\n')
                {
                        LineNo = LineNo + 1;
                        printf ("\t%6d\t ", LineNo);
                        ++Numofline;
                }/* End if */
                putchar( tolower(Char));
        }/* End for */
        printf ("\tNumber of Lines = %6d\t ", Numofline);
        return 0;
}/* End main */
```

ผลลัพธ์ที่ได้

```
        LineNo
The quick brown fox jumps over the lazy dog. 0123456789
             1   the quick brown fox jumps over the lazy dog. 012345678
        Number of Lines =      1
```

ตัวอย่างที่ 2.8 เปลี่ยนการใช้ฟังก์ชัน tolowerในตัวอย่างที่ 2.7 เป็น toupper

```
#include <stdio.h>
#include <ctype.h>

main()
{
        int Char,LastChar,LineNo,Numofline;
        char e;
        e= '9';
        printf("\n\tLineNo\n");
        LineNo=0;
        Numofline=0;
        for(LastChar = '\n'; ((Char = getchar()) != e) ; LastChar = Char)
        {
                if (LastChar == '\n')
                {
                        LineNo = LineNo + 1;
                        printf ("\t%6d\t ", LineNo);
                        ++Numofline;
                }/* End if */
                putchar(toupper(Char));
        }/* End for */
        printf ("\tNumber of Lines = %6d\t ", Numofline);
        return 0;
} /* End main */
```

ผลลัพธ์ที่ได้

```
        LineNo
The quick brown fox jumps over the lazy dog. 012345678
             1   THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG. 012345678
9
        Number of Lines =      1
```

ตัวอย่างที่ 2.9 โปรแกรมแปลงตัวอักษรที่รับเข้าเป็นตัวเล็กและนับจำนวนคำของข้อความ

```
#include <stdio.h>
#include <ctype.h>

#define ON 1
#define OFF 0

main()
{
int Char, Word, nc;
nc = 0;
Word = OFF;
printf("\n");
while ((Char = getchar()) != EOF)
        if (isalpha(Char) || (Word && isdigit(Char)))
        {
                putchar(tolower(Char));  /* Change to lower case */
                Word = ON;
        }/* End if */
        else
        if (Word)
                {
                        putchar ('\t');
                        ++nc;
                        Word = OFF;
                }/* End if */
        printf ("Number of words in this line = %d",nc);
return 0;
}/* End main */
```

ผลลัพธ์ที่ได้

```
The quick brown fox jumps over the lazy dog. 0123456789
the     quick   brown   fox     jumps   over    the     lazy    dog     ^Z
Number of words in this line = 9
```

แบบฝึกหัดท้ายบท

1. ชื่อตัวแปรต่อไปนี้ ชื่อใดถูกต้องตามกฎการตั้งชื่อของภาษา C ให้ยกตัวอย่างโดยการเขียนโปรแกรมซึ่งใช้ชื่อตัวแปรนั้นประกอบ

   (a) Theater

   (b) 75Road

   (c) Road*6

   (d) V#27

2. จงเขียนจำนวนต่อไปนี้ในรูปแบบของการประกาศในภาษา C

   (a) 12345

   (b) -17850

3. โปรแกรมต่อไปนี้มีที่ผิดอยู่หลายจุด ให้แก้ไขและระบุผลลัพธ์

```
#include <stdio.h>
/* exc.cpp*/
char  X,Y,Z;
main()
{
        X = 'I';
        Y = 'L'
        Z = 'U';
        put char (x) ;put char (Y); X=y;
        put char (Y) ;put char (X); put char (Z);
        putchar (\n);
        return 0;
}
```

4. ให้เขียนโปรแกรมเพื่อนับจำนวนบรรทัดทั้งหมดที่ป้อนข้อมูล

5. ให้พิมพ์เฉพาะตัวเลขที่อยู่ในประโยคนั้น ๆ