1

259201
Computer Programming for Engineers

Variables

Week 2
Introduction to C Programming

Outline

2.1 ตัวอย่างโปรแกรมภาษาซี : การแสดงข้อความ 1 บรรทัด

```
1   /* Fig. 2.1: fig02_01.c
2      A first program in C */
3   #include <stdio.h>
4
5   /* function main begins program execution */
6   int main()
7   {
8      printf( "Welcome to C!\n" );
9
10     return 0; /* indicate that program ended successfully */
11
12  } /* end function main */
Welcome to C!
```

- Comments
  - ข้อความที่อยู่ระหว่างเครื่องหมาย `/*` และ `*/` คือ Comment เพื่อช่วยอธิบายการทำงานของโปรแกรม แต่ไม่มีผลกับการทำงาน
- `#include <stdio.h>`
  - เป็นการระบุว่าโปรแกรมนี้ต้องมีการใช้ Header file ชื่อ `stdio.h` เพื่อให้โปรแกรมสามารถทำงานที่เกี่ยวกับ input/output มาตรฐานได้

2.1 ตัวอย่างโปรแกรมภาษาซี : การแสดงข้อความ 1 บรรทัด

- `int main()`
  - โปรแกรมภาษาซีจะต้องประกอบไปด้วย 1 ฟังก์ชันหรือมากกว่า และฟังก์ชันที่จะต้องให้มีทุกโปรแกรมคือ ฟังก์ชันชื่อว่า `main`
  - เครื่องหมายวงเล็บจะตามหลังชื่อฟังก์ชัน
  - `int` หมายถึงฟังก์ชัน `main` ทำการคืนค่า หรือ Return ค่าที่เป็นชนิดจำนวนเต็ม (integer)
  - เครื่องหมาย Braces ({ และ }) เป็นการระบุขอบเขตของการทำงาน
    - ในส่วนชุดคำสั่งของฟังก์ชันทุกฟังก์ชันจะต้องอยู่ในเครื่องหมาย { }

2.1 ตัวอย่างโปรแกรมภาษาซี : การแสดงข้อความ 1 บรรทัด

- `printf( "Welcome to C!\n" );`
  - สั่งคอมพิวเตอร์ให้ทำงาน ในกรณีนี้จะเป็นการแสดงผลข้อความที่อยู่ในเครื่องหมายคำพูด (" ")
  - บรรทัดหนึ่งเรียกว่า Statement
    - ทุก Statement จะต้องปิดท้ายด้วยเครื่องหมาย Semicolon (;)
  - Escape character (\)
    - เป็นการระบุให้ printf ทำบางสิ่งบางอย่างที่ไม่ปกติ เช่น \n คือการให้ขึ้นบรรทัดใหม่ ไม่ใช่ให้แสดงผล \n ออกไปบนหน้าจอ

2.1 ตัวอย่างโปรแกรมภาษาซี : การแสดงข้อความ 1 บรรทัด

Escape Sequence	Description
\n	Newline. Position the cursor at the beginning of the next line.
\t	Horizontal tab. Move the cursor to the next tab stop.
\a	Alert. Sound the system bell.
\\	Backslash. Insert a backslash character in a string.
\"	Double quote. Insert a double quote character in a string.

Table 2.1 Some common escape sequences.

2.1 ตัวอย่างโปรแกรมภาษาซี : การแสดงข้อความ 1 บรรทัด

- `return 0;`
  - การสิ้นสุดการทำงาน และคืนค่า 0 โดยทั่วไปหมายถึงการทำงานเป็นปกติ
- Right brace }
  - ระบุจุดสิ้นสุดของฟังก์ชัน `main`

```
1  /* Fig. 2.2: fig02_02.c
2     Printing on one line with two printf statements */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     printf( "Welcome " );
9     printf( "to C!\n" );
10
11    return 0; /* indicate that program ended successfully */
12
13 } /* end function main */
```

fig02_02.c

```
Welcome to C!
```

Program Output

```
1  /* Fig. 2.3: fig02_03.c
2     Printing multiple lines with a single printf */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     printf( "Welcome\nto\nC!\n" );
9
10    return 0; /* indicate that program ended successfully */
11
12 } /* end function main */
```

fig02_03.c

```
Welcome
to
C!
```

Program Output

ชนิดของข้อมูลในภาษาซี (Data Type)

ข้อมูลที่มีการใช้งานในภาษาซี แบ่งออกเป็น 6 ชนิด คือ

1) ข้อมูลชนิดตัวเลขจำนวนเต็ม (Integer)
2) ข้อมูลชนิดตัวเลขทศนิยม (Float)
3) ข้อมูลชนิดเลขฐานแปด (Octal)
4) ข้อมูลชนิดเลขฐานสิบหก (Hexadecimal)
5) ข้อมูลชนิดตัวอักขระ (Character)
6) ข้อมูลชนิดข้อความ (String)

การประกาศตัวแปร (Variable Declaration)

ตัวแปรในภาษาซีแบ่งได้ 2 ประเภทใหญ่ๆ คือ ตัวแปรพื้นฐาน ที่หมายถึงตัวแปรที่เก็บข้อมูลได้เพียงค่าเดียว และตัวแปรชุด คือ ตัวแปรที่สามารถเก็บข้อมูลไว้ได้หลายค่าภายในตัวแปรตัวเดียว ซึ่งในหัวข้อนี้เราจะกล่าวถึงแค่ตัวแปรพื้นฐาน ส่วนตัวแปรชุดจะเรียนในหัวข้อถัดไป

ตัวแปรพื้นฐานในภาษาซีที่กำหนดตามมาตรฐาน ANSI C

ชนิดของตัวแปร	ความหมาย
char	เก็บข้อมูลชนิดอักขระ โดยใช้พื้นที่ในการจัดเก็บ 8 บิต
unsigned char	เก็บข้อมูลชนิดอักขระแบบไม่คิดเครื่องหมาย
int	เก็บข้อมูลชนิดตัวเลขจำนวนเต็ม โดยใช้พื้นที่ในการจัดเก็บ 16 บิต
unsigned int	เก็บข้อมูลชนิดตัวเลขจำนวนเต็มแบบไม่คิดเครื่องหมาย
short	เก็บข้อมูลชนิดตัวเลขจำนวนเต็มแบบสั้น โดยใช้พื้นที่ในการจัดเก็บ 8 บิต
unsigned short	เก็บข้อมูลชนิดตัวเลขจำนวนเต็มแบบสั้นโดยไม่คิดเครื่องหมาย
long	เก็บข้อมูลชนิดตัวเลขจำนวนเต็มแบบยาว โดยใช้พื้นที่ในการจัดเก็บ 32 บิต
unsigned long	เก็บข้อมูลชนิดตัวเลขจำนวนเต็มแบบยาวโดยไม่คิดเครื่องหมาย
float	เก็บข้อมูลชนิดตัวเลขทศนิยม โดยใช้พื้นที่ในการจัดเก็บ 32 บิต
double	เก็บข้อมูลชนิดตัวเลขทศนิยม โดยใช้พื้นที่ในการจัดเก็บ 64 บิต
long double	เก็บข้อมูลชนิดตัวเลขทศนิยม โดยใช้พื้นที่ในการจัดเก็บ 128 บิต

13

รูปแบบการประกาศตัวแปร

- ในภาษาซีมีรูปแบบการประกาศตัวแปร ดังนี้

type variable;

type คือ ชนิดของตัวแปรที่จะสร้าง
variable คือชื่อของตัวแปรที่ต้องการใช้

- ตัวอย่างการประกาศตัวแปร เช่น `int num;`
  `float grade;`
- หากต้องการประกาศตัวแปรหลายตัวชนิดเดียวกันก็สามารถทำได้ โดยการใช้เครื่องหมาย , คั่น เช่น `int a,b,c;`

14

- นอกจากนี้ยังสามารถกำหนดค่าเริ่มต้นให้กับตัวแปรไปพร้อมกับการประกาศตัวแปร เช่น

```
int num = 1;
char ch = '#', d = 'D';
```

- โดยหลักการตั้งชื่อตัวแปรมาใช้งานนั้น ควรคำนึงถึงว่าจะต้องตั้งให้ถูกต้องตามข้อกำหนดของภาษาซี และ ควรจะตั้งชื่อตัวแปรให้สอดคล้องกับการทำงานหรือหน้าที่ของตัวแปรนั้นๆ เพราะเมื่อถึงเวลาต้องมาทำการปรับปรุงแก้ไขโปรแกรม จะสามารถทำได้โดยไม่ยากนัก

15

Format ที่ใช้บ่งบอกชนิดของตำแหน่ง (Place Holders) ที่จะมีการส่งข้อมูล

- ในการเขียนโปรแกรมนั้น เราจะต้องมีที่ใช่ Place Holders ในการรับส่งข้อมูลเพื่อการแสดงผล โดยรูปแบบการใช้ Place Holders นั้น จะขึ้นอยู่กับชนิดของตัวแปรที่ต้องการส่งข้อมูล ดังเช่น

```
#include <stdio.h>

void main()
{
   int product_price;
   printf("How much is that ? \n");
   scanf("%d", &product_price);
   printf("oh! %d ?, it's too cheap\n\n", product_price);
}
```

16

Control Character	Explanation
%c	a single character
%d	a decimal integer
%i	an integer
%e	scientific notation, with a lowercase "e"
%E	scientific notation, with a uppercase "E"
%f	a floating-point number
%g	use %e or %f, whichever is shorter
%G	use %E or %f, whichever is shorter
%o	an octal number
%x	unsigned hexadecimal, with lowercase letters
%X	unsigned hexadecimal, with uppercase letters
%u	an unsigned integer
%s	a string
%x	a hexadecimal number
%p	a pointer
%n	the argument shall be a pointer to an integer into which is placed the number of characters written so far
%%	a percent sign

17

การคำนวณทางคณิตศาสตร์

สัญลักษณ์ที่ใช้ในการคำนวณ	ความหมาย	ตัวอย่าง
+	การบวก	5.0 + 2.0 = 7.0
-	การลบ	5.0 -2.0 = 3.0
*	การคูณ	5.0 * 2.0= 10.0
/	การหาร	5.0/2.0 = 2.5
%	การหาค่าเศษ	5 % 2 = 1

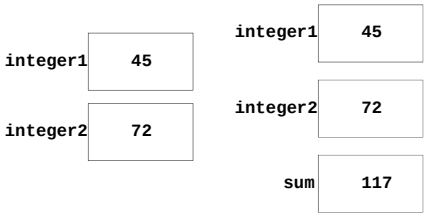
```
1  /* Fig. 2.4: fig02_04.c
2     Addition program */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     int integer1; /* first number to be input by user  */
9     int integer2; /* second number to be input by user */
10    int sum;      /* variable in which sum will be stored */
11
12    printf( "Enter first integer\n" );  /* prompt */
13    scanf( "%d", &integer1 );           /* read an integer */
14
15    printf( "Enter second integer\n" ); /* prompt */
16    scanf( "%d", &integer2 );           /* read an integer */
17
18    sum = integer1 + integer2;          /* assign total to sum */
19
20    printf( "Sum is %d\n", sum );       /* print sum */
21
22    return 0;  /* indicate that program ended successfully */
23
24 } /* end function main */
```

fig02_04.c

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

Program Output

•A visual representation



21

ลำดับการทำงาน

ลำดับความสำคัญ	เครื่องหมาย	การดำเนินการ
1	( )	Left to Right
2	!, ++, --, (typecast)	*Right to Left*
3	*, /, %	Left to Right
4	+, -	Left to Right
5	<, <=, >, >=	Left to Right
6	= =, !=	Left to Right
7	&&	Left to Right
8	||	Left to Right
9	*=, /=, %=, +=, -=	Left to Right

ตัวอย่างลำดับการทำงาน

Step 1. `y = 2 * 5 * 5 + 3 * 5 + 7;` (Leftmost multiplication)
`2 * 5 is 10`

Step 2. `y = 10 * 5 + 3 * 5 + 7;` (Leftmost multiplication)
`10 * 5 is 50`

Step 3. `y = 50 + 3 * 5 + 7;` (Multiplication before addition)
`3 * 5 is 15`

Step 4. `y = 50 + 15 + 7;` (Leftmost addition)
`50 + 15 is 65`

Step 5. `y = 65 + 7;` (Last addition)
`65 + 7 is 72`

Step 6. `y = 72;` (Last operation—place 72 in y)

23

เครื่องหมาย	ตัวอย่างการใช้งาน	ความหมาย
++	y = ++x y = x++	บวกค่าในตัวแปร x เพิ่มขึ้น 1 ก่อนที่จะกำหนดค่า x ให้กับตัวแปร y กำหนดค่า x ให้กับตัวแปร y ก่อนที่จะบวกค่า x เพิ่มขึ้น 1
--	y = --x y = x--	ลบค่าในตัวแปร x ลง 1 ก่อนที่จะกำหนดค่า x ให้กับตัวแปร y กำหนดค่า x ให้กับตัวแปร y ก่อนที่จะลดค่า x ลง 1
+=	y += x	บวกค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y
-=	y -= x	ลบค่าในตัวแปร y ด้วยค่าในตัวแปร xผลลัพธ์ที่ได้กำหนดกลับไปให้ y
*=	y *= x	คูณค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y
/=	y /= x	หารค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y
%=	y %= x	หารค่าในตัวแปร y ด้วยค่าในตัวแปร x เศษจากการหารเป็นผลลัพธ์ที่ได้กำหนดกลับไปให้ y

24

การเปรียบเทียบ

เครื่องหมาย	การเปรียบเทียบ	ตัวอย่างการใช้งาน	ความหมาย
= =	เท่ากับ	x = = y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x เท่ากับค่าในตัวแปร y
!=	ไม่เท่ากับ	x!=y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x ไม่เท่ากับค่าในตัวแปร y
<	น้อยกว่า	x<y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x น้อยกว่าค่าในตัวแปร y
<=	น้อยกว่าหรือเท่ากับ	x<=y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x น้อยกว่า หรือ เท่ากับค่าในตัวแปร y
>	มากกว่า	x>y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x มากกว่าค่าในตัวแปร y
>=	มากกว่าหรือเท่ากับ	x>=y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x มากกว่า หรือเท่ากับค่าในตัวแปร y

```c
/* Fig. 2.9: fig02_09.c
   Using if statements, relational
   operators, and equality operators */
#include <stdio.h>

/* function main begins program execution */
int main()
{
   int num1; /* first number to be read from user  */
   int num2; /* second number to be read from user */

   printf( "Enter two integers, and I will tell you\n" );
   printf( "the relationships they satisfy: " );

   scanf( "%d%d", &num1, &num2 ); /* read two integers */

   if ( num1 == num2 ) {
      printf( "%d is equal to %d\n", num1, num2 );
   } /* end if */

   if ( num1 != num2 ) {
      printf( "%d is not equal to %d\n", num1, num2 );
   } /* end if */

```

fig02_09.c (Part 1 of 2)

```c
   if ( num1 < num2 ) {
      printf( "%d is less than %d\n", num1, num2 );
   } /* end if */

   if ( num1 > num2 ) {
      printf( "%d is greater than %d\n", num1, num2 );
   } /* end if */

   if ( num1 <= num2 ) {
      printf( "%d is less than or equal to %d\n", num1, num2 );
   } /* end if */

   if ( num1 >= num2 ) {
      printf( "%d is greater than or equal to %d\n", num1, num2 );
   } /* end if */

   return 0;   /* indicate that program ended successfully */

} /* end function main */
```

fig02_09.c (Part 2 of 2)

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
```

Program Output

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
```

Program Output (continued)