บทที่ 2 ชนิดข้อมูลและตัวดำเนินการพื้นฐาน

ในการเรียนคณิตศาสตร์ สิ่งแรกที่ต้องรู้ ก็คือ เลขชนิดต่าง ๆ และสัญลักษณ์ต่าง ๆ ที่ใช้ในการเขียนสมการ ในการเขียนโปรแกรม สิ่งแรกที่เราต้องรู้ก็คล้าย ๆ กับการเรียนคณิตศาสตร์ เนื่องจากภาษาคอมพิวเตอร์นั้นมีรากฐานมาจากคณิตศาสตร์ (สาเหตุเนื่องมาจากจุดประสงค์ที่ภาษาคอมพิวเตอร์ถูกพัฒนาขึ้นก็เพื่อการคำนวณ) กล่าวคือ เราจะต้องรู้ถึงตัวแปร ข้อมูลชนิดต่างๆ การกำหนดค่าของตัวแปร และตัวดำเนินการต่างๆ ที่ใช้ในการเขียนโปรแกรม

2.1 ตัวแปรในโปรแกรม (Variables in Program)

ตัวแปรทางคณิตศาสตร์ใช้แทนค่าที่*สามารถเปลี่ยนแปลงได้* ในทำนองเดียวกัน ตัวแปรในโปรแกรมก็หมายถึงสิ่งที่สามารถเปลี่ยนแปลงได้เช่นกัน นั่นหมายความว่า จะต้องสามารถเปลี่ยนแปลงค่าของตัวแปรได้ และสิ่งที่ทำให้สามารถเปลี่ยนแปลงค่าของตัวแปรได้ก็คือ การเก็บตัวแปรในหน่วยความจำที่ใช้เก็บข้อมูลต่างๆ ดังนั้น ตัวแปรในโปรแกรมจึงหมายถึง **หน่วยความจำที่สามารถสามารถอ้างอิงได้ด้วยชื่อตัวแปร**

ในอดีต การอ้างอิงถึงค่าในหน่วยความจำ (หรือค่าของตัวแปร) จำเป็นต้องใช้ตำแหน่งที่อยู่ ซึ่งเกิดความยุ่งยากโดยเฉพาะอย่างยิ่งในคอมพิวเตอร์ปัจจุบันที่มีขนาดหน่วยความจำหลายกิกะไบต์ ดังนั้น ในภาษาชั้นสูง จึงมีการแสดงหน่วยความจำในรูปตัวแปร แล้วทำการอ้างอิงค่าในหน่วยความจำนั้นๆด้วยการอ้างอิงถึงตัวแปรที่ใช้แสดงหน่วยความจำนั้นๆ วิธีนี้ทำให้สามารถลดความซับซ้อนของโปรแกรมเป็นอย่างมาก ส่งผลให้ประสิทธิภาพการพัฒนาโปรแกรมสูงขึ้นมาก

ในการเขียนโปรแกรมภาษา C ที่มีการใช้หน่วยความจำเพื่อเก็บค่าประกอบการคำนวณ จำเป็นต้องมีการจองเนื้อที่ในหน่วยความจำสำหรับเก็บค่าเหล่านั้น ซึ่งเรียกกันว่า *การประกาศตัวแปร (variable declaration)* อันเป็นการบอกให้โปรแกรม Compiler จองเนื้อที่หน่วยความจำให้เพียงพอกับข้อมูลที่ต้องการเก็บ อันได้แก่ ตัวอักษร จำนวนเต็ม หรือเลขทศนิยม

เมื่อโปรแกรมเริ่มทำงาน ค่าของตัวแปรจะเป็นตัวเลขที่ค้างอยู่ในหน่วยความจำสำหรับตัวแปรเหล่านั้น ซึ่งเราไม่สามารถตั้งสมมุติฐานว่าเป็น 0 ถ้าเราต้องการให้ตัวแปรมีค่าเริ่มต้นเป็น 0 เราจำเป็นต้องสั่งให้โปรแกรมกำหนดค่าของตัวแปรนั้นให้เป็น 0 ซึ่งหมายความว่า นำเลข 0 ไปเก็บในหน่วยความจำสำหรับตัวแปรนั้น ซึ่งในภาษา C ทำได้โดยการใช้ตัวดำเนินการกำหนดค่า (Assignment Operator) ยกตัวอย่างเช่น สมมุติว่า เราตั้งชื่อตัวแปรให้เป็น x แล้วต้องการกำหนดให้เป็น 0 จะต้องใช้คำสั่ง

```
x = 0;
```

เพื่อเป็นการสั่งให้คอมพิวเตอร์นำค่า 0 ไปเก็บที่หน่วยความจำสำหรับตัวแปร x หลังคอมพิวเตอร์ทำคำสั่งนี้แล้ว ค่าของตัวแปร x ก็จะเปลี่ยนเป็น 0

ในการทำงานทั่วไปของโปรแกรม เริ่มต้นก็จะทำการจองเนื้อที่ในหน่วยความจำ กำหนดค่าเริ่มต้น คำนวณหาค่าต่างๆแล้วเปลี่ยนแปลงค่าของตัวแปร ทำกระบวนการนี้ไปจนกว่าจะได้ผลลัพธ์ที่ต้องการ ตัวอย่างเช่น ถ้าต้องการหาผลรวมของเลข 10 จำนวน เริ่มต้นก็เตรียมตัวแปรสำหรับเก็บค่าผลรวม กำหนดค่าเริ่มต้นให้เป็น 0 แล้วจึงบวกเลข 10 จำนวนนั้นเข้าในตัวแปรนั้นทีละ 1 จำนวนควบคู่กับการปรับค่าของตัวแปร (อันหมายถึง การกำหนดค่าของตัวแปรให้เป็นค่าใหม่) จนครบ 10 จำนวน ก็จะได้ผลรวมของเลข 10 จำนวนที่ต้องการ

2.2 ชนิดข้อมูลพื้นฐานของภาษา C (Basic Data Types in C Language)

เนื่องจากภาษา C สร้างขึ้นเพื่อพัฒนาโปรแกรมสำหรับสั่งงานบนคอมพิวเตอร์ ดังนั้น ชนิดข้อมูลพื้นฐานของภาษา C จึงจำเป็นต้องเป็นข้อมูลที่คอมพิวเตอร์แสดงได้ ดังได้อธิบายในบทที่ 5 โดยคำศัพท์ที่ใช้ในการประกาศตัวแปรสำหรับชนิดข้อมูลพื้นฐาน มีดังนี้

- **char** ใช้ประกาศตัวแปรสำหรับเก็บข้อมูลที่เป็นตัวอักษร ขนาด 1 ไบต์
- **int** ใช้ประกาศตัวแปรสำหรับเก็บข้อมูลที่เป็นเลขจำนวนเต็มที่มีเครื่องหมาย (signed integer) กล่าวคือ จำนวนเต็มที่เป็นได้ทั้งบวกและลบ โดยการแสดงจำนวนเต็มลบจะใช้วิธี Two's complement ในคอมพิวเตอร์ส่วนใหญ่ในปัจจุบันที่ใช้โปรเซสเซอร์ขนาด 32 บิต จำนวนเต็ม 1 ตัวจะใช้เนื้อที่ 4 ไบต์ในการเก็บ นอกจากนี้ ภาษา C ยังได้เตรียมจำนวนเต็มขนาดและชนิดอื่น อันได้แก่
  - short int (หรือเขียนสั้นๆว่า short) ใช้ประกาศตัวแปรสำหรับเก็บข้อมูลจำนวนเต็มโดยใช้ความยาวสั้นกว่าความยาว default โดยทั่วไปจะใช้หน่วยความจำ 2 ไบต์
  - long int (หรือเขียนสั้นๆว่า long) ใช้ประกาศตัวแปรสำหรับเก็บข้อมูลจำนวนเต็มโดยใช้ความยาวมากกว่าความยาว default ซึ่งโดยทั่วไปจะหมายถึง หน่วยความจำ 8 ไบต์ แต่ในคอมพิวเตอร์ทั่วไปในปัจจุบัน ยังไม่มีการใช้จำนวนเต็ม 8 ไบต์ ดังนั้น long int จะไม่ให้ผลแตกต่างจาก int
  - unsigned ใช้ประกาศตัวแปรที่เก็บจำนวนเต็มที่ไม่มีเครื่องหมาย กล่าวคือ จำนวนเต็มที่มากกว่าหรือเท่ากับ 0 ซึ่งจะใช้ร่วมกับคำศัพท์อื่นข้างต้น เช่น unsigned int unsigned long เป็นต้น

ข้อพึงระวังเกี่ยวกับการใช้จำนวนเต็มคือ ช่วงของค่าที่แสดงได้จะต่างกันไปตามขนาดหน่วยความจำที่ใช้เก็บและการมีเครื่องหมายหรือไม่มี (signed or unsigned) ซึ่งได้แสดงในตารางที่ 5.5

- **float** ใช้ประกาศตัวแปรสำหรับเก็บข้อมูลที่เป็นเลขทศนิยมที่ใช้ความเที่ยงตรงเท่าเดียว (Single precision) ซึ่งใช้ 4 ไบต์ในการแสดง
- **double** ใช้ประกาศตัวแปรสำหรับเก็บข้อมูลที่เป็นเลขทศนิยมที่ใช้ความเที่ยงตรงสองเท่า (Double precision) ซึ่งใช้ 8 ไบต์ในการแสดง
- **long double** ใช้ประกาศตัวแปรสำหรับเก็บข้อมูลที่เป็นเลขทศนิยมที่ใช้ความเที่ยงตรงสองเท่าแบบยาว (Long double precision) ซึ่งใช้ 10 ไบต์ในการแสดง แต่ในคอมพิวเตอร์ส่วนใหญ่ ยังไม่สนับสนุนการใช้เลขทศนิยมรูปแบบนี้

การประกาศ (Declaration) เป็นการบอกให้โปรแกรม Compiler รู้ว่า ใช้ตัวแปรอะไรบ้าง และเป็นตัวแปรชนิดใด เพื่อที่ว่าโปรแกรม Compiler จะได้ทำการเตรียมเนื้อที่ในหน่วยความจำหลักให้เพียงพอที่จะรันโปรแกรมนั้น วิธีการประกาศโดยทั่วไปจะทำโดย

```
data type       variable_name ;  หรือ
data type       variable_name1, variable_name2, ... ;
```

ในกรณีที่ทำการประกาศตัวแปรหลาย ๆ ตัวพร้อมกัน

การตั้งชื่อตัวแปร ตัวแปรเป็นส่วนประกอบสำคัญอันหนึ่งที่จะมีผลต่อความถูกต้องของโปรแกรม และทำให้ง่ายต่อการอ่านและการตรวจสอบโปรแกรม ดังนั้น เพื่อให้การเขียนโปรแกรมเป็นไปอย่างมีประสิทธิภาพสูงสุด ในการตั้งชื่อของตัวแปร ควรยึดตามหลักเกณฑ์ต่อไปนี้

1. จะต้องไม่ใช้ Keyword เพราะอาจทำให้เกิดปัญหาในการคอมไพล์ได้
2. ไม่ควรใช้ชื่อซ้ำกับชื่อฟังก์ชัน
3. ห้ามนำหน้าด้วยตัวเลข เพราะโปรแกรม Compiler จะตีความหมายเป็นตัวเลข
4. ห้ามใช้ตัวดำเนินการ (Operator) ต่างๆอันได้แก่ +,-,*,/,% ในชื่อตัวแปร
5. ควรตั้งชื่อให้สื่อความหมายถึงค่าของตัวแปรตัวนั้น

ยกตัวอย่าง เช่น

```
int   sum ;
float radius, area, length ;
char grade ;
double gpa, class_gpa;
```

2.2 สัญพจน์ค่าคงที่ (Constant Literals)

สัญพจน์ค่าคงที่ (Constant Literals) เป็นสิ่งที่ใช้แสดงค่าคงที่ในโปรแกรม ซึ่งมีวิธีการแสดงแตกต่างกันไปตามชนิดข้อมูล โดยคอมพิวเตอร์จะทำการเก็บค่าคงที่ในหน่วยความจำ*ที่ไม่สามารถเปลี่ยนแปลงค่าได้* ขนาดของหน่วยความจำที่ใช้ในการเก็บค่าคงที่แต่ละค่าจะแตกต่างกันไปตามชนิดข้อมูล ดังต่อไปนี้

ตัวอักษรและชุดตัวอักษร (สตริง) ตัวอักษรทุกตัวจะมีขนาด 1 ไบต์ ในขณะที่หน่วยความจำสำหรับสตริงเท่ากับ n+1 ไบต์โดย n แทนจำนวนตัวอักษรในสตริง

1. ใช้เครื่องหมายคำพูด Single quote (' ') ล้อมรอบตัวอักษร เช่น 'a' '%' '#' เป็นต้น
2. ใช้เครื่องหมายคำพูด Single quote (' ') ล้อมรอบเครื่องหมาย \ และตัวเลขฐานแปด เช่น '\102' (มีค่าเท่ากับ 'B') เป็นต้น
3. ใช้เครื่องหมายคำพูด Single quote (' ') ล้อมรอบเครื่องหมาย \x และตัวเลขฐานสิบหก เช่น '\x43' (มีค่าเท่ากับ 'C') เป็นต้น
4. ใช้เครื่องหมายคำพูด Double quote (" ") ล้อมรอบชุดตัวอักษร เพื่อแสดงสตริง เช่น "Thammasat" "Engineering" เป็นต้น

จำนวนเต็ม ในเครื่องคอมพิวเตอร์ที่ใช้โปรเซสเซอร์ขนาด 32 บิต ถ้าไม่ได้ระบุเป็นพิเศษ ค่าคงที่จำนวนเต็ม 1 ค่าจะใช้หน่วยความจำ 4 ไบต์ในการเก็บ

1. ใช้เลขจำนวนเต็มฐานสิบ เช่น 10 -20 250 เป็นต้น
2. ใช้เลขฐานแปดนำหน้าด้วย 0 เช่น 020 (เท่ากับ 16) เป็นต้น
3. ใช้เลขฐานสิบหกนำหน้าด้วย 0x เช่น 0x20 (เท่ากับ 32) เป็นต้น
4. ใช้ตัวอักษร U ตามหลังเลขฐานสิบเพื่อระบุว่าเป็น ชนิด unsigned เช่น 120U เป็นต้น
5. ใช้ตัวอักษร L ตามหลังเลขฐานสิบเพื่อระบุว่าเป็น ชนิด long int เช่น 120L เป็นต้น

อนึ่ง สามารถใช้ U, L พร้อมกันได้ เช่น 120UL เป็นต้น ยิ่งไปกว่านั้น สามารถใช้กับเลขฐานแปดหรือสิบหกก็ได้ เช่น 050UL เป็นต้น โดยเลขฐานแปดและเลขฐานสิบหกจะสะดวกในการใช้แสดงตำแหน่งที่อยู่ของหน่วยความจำ หรือในกรณีที่ต้องการแสดงค่าของแต่ละบิตในหน่วยความจำ

เลขทศนิยม ถ้าไม่ได้ระบุเป็นพิเศษ เลขทศนิยม 1 ค่าจะถูกเก็บด้วยชนิดข้อมูล double ซึ่งใช้หน่วยความจำ 8 ไบต์ในการเก็บ

1. ใช้การแสดงแบบจุดทศนิยมตายตัว (Fixed-point representation) เช่น 2.5 -0.275 เป็นต้น

2. ใช้การแสดงแบบวิทยาศาสตร์ (Scientific representation) โดยแสดงอยู่ในรูปเลขยกกำลังโดยใช้ตัวอักษร e หรือ E เช่น 2.75e10 (เท่ากับ 2.75×10^{10}) -3.25E-5 (เท่ากับ -3.25×10^{-5}) เป็นต้น

3. ใช้ตัวอักษร F ตามหลังเพื่อสั่งให้เก็บเลขทศนิยมนั้นด้วยชนิดข้อมูล float ซึ่งใช้หน่วยความจำ 4 ไบต์ในการเก็บ

4. ใช้ตัวอักษร L ตามหลังเพื่อสั่งให้เก็บเลขทศนิยมนั้นด้วยชนิดข้อมูล long double ซึ่งใช้หน่วยความจำ 10 ไบต์ในการเก็บ

การกำหนดค่าเริ่มต้นของตัวแปร (Initializing Variables)

ในภาษา C เราสามารถกำหนดค่าเริ่มต้นของตัวแปรได้ในบรรทัดเดียวกับคำสั่งประกาศตัวแปร โดยใช้ตัวดำเนินการกำหนดค่า (=) แล้วตามด้วยสัญพจน์ค่าคงที่ ยกตัวอย่าง เช่น

```
char  c = 'a', d = 100;
int   counter = 0 ;
float value = 2.3 ;
```

ตัวอย่างที่ 2.1 การใช้ตัวแปรชนิด char

```
/* Usage of char type variable */
#include <stdio.h>

int main()
{
  char  a1, a2, a3, a4, a5, a6, a7, a8 ;
  a1 = 'A' ;    /* assign character 'A' to a1 */
  a2 = '\102' ;  /* assign character 'B' to a2 */
  a3 = '\x43' ;  /* assign character 'C' to a3 */
  a4 = 32 ;     /* assign character SPACE (ASCII code 32) to a4 */
  printf("Enter 3 characters\n") ;
  scanf( "%c %c %c", &a5, &a6, &a7 ) ; /* read 3 characters */
  printf( "%c%c%c%c%c%c%c\n", a1, a2, a3, a4, a5, a6, a7 ) ;
  /* Print all characters on standard output */
  return 0 ;
}
```

ผลการรันโปรแกรม

```
Enter 3 characters
xyz
ABC xyz
```

ตัวอย่างที่ 2.2 : การประกาศชนิดข้อมูลพื้นฐานและการใช้งาน

```
#include <stdio.h>
int main()
{
      char  c ;   /* Character variable */
      int   a ;   /* integer variable */
      float x ;   /* float variable */

      c = 'c' ;   /* assign alphapet 'c' to variable c. */
      a = 2 ;     /* assign 2 to variable a */
      x = 3.0 ;   /* assign 3.0 to variable x */

      printf("c = %c a = %d x = %f\n", c, a, x) ;
                  /* Notice usage of printf, %c, %d, %f */
      return 0 ;
}
```

ผลการรันโปรแกรม

```
c = c a = 2 x = 3.000000
```

2.3 ตัวดำเนินการ (Operator)

ตัวถูกดำเนินการ (Operator) คือ สัญลักษณ์หรือเครื่องหมายที่ใช้แสดงการทำงานต่างๆ อันได้แก่ การคำนวณทางคณิตศาสตร์ การเปรียบเทียบ เป็นต้น ตัวแปรหรือค่าที่ใช้ประกอบการทำงานของตัวดำเนินการมีชื่อเรียกว่า ตัวถูกดำเนินการ (Operand) เราสามารถจำแนกประเภทของตัวดำเนินการตามจำนวนตัวถูกดำเนินการดังนี้

1. ตัวดำเนินการยูนารี (Unary Operator) หมายถึง ตัวดำเนินการที่ต้องการตัวถูกดำเนินการ (Operand) เพียงตัวเดียว อันได้แก่ -, ++, -- โดยเครื่องหมาย – แทนตัวดำเนินการยูนารี – สำหรับนำหน้าค่าคงที่หรือตัวแปรเพื่อทำให้เป็นลบ เช่น -10 –x เป็นต้น
2. ตัวดำเนินการไบนารี (Binary Operator) หมายถึง ตัวดำเนินการที่ต้องการตัวถูกดำเนินการ (Operand) จำนวน 2 ตัว ซึ่งตัวดำเนินการส่วนใหญ่ในภาษา C ถูกจัดอยู่ในประเภทนี้
3. ตัวดำเนินการเทินารี (Ternary Operator) หมายถึง ตัวดำเนินการที่ต้องการตัวถูกดำเนินการ (Operand) จำนวน 3 ตัว ซึ่งมีเพียงหนึ่งเดียวคือ ตัวดำเนินการเงื่อนไข (conditional operator) หรือตัวดำเนินการ ?: (?: operator)

ตัวดำเนินการพื้นฐานในภาษา C แบ่งเป็นหมวดหมู่ตามการทำงานได้ดังนี้

1. ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operator) เป็นตัวดำเนินการที่ใช้เกี่ยวกับการคำนวณทางคณิตศาสตร์ ซึ่งจัดเป็นตัวดำเนินการไบนารี ได้แก่

+	การบวก
-	การลบ
*	การคูณ
/	การหาร
%	การหาเศษที่ได้จากการหารของจำนวนเต็ม (Modulo)

ตัวอย่าง เช่น ถ้าตัวแปร a และตัวแปร b มีค่าเท่ากับ 3 และ 2 ตามลำดับ

`a + b`	ให้ค่าเท่ากับ 5
`a * b`	ให้ค่าเท่ากับ 6
`a % b`	ให้ค่าเท่ากับ 1

2. ตัวดำเนินการกำหนดค่า (Assignment Operator) เป็นตัวดำเนินการสำหรับกำหนดค่าต่าง ๆ โดยใช้สัญลักษณ์ "=" และต้องการตัวถูกดำเนินการ 2 ตัว ตัวอย่างเช่น

```
x = 5;
```

หมายถึง กำหนดให้ค่าของตัวแปร x เท่ากับ 5 (กล่าวอีกนัยหนึ่งคือ นำ 5 ไปเก็บที่หน่วยความจำสำหรับตัวแปร x)

```
a = b;
```

หมายถึง กำหนดให้ค่าของตัวแปร a เท่ากับค่าของตัวแปร b

ข้อควรระวังในการใช้ตัวดำเนินการกำหนดค่าคือ ด้านขวามือของตัวดำเนินการนี้จะต้องให้ค่า ในขณะที่ด้านซ้ายมือจะต้องเป็นสิ่งที่เก็บค่าได้ ซึ่งหมายถึง *ตัวแปร* นั่นเอง

นอกจากนี้ ในภาษา C ยังสามารถนำตัวดำเนินการนี้ มาประกอบกับตัวดำเนินการทางคณิตศาสตร์ ซึ่งมีชื่อเรียกรวมว่า *ตัวดำเนินการผสม* (*Compound Assignment*) อันได้แก่ +=, -=, *=, /=, %=
ตัวอย่างเช่น

p += q ; มีค่าเท่ากับ p = p + q ;
p -= q ; มีค่าเท่ากับ p = p - q ;
p *= q ; มีค่าเท่ากับ p = p * q ;
p /= q ; มีค่าเท่ากับ p = p / q ;
p %= q ; มีค่าเท่ากับ p = p % q ;

3. ตัวดำเนินการ increment และ decrement ตัวดำเนินการทั้งสองนี้จัดเป็นตัวดำเนินการ unary ตัวดำเนินการ increment ใช้เครื่องหมาย ++ (เครื่องหมาย + 2 ตัวติดกัน) ใช้เพิ่มหนึ่งให้กับค่าของตัวแปรที่เป็นตัวถูกดำเนินการ ในขณะที่ตัวดำเนินการ decrement จะตรงข้ามกับตัวดำเนินการ increment โดยใช้เครื่องหมาย -- (เครื่องหมาย - 2 ตัวติดกัน) ใช้ลดหนึ่งให้กับค่าของตัวแปรที่เป็นตัวถูกดำเนินการ ดังนั้น

x++ มีค่าเท่ากับ x = x+1 มีค่าเท่ากับ x += 1
x-- มีค่าเท่ากับ x = x-1 มีค่าเท่ากับ x -= 1

ข้อสังเกตเกี่ยวกับตัวดำเนินการทั้งสองนี้ได้แก่

- ตัวดำเนินการทั้งสองนี้จัดเป็นตัวดำเนินการแบบยูนารี
- เนื่องจากการเพิ่มค่าหรือการลดค่าเป็นการเปลี่ยนแปลงค่า จึงมีการกำหนดค่ารวมอยู่ด้วย ดังนั้น ตัวถูกดำเนินการจะต้องเป็นตัวแปรเท่านั้น
- ตำแหน่งของตัวดำเนินการทั้งสองนี้จะอยู่หน้าหรือหลังตัวถูกดำเนินการก็ได้ กรณีที่อยู่หน้าตัวถูกดำเนินการจะเรียกว่า เติมหน้า (Prefix) ในขณะที่กรณีอยู่หลังตัวถูกดำเนินการ จะเรียกว่า เติมหลัง (Postfix)
- ในกรณีที่ตัวดำเนินการทั้งสองนี้ไม่ได้ถูกใช้ร่วมกับตัวดำเนินการอื่น จะเติมหน้าหรือเติมหลังก็ไม่ทำให้คำสั่งเปลี่ยนไป กล่าวคือ

  x++; มีค่าเท่ากับ ++x;
- ในกรณีที่ตัวดำเนินการทั้งสองนี้ถูกใช้ร่วมกับตัวดำเนินการอื่น ลำดับการทำงานของโปรแกรมจะเปลี่ยนไปตามตำแหน่งของตัวดำเนินการทั้งสองนี้ ยกตัวอย่างเช่น

  p *= q++ ; มีค่าเท่ากับ p = p * q ; q = q + 1 ;
  p *= ++q ; มีค่าเท่ากับ q = q + 1 ; p = p * q ;

กล่าวคือ หากเป็นการเติมหน้า ตัวดำเนินการทั้งสองนี้จะได้รับการประเมินค่า (evaluation) ก่อน แล้วจึงทำการคำนวณส่วนที่เหลือ ในขณะที่ถ้าเป็นการเติมหลัง ตัวดำเนินการทั้งสองจะได้รับการประเมินค่าหลังสุดก่อนจะเสร็จสิ้นการทำงานของคำสั่งนี้

การแปลงชนิดข้อมูล (Type Conversion) ในกรณีที่เรามีความจำเป็นต้องเปลี่ยนชนิดของข้อมูลในระหว่างการคำนวณ เราสามารถทำได้โดยใช้ตัวดำเนินการ typecast (หรือเรียกสั้นๆว่า ตัวดำเนินการ cast (cast operator)) ซึ่งมีรูปแบบสั่งงานดังนี้

วิธีใช้: `variable = (data type) expression ;`
ตัวอย่าง: `z = (int) (x + y) ;`

ตัวอย่างที่ 2.3 : การคำนวณพื้นฐานทางคณิตศาสตร์

```
#include <stdio.h>
int main()
{
       int    a, b, c ;
       float  x, y, z ;
       a = 2 ;       b = 3 ;
       c = a + b ;
       x = 2.0 ;     y = 3.0 ;
       z = x + y ;
       printf("c = %d z = %f\n", c, z) ;       /* print output */
       c++ ;                                   /* increment */
       z = (float) c ;                         /* data type conversion */
       printf("c = %d z = %f\n", c, z) ;       /* print output */
       c += b++ ;                  /* compound assignment & increment */
       printf("c = %d b = %d\n", c, b) ;
       c += ++b ;
       printf("c = %d b = %d\n", c, b) ;
       return 0 ;
}
```

ผลการรันโปรแกรม

```
c = 5 z = 5.000000
c = 6 z = 6.000000
c = 9 b = 4
c = 14 b = 5
```

4. ตัวดำเนินการความสัมพันธ์ (Relational Operator) เป็นตัวดำเนินการที่ใช้ในการตรวจสอบความสัมพันธ์หรือการเปรียบเทียบ อันประกอบด้วย

>	มากกว่า
<	น้อยกว่า
>=	มากกว่าหรือเท่ากับ
<=	น้อยกว่าหรือเท่ากับ
==	เท่ากับ
!=	ไม่เท่ากับ

โดยการประมวลผลของตัวดำเนินการเหล่านี้จะให้ผลลัพธ์ 1 ในกรณีผลการเปรียบเทียบเป็นจริง และให้ผลลัพธ์ 0 เมื่อผลการเปรียบเทียบเป็นเท็จ ยกตัวอย่าง เช่น ถ้ากำหนดให้ q = 0, r = 5 และ s = 10 แล้ว

q < s-r	ให้ผลลัพธ์	1
s/r <= q	ให้ผลลัพธ์	0
q+r == s-5	ให้ผลลัพธ์	1
q-4 == s-q+r	ให้ผลลัพธ์	0

ให้สังเกตว่า ตัวดำเนินการที่ใช้แสดง “เท่ากับ” จะใช้สัญลักษณ์ “=” 2 ตัวติดกัน ซึ่งต่างจากตัวดำเนินการกำหนดค่า ซึ่งใช้สัญลักษณ์ “=” เพียงตัวเดียว ยกตัวอย่างเช่น ถ้ากำหนดให้ a = 5, b = 10 แล้ว

a == b	ให้ผลลัพธ์	0
a = b	ให้ผลลัพธ์	10

โดยทั่วไป ตัวดำเนินการเหล่านี้จะใช้ในการกำหนดเงื่อนไขในคำสั่ง if, if-else, for, while และอื่นๆ เช่น

```
if (score >= 50) printf("Passed\n");
```

5. ตัวดำเนินการตรรกศาสตร์ (Logical Operator) เป็นตัวเนินการสำหรับการคำนวณทางตรรกศาสตร์ อันประกอบด้วย

&&	และ
||	หรือ
!	ไม่

โดยผลที่ได้จากตัวดำเนินการเหล่านี้ จะเป็นดังนี้

a	b	a && b	a || b	!a	!b
0	0	0	0	1	1
0	ไม่เท่ากับ 0	0	1	1	0
ไม่เท่ากับ 0	0	0	1	0	1
ไม่เท่ากับ 0	ไม่เท่ากับ 0	1	1	0	0

ให้สังเกตว่า

- ในภาษา C โปรแกรมจะกำหนดให้ 0 แทนค่าเท็จ ในขณะที่ค่าไม่เป็น 0 แทนค่าจริง
- ผลลัพธ์จากการคำนวณของตัวดำเนินการเหล่านี้จะเป็น 0 หรือ 1 เท่านั้น

รูปแบบการใช้งานของตัวดำเนินเหล่านี้จะเป็นดังนี้

```
expression_1 && expression_2
expression_1 || expression_2
! expression
```

ให้สังเกตว่า ตัวดำเนินการ && (AND) และ || (OR) เป็นตัวดำเนินการประเภทไบนารี ในขณะที่ตัวดำเนินการ ! (NOT) เป็นตัวดำเนินการยูนารี

ตัวอย่างเช่น ถ้ากำหนดให้ a = 6, b = 2, c = 1

(a / b == 3) && (c != b)	ให้ผลลัพธ์	1
(b < 4) || (a > 9)	ให้ผลลัพธ์	1
(c > 2) && (b < 4)	ให้ผลลัพธ์	0
! (a > 5)	ให้ผลลัพธ์	0

โดยทั่วไป ตัวดำเนินการเหล่านี้ใช้ในการกำหนดเงื่อนไขที่ซับซ้อนมากขึ้น เช่น

```
if (score >= 70 && score < 80) printf("Your grade is B\n");
```

6. ตัวดำเนินการเงื่อนไข (Conditional Operator) หรือตัวดำเนินการ ?: (?: Operator) เป็นตัวดำเนินการสำหรับแสดงเงื่อนไข ซึ่งมีความหมายเหมือนกับคำสั่ง *if/then/else* แต่ต่างจาก *if/then/else* ตรงที่ว่าตัวดำเนินการ ? : สามารถนำไปไว้เป็นส่วนหนึ่งของนิพจน์ได้ รูปแบบการใช้งานทั่วไปเป็นดังนี้

```
( condition ) ? value_when_true : value_when_false ;
```

โดย condition แสดงเงื่อนไข ในขณะที่ value_when_true และ value_when_false แสดงค่ากรณีที่เป็นเงื่อนไขเป็นจริงและเท็จตามลำดับ ยกตัวอย่าง เช่น

```
y = ( x < 3 ) ? 2 : 1 ;
printf("%s\n", ( value < 0 ) ? "minus" : "plus" ) ;
```

เราสามารถใช้ตัวดำเนินการนี้ในการกำหนดนิยามของแมคโคร (Macro) ซึ่งเป็นส่วนหนึ่งของส่วนพรีโปรเซสเซอร์ (Preprocessor) โดยตัวอย่างแมคโครต่อไปนี้จะคืนค่าน้อยกว่า ใน 2 จำนวนที่เป็นอาร์กิวเมนต์ (argument)

```
#define min(x,y) ((x < y) ? x : y )
```

อนึ่ง แมคโครเป็นค่าคงที่ หรือสัญลักษณ์ที่แทนความหมายต่าง ๆ ที่ได้รับการกำหนดนิยามในส่วนพรีโปรเซสเซอร์

2.4 ลำดับก่อนหลัง (Precedence) และคุณลักษณะเปลี่ยนหมู่ (Associativity)

โดยทั่วไป เนื่องจากเป็นไปได้ที่คำสั่งการคำนวณทางคณิตศาสตร์หรือตรรกศาสตร์จะประกอบด้วยตัวดำเนินการมากกว่า 1 ตัว จึงจำเป็นต้องจัดลำดับก่อนหลังของการประเมินค่า เพื่อให้เป็นกฎเกณฑ์สำหรับใช้ในการกำหนดลำดับการประเมินค่าของโปรแกรม โดยกฎที่เกี่ยวข้องกับการกำหนดลำดับการประเมินค่านี้ได้แก่ ลำดับก่อนหลัง (Precedence) และคุณลักษณะเปลี่ยนหมู่ (Associativity)

ลำดับก่อนหลัง (Precedence) เป็นกฎที่ใช้ในการกำหนดลำดับก่อนหลังของตัวดำเนินการต่างประเภทกัน ยกตัวอย่างเช่น x+y*z-w จะเท่ากับ x+(y*z)-w กล่าวคือ จะทำการคูณ y กับ z ก่อนแล้วจึงทำการบวกและลบ นั่นหมายความว่า ตัวดำเนินการคูณจะมีลำดับก่อนหน้าตัวดำเนินการบวกและลบ โดยกฎลำดับก่อนหลังสามารถสรุปได้ดังนี้

- ตัวดำเนินการกำหนดค่ามีลำดับหลังสุด เนื่องจากต้องทำการประเมินค่าทุกตัวดำเนินการก่อนแล้วจึงทำการกำหนดค่า
- ตัวดำเนินการยูนารีมีลำดับก่อนหน้าตัวดำเนินการไบนารี
- ตัวดำเนินการ *,/,% มีลำดับก่อนหน้าตัวดำเนินการ +,- (ตัวดำเนินการลบประเภทไบนารี)
- ตัวดำเนินการทางคณิตศาสตร์มีลำดับก่อนหน้าตัวดำเนินการความสัมพันธ์ เนื่องจากต้องทำการคำนวณหาค่าก่อน แล้วจึงทำการเปรียบเทียบ
- ตัวดำเนินการความสัมพันธ์มีลำดับก่อนหน้าตัวดำเนินการทางตรรกศาสตร์ประเภทไบนารี (&&, ||)
- ตัวดำเนินการเงื่อนไขมีลำดับตามหลังตัวดำเนินการทางตรรกศาสตร์

คุณลักษณะเปลี่ยนหมู่ (Associativity) เป็นกฎเกณฑ์ที่ใช้กำหนดลำดับก่อนหลังของตัวดำเนินการที่มีลำดับก่อนหลังเดียวกัน ยกตัวอย่างเช่น ตัวดำเนินการ +,- ตัวดำเนินการ *,/,% เป็นต้น โดยประกอบด้วย 2 กฎหลักอันได้แก่

1. ซ้ายไปขวา (Left-to-Right) หรือเปลี่ยนหมู่ซ้าย (Left-associative) ในกรณีนี้ การประเมินค่าจะทำจากตัวถูกดำเนินการที่อยู่ทางซ้ายมือก่อน แล้วจึงทำการประเมินค่าตัวถูกดำเนินการที่อยู่ทางขวามือ ตัวอย่างตัวดำเนินการที่ใช้กฎนี้ได้แก่ ตัวดำเนินการทางคณิตศาสตร์ ตัวดำเนินการทางตรรกศาสตร์ ตัวดำเนินการความสัมพันธ์ ยกตัวอย่าง เช่น a*b/c%d เท่ากับ ((a*b)/c)%d เนื่องจากโปรแกรมคอมไพเลอร์ทำการอ่านจากซ้ายไปขวา และตัวดำเนินการใช้กฎซ้ายไปขวา จึงมีลำดับการทำงานดังนี้

   (i) อ่านค่าของ a

   (ii) อ่านตัวดำเนินการ * ซึ่งใช้ a เป็นตัวถูกดำเนินการ

   (iii) อ่านค่าของ b ซึ่งเป็นตัวถูกดำเนินการของตัวดำเนินการ * จึงเป็นการประเมินค่า a*b

   (iv) อ่านตัวดำเนินการ / ซึ่งใช้ a*b เป็นตัวถูกดำเนินการ

   (v) อ่านค่าของ c ซึ่งเป็นตัวถูกดำเนินการของตัวดำเนินการ / จึงเป็นการประเมินค่า (a*b)/c

   (vi) อ่านตัวดำเนินการ % ซึ่งใช้ (a*b)/c เป็นตัวถูกดำเนินการ

   (vii) อ่านค่าของ d ซึ่งเป็นตัวถูกดำเนินการของตัวดำเนินการ % จึงเป็นการประเมินค่า ((a*b)/c)%d

2. ขวาไปซ้าย (Right-to-Left) หรือเปลี่ยนหมู่ขวา (Right-associative) ในกรณีนี้ การประเมินค่าจำทำจากตัวถูกดำเนินการทางขวามือ ไปยังตัวดำเนินการทางซ้ายมือ ตัวอย่างที่เห็นได้ชัดของตัวดำเนินการที่ใช้กฎนี้คือ ตัว

ดำเนินการกำหนดค่า เพราะต้องทำการคำนวณหาค่าก่อนแล้วจึงทำการกำหนดค่า อนึ่ง หากตัวดำเนินการทางคณิตศาสตร์ใช้กฎนี้ ตัวอย่างข้างต้น a*b/c%d จะเปลี่ยนเป็นมีค่าเท่ากับ a*(b/(c%d))

อนึ่ง ตาราง สรุปเกี่ยวกับกฎลำดับก่อนหลังและคุณลักษณะเปลี่ยนหมู่

<u>สิ่งที่ต้องคำนึงถึงในการใช้ Operator</u> *ลำดับก่อนหลัง, ชนิดข้อมูลและ ค่าของตัวแปร*

แบบฝึกหัด

1. เขียนคำสั่งสำหรับประกาศตัวแปร a, b และ c ที่เป็นชนิด int ให้อยู่ในบรรทัดเดียวกัน
2. เขียนคำสั่งสำหรับประกาศตัวแปร a, b และ c ที่เป็นชนิด char ให้อยู่ในบรรทัดเดียวกัน
3. เขียนคำสั่งสำหรับประกาศตัวแปร a, b และ c ที่เป็นชนิด int ให้อยู่ในบรรทัดเดียวกัน และทำการกำหนดค่าเริ่มต้นให้ค่า a, b และ c เท่ากับ 25, 0 และ -30 ตามลำดับ
4. จงบอกผลที่ได้จากโปรแกรมต่อไปนี้

```
#include <stdio.h>
void main()
{
      int a ;
      char c ;
      a = 90 ;
      c = 105 ;
      printf( ”%c %d %c %d \n”, a, a, c, c ) ;
}
```

5. จงแสดงเลขทศนิยมต่อไปนี้ ให้อยู่ในรูปเลขยกกำลังที่ใช้ในโปรแกรมภาษา C ได้

   a) 499123.889 c) -88237562.1400023

   b) 0.0000001234567 d) -0.00092341278

6. จงบอกผลที่ได้จากโปรแกรมต่อไปนี้

```
#include <stdio.h>
void main()
{
      float x ;
      double y ;
      char c = '\n' ;
      x = 555.23478 ;
      y = -0.0008765 ;
      printf( "%f %f %c", x, y, c ) ;
      printf( "%e %f %c", x, y, c ) ;
}
```

7. จงบอกค่าของ Expression ในข้อ a) - j) ในกรณีที่อยู่ต่อหลังส่วนของโปรแกรมต่อไปนี้ และสมมุติให้ทำข้อ a) ถึง j) อย่างต่อเนื่องกัน

```
int   i, j, k ;
double x, y, z ;
i = 5 ;
j = 3 ;
x = 2.5 ;
y = -3.75 ;

a) k = i * j ;              f) k += j ;
b) z = x / i ;              g) i /= k ;
c) z = y / x ;              h) k++ ;
```

```
d) i = k + 3 * j ;          i) j = y / x ;
e) k = i % j ;              j) k = i * j + x ;
```

8. ถ้า value เท่ากับ 1 แล้ว (value++ * 5) + (value++ * 3) มีค่าเท่ากับเท่าไร และหลังจากคำนวณแล้ว ค่าของ value จะเปลี่ยนไปอย่างไร
9. ถ้า value เท่ากับ 1 แล้ว (++value * 5) + (++value * 3) มีค่าเท่ากับเท่าไร และหลังจากคำนวณแล้ว ค่าของ value จะเปลี่ยนไปอย่างไร
10. สมมุติให้ค่าของตัวแปร x, y และ z เท่ากับ 5, -12 และตัวอักษร 'e' ตามลำดับ และโปรแกรมใช้รหัส ASCII ในการแสดงค่าตัวอักษร จงบอกค่าทางตรรกศาสตร์ของ Expression ต่อไปนี้

```
a) x > 5 && z < 50
b) x == 5 || y > 0
c) !( x < 5 )
d) !( z > 20 || y > 10 )
e) x > 0 && y <= 0 && z >= 20
f) z >= 'a' && z <= 'z'
g) z >= 'A' || z <= 'Z'
h) x != y && z != 100
```

11. เขียนโปรแกรมแปลงค่าเงินดอลลาร์สหรัฐอเมริกา ให้เป็นเงินบาท โดยให้สามารถกำหนดอัตราแลกเปลี่ยนได้
12. เขียนโปรแกรมอ่านค่า *x, y, z* เข้ามา แล้วคำนวณหาค่า $a = x^2 + y^2 + z^2$
13. เขียนโปรแกรมสำหรับแก้สมการ (ในกรณีที่ $A \cdot E \neq B \cdot D$)

    $Ax + By = C \quad Dx + Ey = F$

    โดยทำการรับค่า *A, B, C* แล้วรับค่า *D, E, F*
14. เขียนโปรแกรมสำหรับแก้สมการ $ax^2 + bx + c = 0$ (กรณีที่ $b^2 - 4ac \geq 0$) ทั้งนี้ ให้ใช้ฟังก์ชัน sqrt ซึ่งเป็นฟังก์ชันสำหรับหาค่ารากกำลังสอง
15. เขียนโปรแกรมสำหรับแปลงองศาเซลเซียสเป็นองศาฟาเรนไฮต์ ด้วยสมการต่อไปนี้

$$F = \frac{9}{5}C + 32$$

16. เขียนโปรแกรมสำหรับคำนวณดอกเบี้ยเงินฝาก โดยให้สามารถกำหนดอัตราดอกเบี้ยได้