Chapter 4 – Variable, Data Types, and Arithmetic Expressions

- ในบทนี้จะได้ศึกษาเกี่ยวกับตัวแปร ค่าคงที่ ชนิดของข้อมูล และการกระทำทางคณิตศาสตร์ ในภาษา C

Chapter outline

4.1 ตัวแปร

4.2 ชนิดของข้อมูล

4.2.1 integer

4.2.2 float

4.2.3 double

4.2.4 char

4.2.5 long, long long, short, unsigned

4.2.6 _Complex, _Imaginary

4.3 การกระทำทางคณิตศาสตร์

4.3.1 +, -, *, /

4.3.2 Modulus

4.3.3 การเพิ่มและลดค่าทีละหนึ่ง

4.3.4 การแปลงระหว่างเลขจำนวนเต็มและเลขทศนิยม

4.3.5 Assignment operators

4.1 ตัวแปร (variable)

- ตัวแปร ใช้ในการเก็บค่าการคำนวณและผลลัพธ์ที่ได้จากการคำนวณ
- โดยชื่อของตัวแปรสามารถเลือกให้มีความหมาย เป็นที่เข้าใจได้โดยง่าย
- ในบทที่ 3 เราได้ทดลองใช้ตัวแปรเพื่อเก็บค่าตัวเลขจำนวนเต็ม

Program 3.5 Displaying Multiple Values

```
#include <stdio.h>
#include <conio.h>
int main (void)
{
    int value1, value2, sum;
    value1 = 50;
    value2 = 25;
    sum = value1 + value2;
    printf ("The sum of %i and %i is %i\n", value1, value2,
sum);
    return 0;
}
```

- โดยตัวแปรในโปรแกรมนี้เก็บได้เฉพาะเลขจำนวนเต็ม หรือ integer เท่านั้น
- ภาษา C อนุญาตตัวแปรให้เก็บข้อมูลชนิดอื่นๆ เช่นเลขทศนิยม ตัวอักษร โดยจะต้องมีการประกาศตัวแปรให้เหมาะสม *ก่อน*ที่จะมีการใช้งานตัวแปรนั้น
- การตั้งชื่อตัวแปรมีกฎอยู่ว่า ตัวแปรจะต้องเริ่มด้วยตัวอักษร หรือ underscore _ และสามารถตามด้วยตัวอักษร underscore หรือตัวเลข 0 ถึง 9
- ตัวอย่างชื่อตัวแปรที่ถูกต้อง

  sum, pieceFlag, I, J5x7, Number_of_moves, _sysflag

- ตัวอย่างชื่อตัวแปรที่ไม่ถูกต้อง

  sum$value, piece flag, 3Spencer, me@mail, a&b, int

  เนื่องจาก _____
- ข้อสำคัญสำหรับภาษา C คือ ตัวอักษรตัวเล็กและใหญ่ แตกต่างกัน เช่น sum, Sum, SUM จะหมายถึงตัวแปรคนละตัวกัน
- ชื่อตัวแปรจะยาวเท่าใดก็ได้ แต่ปกติแล้วก็จะตั้งให้สั้นเพื่อความสะดวก
- การตั้งชื่อตัวแปรควรจะเลือกให้มีความหมาย เพื่อง่ายต่อความเข้าใจและการแก้ไขโปรแกรมในภายหลัง

4.2 ชนิดข้อมูล (data types)

- ในบทที่ 3 เราได้รู้จักชนิดข้อมูล int แล้ว ซึ่งใช้สำหรับประกาศตัวแปรเพื่อเก็บข้อมูลประเภท _____
- นอกจากชนิดข้อมูลแบบ int แล้ว ภาษา C ยังได้กำหนดชนิดข้อมูลอื่น ๆ อีก ได้แก่ float, double, char, และ _Bool
- โดยตัวแปรที่ถูกประกาศเป็นชนิด float (ย่อมาจาก floating-point number) จะใช้เก็บตัวเลขทศนิยม
- ตัวแปรชนิด double (มาจาก double precision) ก็ใช้เก็บตัวเลขทศนิยมเช่นเดียวกัน แต่จะมีความแม่นยำ หรือจำนวนตำแหน่งทศนิยมมากกว่าประมาณสองเท่าของตัวแปรชนิด float

- ซึ่งความแม่นยำนี้แลกมาด้วยเนื้อที่ในหน่วยความจำที่จะต้องใช้เพิ่มขึ้น สำหรับตัวแปรชนิด double
- ตัวแปรชนิด char (ย่อมาจาก character) ใช้เก็บ_____ได้หนึ่งตัว
- ตัวแปรชนิด _Bool (มาจาก boolean) ใช้เก็บได้เพียงค่า 0 หรือ 1 ซึ่งมีประโยชน์สำหรับใช้แทนสถานะถูกหรือผิด (true/false)
- ชนิดข้อมูลแต่ละชนิด กินเนื้อที่ในหน่วยความจำแตกต่างกัน
- ข้อมูลชนิดเดียวกันก็อาจใช้เนื้อที่หน่วยความจำแตกต่างกันขึ้นอยู่กับระบบคอมพิวเตอร์ที่ใช้ เช่น int อาจจะใช้เนื้อที่ 32 บิท หรือ 64 บิทเพื่อเก็บก็ได้ และก็จะมีขอบเขตการเก็บตัวเลขได้ต่างกัน เช่น int อาจจะเก็บค่าใดๆในช่วง -32768 ถึง 32767 หรือ -65536 ถึง 65535
- นอกจากชนิดของข้อมูลแบบทั่วไปแล้ว ยังมีตัวกำหนดชนิดข้อมูลพิเศษ ได้แก่ long, long long, short, unsigned และ signed
- คำสั่งที่ใช้ประกาศตัวแปร จะอยู่ในรูป

  type name = initial_value;

  เช่น

  int x2 = 10; float y3;

  double a1 = 10.45, a2 = 5.43, a3 = 8.56;

4.2.1 ชนิดข้อมูล int

- ตัวแปรที่ถูกประกาศเป็นชนิด int สามารถเก็บเลขจำนวนเต็มบวกหรือลบได้
- ไม่มีการใช้เครื่องหมายลูกน้ำ สำหรับเลขจำนวนเต็มที่มากกว่า 999 เช่น 12000 ไม่ใช้ 12,000
- นอกจากนี้ชนิดของข้อมูลแบบ int ยังสามารถจัดการกับเลขจำนวนเต็มฐานแปดและฐานสิบหกได้ด้วย

ฐานสิบ	ฐานแปด	ฐานสิบหก	ฐานสิบ	ฐานแปด	ฐานสิบหก
0	0	0	11	13	B
1	1	1	12	14	C
2	2	2	13	15	D
3	3	3	14	16	E
4	4	4	15	17	F
5	5	5	16	21	10
6	6	6	17	22	11
7	7	7	18	23	12
8	10	8	19	24	13
9	11	9	20	25	14
10	12	A	21	26	15

- โดยที่การเขียนเลขฐานแปดในภาษา C ทำได้โดยใส่เลข 0 นำหน้าตัวเลขใดๆ เช่น 0177 จะหมายถึงเลข 177 ในฐานแปด หรือ 177_8 ซึ่งมีค่าเท่ากับ 127 ในฐานสิบ (1 x 64 + 7 x 8 + 7 = 127)
- ดังนั้นเมื่อเขียน 0 นำตัวเลขใดแล้ว ตัวเลขนั้นจะประกอบไปด้วยเลข 0-7 เท่านั้น

- ส่วนการแสดงผลเลขฐานแปดในภาษา C สามารถใช้ %o หรือ %#o ตัวอย่างเช่น

  printf(“%i is equal to %o or %#o in octal notation”, 127, 127, 127);

  จะถูกแสดงเป็น

  127 is equal to 177 or 0177 in octal notation

  หรือ

  printf(“%i is equal to %o or %#o in octal notation”, 0177, 127 0177);

  จะถูกแสดงเป็น

  _____

- สำหรับกรณีของเลขฐานสิบหก จะใช้ตัวอักษร 0x นำหน้าตัวเลขฐานสิบหก เพื่อบ่งบอก โดยตัวเลขที่ตามหลังสามารถประกอบไปด้วย 0-9 และ A-F หรือ a-f ซึ่งหมายถึง 10-15
- ตัวอย่างการใช้งานในภาษา C

  int rgbColor;
  rgbColor = 0xFFEF0D;
- ส่วนการแสดงผล จะใช้ตัว %x หรือ %#x ในคำสั่ง printf เพื่อทำให้แสดงผลเป็นเลขฐานแปด

  เช่น

  printf(“Color is %x or %#x”, rgbColor, rgbColor);

  จะถูกแสดงเป็น

  Color is ffefod or 0xffefod

  หรือ

printf(“Color is %X or %#X”, rgbColor, rgbColor);

จะถูกแสดงเป็น

Color is FFEF0D or 0XFFEF0D

4.2.2 ชนิดข้อมูล float

- ตัวแปรที่ถูกประกาศเป็นชนิด float ใช้เก็บจำนวนที่มีจุดทศนิยมได้
- ข้อมูลที่ถือเป็น float เช่น 3., 125.8, -.0001, 4.0
- นอกจากนี้ ภาษา C ยังอนุญาตการเขียนทศนิยมในรูปแบบทางวิทยาศาสตร์ เช่น 1.7e-4 ซึ่งหมายถึง 1.7×10^{-4} หรือ 0.00017 โดยที่ค่านำหน้าตัวอักษร e เรียกว่า mantissa ซึ่งกำหนดให้มีค่าอยู่ระหว่าง 0-9 ส่วนค่าตามหลัง e เรียก exponent และ ตัวอักษร e สามารถใช้ได้ทั้งตัวใหญ่และตัวเล็ก
    - 0.034 เขียนเป็นรูปแบบทางวิทยาศาสตร์ได้ว่า _____
    - 156.562 เขียนเป็นรูปแบบทางวิทยาศาสตร์ได้ว่า _____
- ในคำสั่ง printf จะใช้ %f หรือ %e หรือ %g เพื่อแสดงผลข้อมูลชนิด float
- ในภาคปฏิบัติ ให้ทดลองเขียนโปรแกรมเพื่อหาความแตกต่างของการใช้ %f, %e, และ %g สำหรับการแสดงผลตัวเลขทศนิยม

4.2.3 ชนิดข้อมูล double

- ตัวแปรที่ถูกประกาศเป็นชนิด double จะเก็บข้อมูลทศนิยมเหมือนชนิด float แต่เก็บได้ละเอียดกว่าประมาณสองเท่า และโดยปกติจะใช้เนื้อที่ในหน่วยความจำประมาณ 64 บิท
- ใช้ %f, %e, หรือ %g ใน printf ได้เช่นเดียวกับตัวแปรชนิด float

4.2.4 ชนิดข้อมูล char

- ตัวแปรที่ถูกประกาศเป็นชนิด char สามารถเก็บตัวอักขระได้หนึ่งตัว โดยตัวอักขระที่จะจัดเก็บจะต้องอยู่ในเครื่องหมาย ‘’ เช่น ‘a’, ‘;’, ‘0’ สังเกตว่า ‘0’ ในที่นี้โปรแกรมจะถือเป็นตัวอักขระ และนำไปคำนวณทางคณิตศาสตร์ไม่ได้
- ‘\n’ ถือเป็นตัวอักขระพิเศษสำหรับขึ้นบรรทัดใหม่ สามารถจัดเก็บในตัวแปรชนิด char ได้เช่นกัน
- ในคำสั่ง printf สามารถใช้ %c สำหรับแสดงค่าที่เก็บในตัวแปรชนิด char ได้

4.2.5 ชนิดข้อมูล _Bool

- ตัวแปรที่ประกาศเป็นชนิด _Bool เก็บได้เพียงค่า 0 และ 1 หรือ false และ true
- ตัวอย่างการใช้งาน ใช้ระบุสถานะการอ่านไฟล์ว่าอ่านสำเร็จหรือไม่ หากสำเร็จ ค่าจะเป็น 1 และถ้าไม่สำเร็จค่าเป็น 0 เป็นต้น

- ชนิดข้อมูล _Bool จะใช้มากในเรื่องของการตัดสินใจ (making decisions) ซึ่งจะกล่าวต่อไปในบทที่ 6

Program 4.1 Using the basic data types

```
#include <stdio.h>
#include <conio.h>
int main(void)
{
    int integerVar = 100;
    float floatingVar = 331.79;
    double doubleVar = 8.44e+11;
    char charVar = ‘W’;
    _Bool boolVar = 0;

    printf(“integerVar = %i\n”, integerVar);
    printf(“floatingVar = %f\n”, floatingVar);
    printf(“doubleVar = %e\n”, doubleVar);
    printf(“doubleVar = %g\n”, doubleVar);
    printf(“charVar = %c\n”, charVar);
    printf(“boolVar = %i\n”, boolVar);
    getch();
    return 0;
}
```

```
integerVar = 100
floatingVar = 331.790009
doubleVar = 8.440000e+11
doubleVar = 8.44e+11
charVar = W
boolVar = 0
```

- สังเกต

  ```
  int integerVar = 100;
  ```

  มีผลเช่นเดียวกับ

  ```
  int integerVar;
  integerVar = 100;
  ```

- สังเกตการแสดงค่าของ floatingVar ที่ผิดไปจากค่าที่ถูกใส่ให้
- สังเกต boolVar ที่ใช้ %i แสดงผล

4.2.6 ตัวกำหนดพิเศษ long, long long, short, และ unsigned

- long ใช้ใส่หน้าชนิดข้อมูลเพื่อขยายขอบเขตการเก็บข้อมูล เช่น

  long int factorial;

  ซึ่งขอบเขตการเก็บข้อมูลที่ขยายได้ ขึ้นอยู่กับระบบคอมพิวเตอร์ที่ใช้งาน
- การแสดงข้อมูลประเภทนี้ จะใช้อักขระ %li หรือ %lo หรือ %lx กับคำสั่ง printf
- long long ใช้เช่นเดียวกับ long เช่น

  long long int maxAllowedStorage;

  เพื่อขยายขอบเขตการเก็บข้อมูลให้มากขึ้นกว่า long int
- การแสดงข้อมูลประเภทนี้ จะใช้อักขระ %lli หรือ %llo หรือ %llx
- long ใช้กับชนิดข้อมูล double ได้เช่นเดียวกัน

  long double US_deficit_2004;
- การแสดงข้อมูลประเภทนี้ จะใช้อักขระ %Lf หรือ %Le หรือ %Lg
- ส่วน short นั้น ใช้กับข้อมูล int เพื่อลดขอบเขตการเก็บข้อมูลลง มีประโยชน์เมื่อต้องการประหยัดการใช้งานหน่วยความจำของเครื่องคอมพิวเตอร์

  เช่น short int small_number;
- การแสดงข้อมูลประเภทนี้ จะใช้อักขระ %hi หรือ %ho หรือ %hx

- unsigned ใช้ร่วมกับข้อมูลแบบ int โดยสามารถใช้ร่วมกับ short, long, long long ได้ด้วย โดยใช้เมื่อต้องการเก็บค่าจำนวนเต็มบวกเท่านั้น ดังนั้นขอบเขตของค่าที่เก็บได้ ทางฝั่งบวกจะเพิ่มขึ้น เช่น

  unsigned short int x1; unsigned int x2;
  unsigned long int x3; unsigned long long int x4;

Table 4.1 **Basic Data Types**

Type	Constant Examples	printf chars
char	'a', '\n'	%c
_Bool	0, 1	%i, %u
short int	—	%hi, %hx, %ho
unsigned short int	—	%hu, %hx, %ho
int	12, -97, 0xFFE0, 0177	%i, %x, %o
unsigned int	12u, 100U, 0XFFu	%u, %x, %o
long int	12L, -2001, 0xffffL	%li, %lx, %lo
unsigned long int	12UL, 100ul, 0xffeeUL	%lu, %lx, %lo
long long int	0xe5e5e5e5LL, 5001l	%lli, %llx, &llo
unsigned long long int	12ull, 0xffeeULL	%llu, %llx, %llo
float	12.34f, 3.1e-5f, 0x1.5p10, 0x1P-1	%f, %e, %g, %a
double	12.34, 3.1e-5, 0x.1p3	%f, %e, %g, %a
long double	12.341, 3.1e-5l	%Lf, $Le, %Lg

4.2.5 ชนิดข้อมูล _Complex และ _Imaginary

- ใช้ในการคำนวณและจัดการจำนวนจินตภาพ
- ต้องประกาศ #include <complex.h> ไว้ที่ header ของโปรแกรมก่อนใช้งาน ซึ่งไลบรารีนี้จะรวมฟังก์ชันที่ใช้ร่วมกับจำนวนจินตภาพไว้
- ตัวอย่างการประกาศตัวแปรเป็นชนิด complex และกำหนดข้อมูลให้ตัวแปร

```
double _Complex c1 = 5 + 10.5*I;
```

- จะต้องใช้ฟังก์ชันพิเศษในการคำนวณทางคณิตศาสตร์สำหรับจำนวนจินตภาพ

4.3 การกระทำทางคณิตศาสตร์ (Arithmetic operation)

4.3.1 +, -, *, /

- Binary arithmetic operators: add (+), subtract (-), multiply (*), divide (/)

Program 4.2 Using the Arithmetic Operators

```
#include <stdio.h>
#include <conio.h>
int main(void)
{
    int a = 100;
    int b = 2;
    int c = 25;
    int d = 4;
    int result;

    result = a - b; // subtraction
    printf("a - b = %i\n", result);

    result = b * c; // multiplication
    printf("b * c = %i\n", result);

    result = a / c; // division
    printf("a / c = %i\n", result);

    result = a + b * c; // precedence
    printf("a + b * c = %i\n", result);
    printf("a * b + c * d = %i\n", a * b + c * d);

    getch();
    return 0;
}
```

- สังเกต ... ซึ่งมี operators มากกว่าหนึ่งตัว ในที่นี้ภาษา C จะมีกฎตายตัวสำหรับลำดับความสำคัญของ operator โดยที่ operator ที่มีลำดับความสำคัญมากกว่าจะถูกกระทำก่อน
- ในที่นี้ a + b * c = a + (b * c) และ a * b + c * d = (a * b) + (c * d)

Program 4.3 More Examples with Arithmetic Operators

```
#include <stdio.h>
#include <conio.h>
int main (void)
{
   int a = 25;
   int b = 2;
   float c = 25.0;
   float d = 2.0;

   printf("6 + a / 5 * b = %i\n", 6 + a / 5 * b);
   printf("a / b * b = %i\n", a / b * b);

   printf("c / d * d = %f\n", c / d * d);
   printf("-a = %i\n", -a);

   return 0;
}
```

```
6 + a / 5 * b = 16
a / b * b = 24
c / d * d = 25.000000
-a = -25
```

- จะต้องเลือกใช้ชนิดของตัวแปรให้เหมาะสมกับการคำนวณ เพื่อความถูกต้องและความแม่นยำขอผลลัพธ์
- โดยในงานปกติ float ก็แม่นยำเพียงพอสำหรับการคำนวณเลขทศนิยมแล้ว

4.3.2 Modulus

Program 4.4 Illustrating the Modulus Operator

```
#include <stdio.h>
#include <conio.h>
int main (void)
{
    int a = 25, b = 5, c = 10, d = 7;
    printf ("a %% b = %i\n", a % b);
    printf ("a %% c = %i\n", a % c);
    printf ("a %% d = %i\n", a % d);
    printf ("a / d * d + a %% d = %i\n", a / d * d + a % d);

    getch();
    return 0;
}
```

```
a % b = 0
a % c = 5
a % d = 4
a / d * d + a % d = 25
```

- Modulus ใช้กับข้อมูลชนิดเลขจำนวนเต็มหรือ integer เท่านั้น
- Modulus ถูกจัดให้มีลำดับความสำคัญเท่ากับคูณหรือหาร ดังนั้น

  table + value % TABLE_SIZE

  มีความหมายเดียวกันกับ

  table + (value % TABLE_SIZE)

4.3.3 การเพิ่มและลดค่าทีละหนึ่ง

- ใช้ได้กับตัวแปรชนิด int (รวมถึง unsigned int, short int, long int, long long int) เท่านั้น

เครื่องหมาย	การดำเนินการ	ตัวอย่าง	ความหมาย
++	Increment	y = ++x;	x = x + 1; y = x;
		y = x++;	y = x; x = x + 1;
--	decrement	y = --x;	x = x – 1; y = x;
		y = x--;	y = x; x = x – 1;

4.3.4 การแปลงระหว่างเลขจำนวนเต็มและเลขทศนิยม

- จะมีการแปลงค่าเกิดขึ้นในการคำนวณที่มีทั้งเลขจำนวนเต็มและทศนิยมเข้ามาเกี่ยวข้อง

Program 4.5 Converting Between Integers and Floats

```
#include <stdio.h>
#include <conio.h>
int main (void)
{
    float f1 = 123.125, f2;
    int i1, i2 = -150;
    char c = 'a';

    i1 = f1; // floating to integer conversion
    printf ("%f assigned to an int produces %i\n", f1, i1);

    f1 = i2; // integer to floating conversion
    printf ("%i assigned to a float produces %f\n", i2, f1);

    f1 = i2 / 100; // integer divided by integer
    printf ("%i divided by 100 produces %f\n", i2, f1);

    f2 = i2 / 100.0; // integer divided by a float
    printf ("%i divided by 100.0 produces %f\n", i2, f2);

    f2 = (float) i2 / 100; // type cast operator
    printf ("(float) %i divided by 100 produces %f\n", i2, f2);

    return 0;
}
```

```
123.125000 assigned to an int produces 123
-150 assigned to a float produces -150.000000
-150 divided by 100 produces -1.000000
-150 divided by 100.0 produces -1.500000
(float) -150 divided by 100 produces -1.500000
```

- เมื่อเลขทศนิยมถูกกำหนดให้กับตัวแปรชนิด integer ค่าหลังจุดทศนิยมจะถูกปัดลงทั้งหมด
- ในทางกลับกัน เมื่อเลขจำนวนเต็มถูกกำหนดให้กับตัวแปรชนิด float จะไม่เกิดการเปลี่ยนแปลงค่า
- การคำนวณแบบ integer เกิดขึ้นเมื่อตัวแปรหรือค่าคงที่ทั้งสองตัวเป็นชนิด integer โดยทศนิยมที่เกิดขึ้นจะถูกตัดทิ้ง ถึงแม้ผลลัพธ์ที่ได้จะถูกกำหนดให้กับตัวแปรชนิด float ก็ตาม
- การคำนวณแบบ floating point เกิดขึ้นเมื่อมีตัวแปรหรือค่าคงที่ตัวใดตัวหนึ่งหรือทั้งสองตัวเป็นชนิด float
- เราสามารถใช้ type cast operator เพื่อเปลี่ยนให้ค่าที่ตามหลังเป็นชนิดที่เราต้องการได้ก่อนการคำนวณ ในตัวอย่าง

  f2 = (float) i2 / 100;

  จะบังคับให้แปลงค่าที่เก็บใน i2 เป็นชนิด float ก่อนการคำนวณ แต่ไม่ทำให้ค่าใน i2 เปลี่ยนเป็น float อย่างถาวร
- ตัวอย่าง (int) 29.55 + (int) 21.99; จะถูกคำนวณเป็น 29 + 21

  หรือ

  (float) 6 / (float) 4 จะถูกคำนวณเป็น 6.0/4.0 = 1.5 และมีความหมายเช่นเดียวกับ (float) 6/4

4.3.5 Assignment operators

- ในภาษา C มี operator พิเศษที่รวมเอาการคำนวณทางคณิตศาสตร์ไว้กับการกำหนดค่าให้ตัวแปร คือ *op*= เมื่อ *op* เป็น +, -, *, /, หรือ % โดยเรียกว่า assignment operators
- พิจารณาคำสั่งนี้

  count += 10;

  การทำงานของคำสั่งคือจะบวกค่าทางด้านขวาของ operator ในที่นี้คือ 10 เข้ากับตัวแปรทางด้านซ้าย หรือ count แล้วเก็บค่าที่ได้กลับเข้าไปในตัวแปรทางด้านซ้าย ดังนั้นคำสั่งด้านบนจะทำงานเหมือนกับคำสั่ง

  count = count + 10;
- เช่นเดียวกัน counter -= 5; มีความหมายเท่ากับ _____
- อีกตัวอย่างหนึ่ง

  a /= b + c;

  ในคำสั่งนี้ การบวกซึ่งมีลำดับความสำคัญมากกว่า /= จะถูกกระทำก่อน นั่นคือ

  a = a / (b + c);
- ประโยชน์ของการใช้งาน operator เหล่านี้คือ คำสั่งจะเขียนได้กระชับขึ้นและอ่านเข้าใจได้ง่ายขึ้น ในบางกรณีจะทำให้โปรแกรมทำงานได้ไวขึ้นเนื่องจากใช้คำสั่งในการคำนวณน้อยลง