# Fundamentals of c programming

### What is C Programming (ภาษา C คืออะไร)

```c
#include <stdio.h>

int main () {
   printf ("Hello KMUTT");
   return 0;
}
```

ภาษาซี (C Programming Language) คือ ภาษาคอมพิวเตอร์ที่ใช้สำหรับพัฒนาโปรแกรมทั่วไป ถูกพัฒนาครั้งแรกเพื่อใช้เป็นภาษาสำหรับพัฒนาระบบปฏิบัติการยูนิกซ์ (Unix Operating System) แทนภาษา Assembly ซึ่งเป็นภาษาระดับต่ำที่สามารถกระทำในระบบฮาร์ดแวร์ได้ด้วยความรวดเร็ว แต่จุดอ่อนของภาษา Assembly ก็คือ ความยุ่งยากในการเขียนโปรแกรมความเป็นเฉพาะตัว และความแตกต่างกันไปในแต่ละเครื่อง เดนนิส ริตซี (Dennis Ritchie) จึงได้คิดค้นพัฒนาภาษาใหม่นี้ขึ้นมาเมื่อประมาณต้นปี ค.ศ. 1970 โดยการรวบรวมเอาจุดเด่นของแต่ละภาษาระดับสูงผนวกเข้ากับภาษาระดับต่ำ เรียกชื่อว่า ภาษาซี

เมื่อภาษาซีได้รับความนิยมมากขึ้น จึงมีผู้ผลิตคอมไพเลอร์ (Compiler) ภาษาซีออกมาแข่งขันกันมากมาย ทำให้เริ่มมีการใส่ลูกเล่นต่าง ๆ เพื่อดึงดูดใจผู้ซื้อ ทาง American National Standard Institute (ANSI) จึงตั้งข้อกำหนดมาตรฐานของภาษาซีขึ้น เรียกว่า ANSI C เพื่อคงมาตรฐานของภาษาไว้ไม่ให้เปลี่ยนแปลงไป

### ข้อดีของภาษา C

- เป็นภาษาที่ทำงานได้เร็ว
- เป็นภาษาที่ง่ายต่อการเรียนรู้
- เป็นพื้นฐาน และต้นกำเนิดของภาษาต่าง ๆ เช่น C++, C#, Python, Java
- เป็นภาษาที่ถูกใช้งานมานานแล้ว ทำให้มีตัวอย่างมากมายให้ศึกษา
- หน่วยประมวลผลกลางในหน่วยประมวลผลขนาดเล็ก (Microprocessor) ส่วนใหญ่ในปัจจุบันรองรับภาษาซี

## C Program Structure (โครงสร้างโปรแกรมภาษา C)

โปรแกรมภาษา C มีโครงสร้างแบ่งออกเป็น 6 ส่วนหลัก ๆ ดังนี้

1. Header file (ไฟล์ส่วนหัว)

เป็นส่วนที่มีไว้เขียนคำสั่งพิเศษ เพื่อให้สามารถนำเข้าโปรแกรมที่ปรับปรุงแล้วของคนอื่นมาใช้ในโปรแกรมของเราได้

ในภาษา C เราจะต้องนำเข้าฟังก์ชันพื้นฐานต่าง ๆ จากไฟล์อื่นเข้ามาใช้งาน โดยส่วนใหญ่แล้วจะมีการรวบรวมชุดของฟังก์ชันเบื้องต้นเหล่านี้เอาไว้เป็นไฟล์ แยกตามชนิดของชุดฟังก์ชันนั้น ๆ ซึ่งไฟล์ที่เก็บชุดของฟังก์ชันนี้จะถูกเรียกว่า Library เพราะเปรียบเสมือนห้องสมุดที่เก็บฟังก์ชันต่าง ๆ ไว้ที่เดียว ตัวอย่างของ Library ที่ใช้งานบ่อยมี ดังนี้

| | |
|---|---|
| <stdio.h> | Input/Output Functions |
| <stdlib.h> | General Utility Functions |
| <string.h> | String Function |
| <time.h> | Date and Time Functions |
| <math.h> | Mathematics Functions |
| <signal.h> | Signal Handling Functions |
| <ctype.h> | Testing Characters |
| <errno.h> | Error Handling operations |
| <iostream> | Input and Output streams |
| <float.h> | Floating-point constants |
| <assert.h> | Diagnostics and program debugging |

ซึ่งคำสั่งที่ใช้ในการนำเข้า Library จะมีรูปแบบ ดังนี้

```
#include <Filename.h>
```

2. Definition (การกำหนด)

คำสั่งในส่วนนี้จะใช้ในการกำหนดค่าคงที่ หรือนิพจน์ในโปรแกรม C ของเรา ซึ่งจะขึ้นต้นด้วย # (แฮช) เช่น #define, #ifdef, #endif, #ifndef ในภาษา C คำสั่งทั้งหมดที่ขึ้นต้นด้วยสัญลักษณ์ # (แฮช) จะถูกดำเนินการก่อนคำสั่งอื่น ๆ เสมอ

3. Global Declaration (ส่วนประกาศ)

ส่วนของตัวแปรหรือฟังก์ชัน โดยที่ตัวแปรและฟังก์ชันที่ประกาศในส่วนนี้ สามารถใช้งานได้ในทุกส่วนของโปรแกรม ต่างจากการประกาศตัวแปรแบบอื่นที่จะสามารถเรียกใช้งานได้เฉพาะภายในขอบเขตที่ประกาศตัวแปรเท่านั้น

4. Main Function (ฟังก์ชันหลัก)

โปรแกรมภาษา C มีจุดเข้าที่ฟังก์ชัน main() ซึ่งฟังก์ชันนี้จะถูกเรียกใช้งานก่อนเมื่อเรารัน execution file (.exe) ของโปรแกรม ซึ่งโปรแกรมจะเริ่มทำงานที่วงเล็บปีกกาเปิด ({) และสิ้นสุดการทำงานที่วงเล็บปีกกาปิด (}) ดังนั้นคำสั่งต่าง ๆ ที่ต้องการให้โปรแกรมทำงานจะถูกเขียนภายในฟังก์ชัน main นี้

5. Documentation (การทำเอกสารประกอบโปรแกรม)

ส่วนนี้ประกอบไปด้วยคำอธิบายของโปรแกรม ชื่อของโปรแกรม วัน เวลา และอื่น ๆ โดยจะถูกเขียนอยู่ในรูปของคอมเมนต์ (comment)

ทุกอย่างที่เขียนแบบ comment จะถูกมองว่าเป็นส่วนคำอธิบายของโปรแกรม ซึ่งไม่ส่งผลต่อโปรแกรมของเรา โดยปกติแล้วเอาไว้อธิบายให้ผู้อ่านโปรแกรม

6. Subprogram (โปรแกรมย่อย)

ส่วนนี้มีไว้เพื่อสร้างโปรแกรมย่อยเพิ่มเติมที่อยู่ในรูปของฟังก์ชัน เพื่ออำนวยความสะดวกในการเขียนโปรแกรม และแบ่งการทำงานของโปรแกรมหลักออกเป็นส่วน ๆ

```c
// Header file
#include <stdio.h>

// Definition
#define Pi 3

// Global Declaration
int R = 32;

// Main Function
int main () {
   printf ("Hello ComCamp#%d", Pi + R);
   return 0;
}
```

```
Hello Comcamp#35
```

ในโปรแกรมตัวอย่างจะเป็นการประกาศค่าคงที่ Pi ให้มีค่าเท่ากับ 3 และประกาศตัวแปร R ให้มีค่าเท่ากับ 32 จากนั้นทำการหาผลรวม แล้วนำไปต่อท้ายข้อความ “Hello Comcamp#” ทำให้เราได้ผลลัพธ์เป็น “Hello Comcamp#35”

โดยในตัวอย่างเราได้ใช้คำสั่ง

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

ซึ่งเป็นการนำเข้า Library stdio.h ที่ประกอบไปด้วยฟังก์ชันพื้นฐานต่าง ๆ ที่จำเป็น เช่น ฟังก์ชันในการแสดงผลออกทางจอภาพ หรือ ฟังก์ชันในการรับค่าจากคีย์บอร์ด เป็นต้น

**คำศัพท์ของภาษา C**

- Block (หน่วยเก็บข้อมูล) คือ สิ่งที่กำหนดขอบเขตและการทำงานของโปรแกรม ซึ่งจะเริ่มต้นด้วยเครื่องหมาย ({) และสิ้นสุดด้วย (}) ซึ่งในภาษา C จะถูกใช้ในหลาย ๆ ฟังก์ชัน และนอกจากนี้บล็อกยังสามารถซ้อนกันได้อีกด้วย

```
#include <stdio.h>

int sum (int A, int B) {
   return A + B;
}

int main () {
   int X = 3, Y = 4;

   if (X < Y) {
      printf ("Summation of X and Y equal to %d\n", sum (X, Y));
   }

   return 0;
}
```

- Comment (คำอธิบาย) เป็นส่วนที่ไม่มีผลต่อการทำงานของโปรแกรม ซึ่งสามารถทำได้ 2 วิธี คือการ Comment แบบหลายบรรทัด เริ่มด้วย slash star (/*) ตามด้วยข้อความที่ต้องการเขียน และลงท้ายด้วย star slash (*/) หรือ Comment แบบหนึ่งบรรทัด เริ่มด้วย double slash (//) และหลังจากนั้นข้อความทั้งหมดที่เขียนจนจบบรรทัดนั้นจะถือว่าเป็น Comment

```c
/* This program will find and show
the summation of x and y,
which have already been assigned
before running the program. */
#include <stdio.h>

int main () {
   int X = 4, Y = 8;
   printf ("%d", X + Y); // Show summation of X and Y

   return 0;
}
```

- Semicolon (เครื่องหมายอัฒภาค ;) เป็นสัญลักษณ์ที่ใช้แบ่งแยกคำสั่งภายในโปรแกรม ซึ่งก็คือการจบคำสั่งนั้น ๆ semicolon เป็นสิ่งที่บังคับเพื่อให้ตัวคอมไพเลอร์สามารถแยกแยะคำสั่งในการทำงานได้

```c
int A;
int B = 5, C = 2;
printf ("%d", B + C);
```

- Whitespace (พื้นที่ว่าง) คือ ตัวอักษรหรือเครื่องหมายที่ใช้แบ่งแยกคำสั่งและ Token ซึ่งในภาษา C นั้น Whitespace จะประกอบไปด้วย การเว้นวรรค การย่อหน้า และ การขึ้นบรรทัดใหม่ ซึ่ง Whitespace ที่ต่อกันเป็นจำนวนมากนั้น ไม่มีผลต่อการทำงานของโปรแกรมและ Compiler แต่มันช่วยให้เราสามารถทำโค้ดให้เป็นระเบียบได้ แต่อยากไรก็ตาม Whitespace ยังคงต้องใช้ในบางที่ เช่น ระหว่างคำสั่งของภาษา C และ ชื่อตัวแปร

```c
int A=1;
int A = 1;
int A      =         1;
int A
=
1;
```

### การใช้คำสั่ง return ใน Main Function

- ปกติแล้วบรรทัดสุดท้ายใน Main function จะมีคำสั่ง return ซึ่งเป็นคำสั่งในการส่งค่าออกไปยังนอกไฟล์ โดยค่านี้จะมีประโยชน์เมื่อเราทำ Project ใหญ่ ๆ ที่ต้องมีการส่งข้อมูลข้ามไฟล์ โดยค่า return นี้จะช่วยบอกว่าสถานะการทำงานของไฟล์นี้เป็นอย่างไร โดยเลข return แต่ละตัวก็จะแทนสถานะต่าง ๆ เช่น 0 คือปกติ 1 คือ มี error

## Variable

ตัวแปร คือ การตั้งชื่อพื้นที่ภายในหน่วยความจำ ซึ่งใช้สำหรับเก็บข้อมูลที่ใช้ในการทำงานของโปรแกรม เพื่อความสะดวกในการเรียกใช้ข้อมูล ดังนั้นถ้าจะใช้ข้อมูลใดก็ให้เรียกผ่านชื่อของตัวแปร โดยในภาษา C จะมีการแยกชนิดตัวแปร ซึ่งจะสอดคล้องกับชนิดของข้อมูลที่จะเก็บ ช่วงของค่าที่สามารถเก็บไว้ในพื้นที่หน่วยความจำนั้น และตัวดำเนินการ (Operator) ที่สามารถนำไปใช้กับตัวแปรนั้นได้

ซึ่งชนิดช่วงของค่า และข้อมูลที่เก็บของตัวแปรในภาษา C มี ดังนี้

| ชนิด | จำนวนบิต | ช่วงของค่า | ประเภทของข้อมูลที่เก็บ |
|---|---|---|---|
| char | 8 | ASCII Character (-128 ถึง 127) | ตัวอักษรหนึ่งตัว |
| short int | 16 | -32,768 ถึง 32,767 | จำนวนเต็ม |
| int | 32 | -2,147,483,648 ถึง 2,147,483,647 | จำนวนเต็ม |
| long long int | 64 | $-2^{63}$ ถึง $2^{63}-1$ ($2^{63}$ = 9223372036854775808) | จำนวนเต็ม |
| Unsigned int | 32 | 0 ถึง 4,294,967,295 | เฉพาะจำนวนเต็มบวก |
| float | 32 | 1.2E-38 ถึง 3.4E+38 | จำนวนที่มีทศนิยม 6 ตำแหน่ง |
| double | 64 | 1.7E-308 ถึง 1.7E+308 | จำนวนที่มีทศนิยม 15 ตำแหน่ง |
| void | - | - | ไม่มีชนิด (รอการระบุชนิดภายหลัง) |

การประกาศตัวแปร (Declaration) หรือการสร้างตัวแปร เป็นการบอกคอมไพเลอร์ว่าจะทำงานกับตัวแปรนี้อย่างไรและต้องจองพื้นที่ให้กับตัวแปรตัวนี้เท่าไหร่

ซึ่งในภาษา C การประกาศตัวแปรเราต้องกำหนดประเภทของตัวแปรและชื่อของตัวแปร รูปแบบทั่วไปของการประกาศตัวแปรทำได้ ดังนี้

```
Type VariableName;
```

Type คือ ประเภทของตัวแปรในภาษา C
VariableName คือ ชื่อของตัวแปร

ซึ่งข้อกำหนดการตั้งชื่อตัวแปรมี ดังนี้

- ชื่อของตัวแปรประกอบด้วยตัวอักษร ตัวเลข และเครื่องหมายขีดล่าง (_)
- ต้องขึ้นต้นด้วยตัวอักษรหรือขีดล่าง
- ตัวอักษรตัวพิมพ์ใหญ่และตัวพิมพ์เล็กมีความหมายแตกต่างกัน (เช่น คำว่า Test จะต่างจากคำว่า test)
- ต้องไม่ซ้ำกับ Keywords

**Keywords**

หมายถึงกลุ่มคำที่ถูกสงวนไว้ โดยเราไม่สามารถใช้คำเหล่านี้ในการประกาศเป็นชื่อตัวแปรหรือฟังก์ชันได้ Keywords มาตรฐานในภาษา C มี ดังนี้

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| auto | break | case | char | const | continue | default | do |
| double | else | enum | extern | float | for | goto | if |
| int | long | register | return | short | signed | sizeof | Static |
| struct | switch | typedef | union | unsigned | void | volatile | while |

ตัวอย่างการประกาศตัวแปร เช่น

```
int Num;
char Alphabet;
float Pi;
```

ในตัวอย่างจะเป็นการประกาศตัวแปร 3 ตัว คือ 1) ตัวแปรประเภท Integer ตั้งชื่อว่า Num 2) ตัวแปรประเภท character ชื่อว่า Alphabet และ 3) ประเภท float ชื่อ Pi

จากรูปแบบข้างต้นเราสามารถประกาศตัวแปรประเภทเดียวกันได้หลายตัวพร้อม ๆ กัน ด้วยการคั่นแต่ละตัวด้วยเครื่องหมายจุลภาค (,)

และนอกจากนี้ เรายังสามารถกำหนดค่าเริ่มต้นให้กับตัวแปรตอนประกาศตัวแปรได้ รูปแบบทั่วไปของการประกาศตัวแปรและกำหนดค่าให้พร้อมกันทำได้ ดังนี้

```
Type VariableName = value;
```

ซึ่งหากเราประกาศตัวแปรหลาย ๆ ตัวพร้อมกัน เราไม่จำเป็นต้องกำหนดค่าเริ่มต้นให้กับตัวแปรของเราทุกตัว

ตัวอย่างการประกาศตัวแปร เช่น

```
int Num = 3;
float e, Pi = 3.14;
char Alphabet1 = 'C', Alphabet2 = 'g';
```

ในตัวอย่าง บรรทัดที่ 1 เป็นการประกาศตัวแปร Integer ชื่อ Num แล้วกำหนดให้มีค่าเท่ากับ 3 บรรทัดที่ 2 ประกาศตัวแปร float 2 ตัวชื่อ e กับ Pi โดยให้ Pi มีค่าเท่ากับ 3.14 แต่ e ยังไม่มีการกำหนดค่าให้ สร้างไว้เฉย ๆ บรรทัดที่ 3 ประกาศตัวแปรประเภท character 2 ตัว ชื่อ Alphabet1 ซึ่งมีค่าเท่ากับ A อีกตัวชื่อ Alphabet2 มีค่าเท่ากับ B

จะเห็นได้ว่าตอนอ้างถึง character จะต้องมีเครื่องหมาย Single quotes (') ครอบด้วย เพื่อให้ไม่ซ้ำกับชื่อตัวแปรที่เขียน เดี่ยว ๆ เช่น **ตัวอักษร 'A'** กับ **ตัวแปร A**

## Operator

ตัวดำเนินการ จะถูกใช้กับตัวแปรเพื่อใช้ในการดำเนินการบางอย่าง เช่น การดำเนินการทางคณิตศาสตร์ในภาษา C ตัวดำเนินการจะถูกแบ่งชนิดตามหน้าที่ที่แตกต่างกัน โดยชนิดของตัวดำเนินการมี ดังนี้

1. Assignment operator

ตัวดำเนินการกำหนดค่า ในภาษา C ใช้สัญลักษณ์เท่ากับ (=) เพื่อกำหนดค่าให้กับตัวแปร โดยตัวดำเนินการนี้จะมี 2 ฝั่ง ซึ่งการทำงานของตัวดำเนินการนี้คือการนำผลลัพธ์ที่ได้จากทางด้านขวาไปเขียนทับตัวแปรทางด้านซ้าย เช่น

```
A = 2; //assign 2 to A
B = A; //make value of B equal to A. That is 2
C = A + B; //change C value to A plus B. That is 4
```

2. Arithmetic operator (+, -, *, /, %)

ตัวดำเนินการทางคณิตศาสตร์ คือ ตัวดำเนินการที่ใช้เพื่อกระทำการดำเนินการทางคณิตศาสตร์ระหว่างตัวแปรหรือค่าคงที่ เช่น การบวก การลบ การคูณ การหาร และการหารเอาเศษ (modulo)

| สัญลักษณ์ | ชื่อ | ตัวอย่างการเขียน |
|---|---|---|
| + | addition | c = a + b |
| - | subtraction | c = a – b |
| * | multiplication | c = a * b |
| / | division | c = a / b |
| % | modulo | c = a % b |

3. Compound assignment

คือ ตัวดำเนินการที่ใช้เพื่ออัพเดทหรือแก้ไขค่าปัจจุบันของตัวแปร โดยการใช้ตัวดำเนินการทางคณิตศาสตร์และตัวดำเนินการกำหนดค่าร่วมกัน

| ตัวดำเนินการ | ตัวอย่าง | เทียบเท่ากับ |
|---|---|---|
| += | a += 2 | a = a + 2 |
| -= | a -= 2 | a = a - 2 |
| *= | a *= 2 | a = a * 2 |
| /= | a /= 2 | a = a / 2 |
| %= | a %= 2 | a = a % 2 |

4. Increment and Decrement (++, --)

ตัวดำเนินการเพิ่มและลดค่า คือ ตัวดำเนินการที่ใช้เพื่อบวกหรือลบค่าออกจากตัวแปรด้วย 1 โดยจะใช้เครื่องหมาย ++ หรือ –- ซึ่งการดำเนินการนี้มีทั้งรูปแบบที่ใส่เครื่องหมายด้านหน้าและด้านหลังตัวแปร ดังนี้

**x++, x--**

**++x, --x**

ซึ่งความแตกต่าง คือ หากใส่ด้านหน้าจะเป็นการดำเนินการนำหน้า (Prefix) คือ มีการเปลี่ยนค่าของตัวแปรก่อนดำเนินการคำสั่งในบรรทัดนั้น แต่หากใส่ด้านหลังจะเป็นการดำเนินการตามหลัง (Postfix) คือ มีการทำคำสั่งในบรรทัดนั้นก่อนที่จะมีการเปลี่ยนค่าของตัวแปร

5. Relational and Comparison operator (==, !=, >, <, >=, <= )

ตัวดำเนินการความสัมพันธ์และเปรียบเทียบ คือ ตัวดำเนินการที่ถูกใช้เพื่อหาค่าความจริง ว่าจะเป็นจริง (True) หรือเป็นเท็จ (False) ซึ่งหากมีค่าความจริงเป็น **จริง** จะมีค่าเป็น 1 หากมีค่าความจริงเป็น **เท็จ** จะมีค่าเป็น 0

| ตัวดำเนินการ | ตัวอย่าง | ผลลัพธ์ |
|---|---|---|
| == | a == b | มีค่า 1 ถ้าค่าของ a เท่ากับ b |
| != | a != b | มีค่า 1 ถ้าค่าของ a ไม่เท่ากับ b |
| < | a < b | มีค่า 1 ถ้าค่า a น้อยกว่า b |
| > | a > b | มีค่า 1 ถ้าค่าของ a มากกว่า b |
| <= | a <= b | มีค่า 1 ถ้าค่าของ a ไม่มากกว่า b |
| >= | a >= b | มีค่า 1 ถ้าค่าของ a ไม่น้อยกว่า b |

## Operator precedence

ลำดับความสำคัญของตัวดำเนินการ (Operator precedence) คือ ลำดับที่มีไว้เพื่อกำหนดว่าตัวดำเนินการใดที่จะถูกทำงานก่อน ยกตัวอย่าง เช่น การทำงานของนิพจน์ 6+3x4 ผลลัพธ์จะต้องได้เป็น 18 เพราะว่าตัวดำเนินการ (x) นั้นมีความสำคัญมากกว่าตัวดำเนินการ (+) เราสามารถบังคับให้การบวกทำงานก่อนได้โดยใช้วงเล็บ (6+3)x4 เพื่อให้การบวกสามารถทำงานก่อนได้ ซึ่งจะได้ผลลัพธ์เป็น 36 เป็นต้น

ตารางเรียงลำดับความสำคัญของตัวดำเนินการจากมากไปน้อย (บนลงล่าง)

| ชื่อ | ตัวดำเนินการ |
|---|---|
| Primary | (), [], ., -> |
| Unary | ++x, --x |
| Multiplicative | *, /, % |
| Additive | +, - |
| Relational | <, >, <=, >= |
| Equality | ==, != |
| Assignment | =, +=, -=, *=, /=, %=<br>x++, x-- |

## Output & Input

ภาษา C มี Library พื้นฐานที่ทำให้เราสามารถรับค่าและแสดงผลในโปรแกรมของเราได้ นั่นคือ Library stdio.h (standard input output library)

## Format specifier

คือ สิ่งที่ใช้ระบุชนิด หรือรูปแบบของข้อมูลที่ต้องการส่งออกหรือรับเข้ามา ซึ่ง specifier ที่ควรรู้มี ดังนี้

| Format specifier | คำอธิบาย | การใช้งาน |
|---|---|---|
| %c | Character | ใช้กับอักษรหนึ่งตัว |
| %s | String of characters | ใช้กับสตริงหรือตัวอักษรหลายตัว |
| %d | Integer | ใช้กับจำนวนเต็ม |
| %ld | Long integer | ใช้กับจำนวนเต็มแบบยาว |
| %u | Unsigned int | ใช้กับจำนวนเต็มบวก |
| %e | Exponential | ใช้แสดงจำนวนในรูปแบบเลขยกกำลัง |
| %f | Float | ใช้กับจำนวนจริง |
| %lf | Double | ใช้กับจำนวนจริงแบบยาว |
| %p | Pointer | ใช้แสดงค่าของตำแหน่งที่อยู่ในหน่วยความจำ |

## Output

การแสดงผลในภาษา C มีฟังก์ชันที่ใช้คือ ฟังก์ชัน printf() (print format) ซึ่งเป็นฟังก์ชันที่สามารถแสดงข้อมูลประเภทต่าง ๆ ออกทางหน้าจอภาพได้ ตัวฟังก์ชันมีรูปแบบการใช้งานดังนี้

```
printf ("string pattern", variable1, variable2, ....);
```

ในการใช้ฟังก์ชัน ตรงส่วนของ **string pattern** จะเป็นส่วนที่เราจะใส่ข้อความที่เราต้องการให้แสดงผล โดยหากเราต้องการแสดงค่าของตัวแปร ส่วนนี้เราจะต้องใส่ format specifier หลังจากนั้นในส่วนของ **variable** เราจะต้องใส่ชื่อของตัวแปรเรียงตาม format specifier ที่เราใส่ไปก่อนหน้า โดยจะมีการคั่น variable แต่ละตัวด้วยเครื่องหมายจุลภาค (,) และมีการคั่นส่วนของ string pattern กับส่วนของ variable ด้วยเครื่องหมายจุลภาค (,) เช่นกัน

```c
printf ("Today is so funny\n");

int number = 35;
printf ("It's ComCamp#%d\n", number);
```

```
printf ("If you use wrong format. It will be like this %f", number);
```

```
Today is so funny
It's Comcamp35
If you use wrong format. It will be like this 0.000000
```

จากตัวอย่าง จะเป็นการใช้ฟังกัน printf() แสดงผลแบบต่าง ๆ คำสั่งแรก คือ การแสดงข้อความ คำสั่งที่สองจะเป็นการแสดงตัวเลขที่เป็น int จึงมีการใช้ %d เพื่อแสดงผล คำสั่งสุดท้ายจะเป็นการแสดงให้เห็นว่าหากเราใช้ format specifier ผิด อาจจะทำให้เกิดความผิดพลาดได้

สำหรับ %f หรือ %lf หากเราต้องการระบุตำแหน่งของทศนิยมที่จะแสดง เราสามารถใช้ precision specifierโดยใส่ไว้หลังเครื่องหมาย modulo (%) เช่น

```
double Pi = 3.141592653589;

//Without using precision specifier
printf ("%lf\n", Pi);

//Using precision specifier
printf ("%.2lf\n", Pi);
printf ("%.15lf", Pi);
```

```
3.141593
3.14
3.141592653589000
```

จากตัวอย่าง เนื่องจากตัวแปรเรามีประเภทเป็น double เราจึงต้องใช้ %lf โดยอันแรกเป็นการแสดงผลแบบไม่ระบุอะไร เราจะได้ทศนิยม 6 ตำแหน่ง (default) อันที่สองเป็นการระบุให้แสดงแค่ 2 ตำแหน่ง อันสุดท้ายเป็นการระบุให้แสดง 15 ตำแหน่ง ซึ่งตัวแปรของเรามีเพียง 12 ตำแหน่ง ดังนั้นหลังจากตำแหน่งที่ 12 จึงเป็นเลข 0

## Escape Character

เป็นตัวอักษรพิเศษในภาษา C ที่จะต้องใช้เครื่องหมาย back slash (\) นำหน้าเสมอเพื่อไม่ให้เกิดข้อผิดพลาดในการคอมไพล์โปรแกรม เพราะว่าตัวอักษรนี้เป็นสัญลักษณ์ที่ตรงกับโครงสร้างของภาษา รายการของตัวอักษรพิเศษในภาษา C มีดังนี้

| Escape sequence | ชื่อ | ความหมาย |
|---|---|---|
| \n | New Line | การขึ้นบรรทัดใหม่ |

| | | |
|---|---|---|
| \t | Horizontal Tab | การย่อหน้า |
| \' | Single quotes | แสดงเครื่องหมาย ' |
| \" | Double quotes | แสดงเครื่องหมาย " |
| \\ | Back Slash | แสดงเครื่องหมาย \ |
| **\0** | NULL value | ค่า NULL character |

ตัวอย่างการใช้งาน Escape character

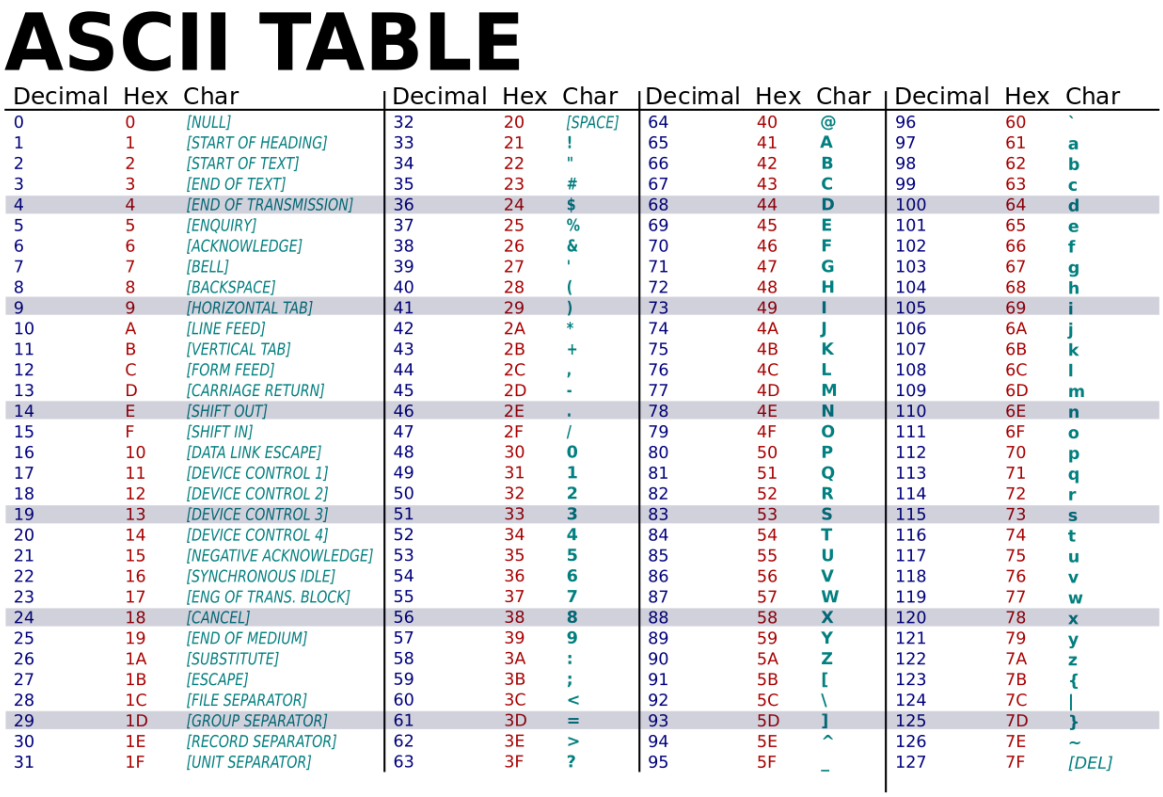
```
Printf ("-\tWOW\t-");

char Newline = '\n';
printf ("%cWhat is \"1+1\" equal to?", Newline);
```

```
-	WOW	-
What is "1+1" equal to?
```

จากตัวอย่าง การใช้ฟังก์ชัน printf() ครั้งแรก จะเป็นตัวอย่างของการใช้ \t เพื่อย่อหน้าเข้าไปก่อนที่จะแสดงคำว่า “WOW” ส่วนครั้งที่สอง เป็นการแสดงให้เห็นว่าตัวแปรประเภท Character สามารถเก็บ Escape character และแสดงผลได้เหมือนกัน

**ASCII Table (American Standard Code for Information Interchange)**



# ASCII TABLE

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | [NULL] | 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | [START OF HEADING] | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | [START OF TEXT] | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | [END OF TEXT] | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | [END OF TRANSMISSION] | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | [ENQUIRY] | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | [ACKNOWLEDGE] | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | [BELL] | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | [BACKSPACE] | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | [HORIZONTAL TAB] | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | A | [LINE FEED] | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | B | [VERTICAL TAB] | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | C | [FORM FEED] | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | D | [CARRIAGE RETURN] | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | E | [SHIFT OUT] | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | F | [SHIFT IN] | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | [DATA LINK ESCAPE] | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | [DEVICE CONTROL 1] | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | [DEVICE CONTROL 2] | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | [DEVICE CONTROL 3] | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | [DEVICE CONTROL 4] | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | [NEGATIVE ACKNOWLEDGE] | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | [SYNCHRONOUS IDLE] | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | [ENG OF TRANS. BLOCK] | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | [CANCEL] | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | [END OF MEDIUM] | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | [SUBSTITUTE] | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | [ESCAPE] | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | [FILE SEPARATOR] | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | [GROUP SEPARATOR] | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | [RECORD SEPARATOR] | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | [UNIT SEPARATOR] | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

ที่มา : https://www.businessinsider.com/the-martian-hexidecimal-language-2015-9

คือ มาตรฐานของการแปลงไปแปลงมาระหว่าง เลขฐาน 16 ฐาน 10 กับตัวอักษรภาษาอังกฤษ ตัวอย่างการใช้งาน เช่น

```
int Example1 = 'A' + 3;
printf ("%d ", Example1);

char Example2 = 'A';
printf ("%d ", Example2);

int Example3 = 65;
printf ("%c", Example3);
```

```
68 65 A
```

จากตัวอย่าง จะสังเกตได้ว่าเราสามารถใช้ตัวอักษรแทนตัวเลขหรือใช้ตัวเลขแทนตัวอักษรได้เลย นอกจากนี้ในตัวอย่างจะเห็นว่า มีการใช้ %c กับ integer และมีการใช้ %d กับ character ได้อีกด้วย

## Input

ในการรับค่านั้นเราจะใช้ฟังก์ชัน scanf() (scan specifier) ในการรับค่าจากคีย์บอร์ด โดยรูปแบบการใช้งานของฟังก์ชันเป็น ดังนี้

```
scanf ("%specifier1%specifier2...", variable1, variable2, ....);
```

ฟังก์ชัน scanf() นั้นมีรูปแบบที่คล้ายกันกับฟังก์ชัน printf() โดยในส่วนแรกจะเป็นลำดับ format specifier ที่ต้องการให้เป็น input หลังจากนั้นในส่วน variable จะเป็นลำดับของตัวแปรที่ตรงกับ specifier ที่เรากำหนดไว้ด้านหน้า โดยที่ชื่อตัวแปรแต่ละตัวที่อยู่ตรงนี้จะต้องมีสัญลักษณ์ ampersand (&) ด้านหน้าเพื่อบ่งบอกถึงที่อยู่ของตัวแปร และเหมือนกันกับ printf() ที่จะต้องมีเครื่องหมายจุลภาค (,) คั่นระหว่าง variable แต่ละตัว และคั่นระหว่างส่วนแรกกับส่วนที่สอง

## Address-of Operator (&)

คือ ตัวดำเนินการที่แทนด้วยสัญลักษณ์ ampersand (&) เมื่อถูกใช้จะได้รับค่าที่อยู่ของตัวแปรในหน่วยความจำ โดยค่าที่อยู่นี้เราสามารถได้รับมาได้ในตอนที่โปรแกรมรันเท่านั้น ซึ่งค่านี้จะบอกว่าตัวแปรที่เก็บอยู่ที่ตำแหน่งไหนของหน่วยความจำ ตัวอย่างการเขียน เช่น

```
&MyVariable;
```

การใช้คำสั่งนี้ เราจะได้รับค่าที่อยู่ของตัวแปร MyVariable โดยค่าที่อยู่นั้นส่วนใหญ่จะอยู่ในรูปแบบของเลขฐานสิบหก (ในหนังสือบางเล่ม หรือตอนเรียนอาจจะใช้เลขฐาน 10 เพื่อให้ง่าย)

```
float a, b;
scanf ("%f%f", &a, &b);

float sum = a + b;
printf ("Sum of a and b is %.2f\n", sum);
```

```
Input: 6.4 8.5
```

```
Sum of a and b is 14.90
```

ในตัวอย่างจะเป็นการรับค่า float เข้าไป 2 ตัว ก่อนจะหาผลบวก และแสดงค่าผลบวกของ 2 จำนวนออกมาทางหน้าจอในรูปแบบทศนิยม 2 ตำแหน่ง

## Buffer

คือ ส่วนหนึ่งของหน่วยความจำที่ใช้สำหรับพักข้อมูลเป็นการชั่วคราวในระหว่างการส่งข้อมูล เช่น หากเราเขียนโปรแกรมแล้วมีการรับค่าจากที่ผู้ใช้งาน 2 ตัว แต่ผู้ใช้งานป้อนเกินเป็น 3 หรือ 4 ตัว โปรแกรมของเราจะนำ 2 ค่าแรกไปใช้ ส่วนที่เหลือก็จะค้างอยู่ใน Buffer

Buffer จะไม่มีผลในการเขียนโปรแกรมมาก ยกเว้นในเคสที่เรามีการ scanf() ค่าบางอย่าง และตามด้วยการ scanf() character เข้ามา หากการป้อนข้อมูลก่อนหน้ามีการกด enter นั่น หมายความว่าตัวอักษร \n จะเข้ามาใน Buffer ด้วย แล้วตัว scanf() character จะไปทำตัว \n เข้าแทนที่จะเป็นตัวที่เราต้องการ ซึ่งวิธีแก้ปัญหาก็ คือ ให้เราเว้นวรรคด้านหน้า format specifier ไว้

ตัวอย่างโปรแกรมที่มีปัญหา

```
int a;
char b;
scanf ("%d%c", &a, &b);
printf ("A%d%cA", a, b);
```

ตัวอย่างการแก้ปัญหาเพื่อทำให้สามารถใส่ Input ตามที่ต้องการ

```
int a;
char b;
scanf ("%d %c", &a, &b);
printf ("A%d%cA", a, b);
```

## Condition

คือ คำสั่งควบคุมที่ใช้ในการสร้างการตัดสินใจให้กับโปรแกรม ซึ่งจะมีหลาย ๆ คำสั่งให้ใช้ตามแต่สถานการณ์ที่เจอ ดังนี้

### if

เป็นคำสั่งพื้นฐานสำหรับควบคุมการทำงานของโปรแกรมเพื่อให้ทำงานตามเงื่อนไขที่กำหนด หรือสร้างทางเลือกการทำงานเพิ่มเติมให้กับโปรแกรม โดยรูปแบบการใช้งานคำสั่ง if จะเป็นดังนี้

```
if (condition) {
   // statement1
   // statement2
   ...
}
```

การใช้งานคำสั่ง if เราจะเขียนโค้ดคำสั่งที่จะให้ทำงานภายในวงเล็บ { } และระบุเงื่อนไขใน condition ซึ่งเป็นสิ่งที่ต้องมีสำหรับคำสั่ง if ซึ่งคำสั่งด้านในบล็อคจะทำงานเมื่อเงื่อนไขเป็นจริง โดย

ภายในบล็อคเราสามารถเขียนคำสั่งหนึ่งหรือหลายคำสั่งก็ได้ และหากว่าเราเขียนเพียงหนึ่งคำสั่ง เราสามารถละเว้นวงเล็บปีกกาออกไปจากคำสั่ง if ได้ ยกตัวอย่างเช่น

```
if (condition)
   // statement
```

ตัวอย่างของการใช้งานคำสั่ง if

```
int T = 9
if (T % 2 != 0) {
   printf ("%d is an odd number\n", T);
}
if (T < 10 && T > 5) {
   printf ("This number is between 5 and 10\n");
}
if (T > 20) {
   printf ("This number is more than 20\n");
}
```

```
9 is an odd number
This number is between 5 and 10
```

จากตัวอย่าง จะเป็นการประกาศตัวแปร T ให้มีค่าเท่ากับ 9 จากนั้นจะมีการเช็คว่า T modulo 2 ไม่ได้มีค่าเป็น 0 ใช่หรือไม่ ซึ่งในกรณีนี้ T modulo 2 จะมีค่าเท่ากับ 1 ซึ่งไม่เท่ากับ 0 ดังนั้นจึงมีการทำคำสั่งภายในบล็อก มีการแสดงข้อความบอกว่าเลข 9 เป็นเลขคี่ ต่อจากนั้นก็มีการเช็คว่า T น้อยกว่า 10 และมากกว่า 5 หรือไม่ ซึ่งก็เป็นจริง ทำให้มีการทำงานในบล็อค และส่งผลให้มีการแสดงข้อความทางหน้าจอว่า จำนวนนี้อยู่ระหว่าง 5 และ 10 และในเงื่อนไขสุดท้าย มีการเช็คว่า T มีค่ามากกว่า 20 หรือไม่ ซึ่งไม่เป็นจริง ทำให้ไม่มีการทำงานภายในบล็อกนี้เกิดขึ้น

## if else

คำสั่ง else ใช้กำหนดการทำงานของโปรแกรมเมื่อเงื่อนไขเป็นเท็จ ต่างจากคำสั่ง if ที่ใช้กำหนดการทำงานของโปรแกรมเมื่อเงื่อนไขเป็นจริง โดยคำสั่ง else จะต้องใช้ร่วมกับคำสั่ง if เสมอ รูปแบบการใช้งาน if else เป็น ดังนี้

```
if (condition)
   // statements when condition is true
} else {
   // statements when condition is not true
}
```

ในการใช้งาน เราจะกำหนดเงื่อนไขใน condition ให้กับคำสั่ง if ถ้าเงื่อนไขเป็นจริง โปรแกรมจะทำงานในบล็อกของคำสั่งของ if ถ้าไม่เป็นจริงโปรแกรมจะทำงานในบล็อกคำสั่งของ else

```
int N;
printf ("Please enter number: ");
scanf (" %d", &N);

if (N % 2 == 0) {
   printf ("\n%d is an even number\n");
} else {
   printf ("\n%d is and odd number\n");
}
printf ("Now program has ended");
```

Input: 7

```
Please enter number: 7
7 is an even number
Now program has ended
```

ตัวอย่างด้านบนนี้จะเป็นการรับจำนวนจากผู้ใช้งานเข้ามา แล้วแสดงข้อความบอกว่าจำนวนนั้นเป็นจำนวนคี่หรือจำนวนคู่ โดยจะเช็คด้วยการนำจำนวนที่ผู้ใช้งานป้อน มาหารด้วย 2 หากมีเศษแสดงว่าเป็นจำนวนคี่หากไม่มีหรือเศษเท่ากับ 0 แสดงว่าจำนวนนั้นเป็นจำนวนคู่

## if else-if

ในกรณีที่การทำงานของโปรแกรมของเรามีทางเลือกที่เป็นไปได้มากกว่าสองทาง เราต้องใช้คำสั่ง else-if เพื่อสร้างทางเลือกให้กับโปรแกรมเพิ่มเติม โดยคำสั่ง else-if จะต้องใช้คู่กับ if เหมือนกัน รูปแบบการใช้งานเป็น ดังนี้

```
if (condition1) {
  // statements for condition1
} else if (condition2) {
  // statements for condition2
} else if (condition3) {
  // statements for condition3
} else {
  // when no previous conditions are fulfilled
}
```

โดยโปรแกรมจะเริ่มตรวจสอบจากเงื่อนไขแรกใน condition1 ไปจนถึง condition3 และจะทำงานในบล็อคแรกที่มีเงื่อนไขเป็นจริง กล่าวคือ โปรแกรมจะเช็คเงื่อนไขใน condition1 ก่อน ถ้าเป็นจริงจะทำงานในบล็อกนี้ ถ้าไม่เป็นจริงจะลงไปเช็คเงื่อนไขที่ condition2 ถ้าเป็นจริงจะทำงานในบล็อกนี้ ถ้าไม่เป็นจริงจะลงไปเช็คด้านล่างต่อ จนสุดท้ายถ้าไม่มีเงื่อนไขใดเป็นจริงแล้วในคำสั่งของเรามีคำสั่ง else โปรแกรมจะทำงานในบล็อคของ else

จากการเขียนข้างต้น เราสามารถเพิ่มเงื่อนไขเพิ่มเติมเข้ามาเท่าไหร่ก็ได้ด้วยคำสั่ง else if และสุดท้ายด้านล่าง จะมีคำสั่ง else หรือไม่ก็ได้

```
int Num1, Num2;
printf ("Two Number Comparison Program\n");
printf ("Enter numbers: ");
scanf (" %d %d ", &Num1, &Num2);

if (Num1 == Num2){
   printf ("They are equal");
} else if (Num1 > Num2) {
   printf ("%d is greater than %d", Num1, Num2);
} else {
   printf ("%d is greater than %d", Num2, Num1);
}
```

```
Input: 10 5
```

```
Two Number Comparison Program
Enter numbers: 10 5
10 is greater than 5
```

```
Input: 8 8
```

```
Two Number Comparison Program
Enter numbers: 8 8
They are equal
```

ในตัวอย่างจะเป็นโปรแกรมที่รับเลข 2 ตัวจากผู้ใช้งาน แล้วบอกผ่านหน้าจอว่าเลขทั้ง 2 ตัวเท่ากัน หรือเลขตัวไหนมากกว่ากัน โดยจะมีการเช็คเงื่อนไข 3 รอบ รอบแรกเช็คว่า เลข 2 ตัวที่ป้อนเข้ามาเท่ากันหรือไม่ หากไม่เท่ากันจะเช็คว่าตัวที่ 1 มากกว่าตัวที่ 2 หรือไม่ หากไม่เป็นจริง ก็หมายความว่าเลขตัวที่ 2 มากกว่าตัวที่ 1 หากเช็คแล้วตรงกับเงื่อนไขไหนก็ทำงานในบล็อกนั้น

นอกจากการเขียน if, if else หรือ if else-if ในตัวอย่างก่อนหน้านี้แล้ว การเขียน if else ยังสามารถซ้อนกันได้อีก หากซ้อนกันก็จะมีการตรวจสอบเงื่อนไขด้านนอกก่อน หากเป็นจริงก็จะตรวจสอบเงื่อนไขด้านใน ตัวอย่างเช่น

```c
int Num1, Num2, Num3;
printf ("Enter 3 numbers: ");
scanf (" %d %d  %d", &Num1, &Num1, &Num1);

if (Num1 >= Num2){
   if (Num1 >= Num3) {
      printf ("Greatest number is %d", Num1);
   } else { // Case Num1 < Num3
      printf ("Greatest number is %d", Num3);
} else { // Case Num2 > Num1
   if (Num2 >= Num3) {
      printf ("Greatest number is %d", Num2);
   } else { // Case Num1 < Num3
      printf ("Greatest number is %d", Num3);
```

```
Input: 10 5 8
```

```
Enter 3 numbers: 10 5 8
Greatest number is 10
```

```
Input: 2 4 9
```

```
Enter 3 numbers: 2 4 9
Greatest number is 9
```

## switch case

เป็นคำสั่งควบคุมเพื่อให้โปรแกรมทำงานแบบหลายทางเลือก คล้ายคำสั่ง if else-if แต่การใช้งานจะเรียบง่ายและมีข้อจำกัดมากกว่า เพราะในการกำหนดเงื่อนไขจะทำได้แค่การเปรียบเทียบความเท่ากันเท่านั้น (==) รูปแบบการใช้ switch case เป็นดังนี้

```
switch (input) {
case Value1 :
   // statements
   break;
case Value2 :
   // statements
   break;
case Value3 :
   // statements
   break;
default :
   // statements
}
```

การใช้งานคือ เราจะใส่ตัวแปรที่จะนำค่าไปเปรียบเทียบไว้ในช่อง Input จากนั้นภายในวงเล็บปีกกา ({ }) แต่ละบล็อก จะมีส่วนที่เป็นโครงสร้างของ switch case ก็คือมีคำสั่ง case ตามด้วยค่า Value และมีเครื่องหมายโคลอน (:) ปิดท้าย โดย Value จะเป็นค่าที่เราจะนำค่าของ Input มาเปรียบเทียบความเท่ากัน หาก Input == Value ก็จะทำคำสั่งในบล็อกนั้น

และเนื่องจากคำสั่ง switch case ไม่ได้ใช้วงเล็บ { } สำหรับกำหนดขอบเขตของบล็อกแต่ละบล็อก ดังนั้นเราต้องจบการทำงานของแต่ละบล็อคด้วยคำสั่ง break เสมอ ถ้าหากละเว้น โปรแกรมจะทำงานคำสั่งที่เหลือทั้งหมดของ case อื่นด้านล่างด้วย ซึ่งตรงนี้ก็สามารถนำไปประยุกต์ใช้ให้เป็นประโยชน์ได้

ในส่วนของ default ก็จะมีการทำงานเหมือนคำสั่ง else ก็คือหากไม่ตรง case ด้านบนเลย โปรแกรมจะทำคำสั่งในบล็อคนี้

เนื่องจากคำสั่ง switch จะเปรียบเทียบ input กับ Value ในแต่ละ case ด้วย (==) เท่านั้น ดังนั้นแต่ละ case Value จะต้องเป็นค่าคงที่ของ int หรือ char **เท่านั้น** ไม่สามารถใช้ตัวแปรหรือข้อมูลประเภทอื่นได้ เช่น

```
case a :
case 1.5 :
case "hello" :
```

ทั้งสาม case นี้ไม่สามารถทำได้ เนื่องจากเป็นตัวแปร ตัวเลขทศนิยม และ String

ตัวอย่างการใช้งานคำสั่ง switch case

```
int Rating;
printf ("Enter 1-5 to rate this book: ");
scanf (" %d", &Rating);

switch (Rating) {
case (1) :
case (2) :
   printf("Bad");
   break;
case (3) :
   printf("Average");
   break;
case (4) :
   printf("Good");
   break;
case (5) :
   printf("Excellent");
   break;
default :
   printf("Invalid Value");
}
```

```
Input: 4
```

```
Enter 1-5 to rate this book: 4
Good
```

```
Input: 8
```

```
Enter 1-5 to rate this book: 8
Invalid Value
```

```
Input: 1
```

```
Enter 1-5 to rate this book: 1
Bad
```

ในตัวอย่างจะเป็นโปรแกรมให้คะแนนหนังสือ โดยจะแสดงข้อความตาม Rating ที่ผู้ใช้ให้ โดยในตัวอย่างจะเห็นว่า case ที่ Rating = 1 จะไม่มีทั้งคำสั่งด้านในและไม่มีการ break ซึ่งจะทำให้ผลลัพธ์เมื่อผู้ใช้งานป้อนเลข 1 หรือ 2 เข้ามาจะเหมือนกัน คือจะแสดงคำว่า Bad

## Ternary Operator

คือ คำสั่ง if else แบบย่อ แต่จะมีข้อกำหนดคือ เมื่อเงื่อนไขเป็นจริงหรือเท็จจะส่งค่าได้แค่ 1 ตัว รูปแบบการใช้งานเป็น ดังนี้

```
condition ? value_if_true : value_if_false
```

รูปแบบการเขียน คือ จะใส่เงื่อนไขตรง condition จากนั้นจะตามด้วยเครื่องหมายคำถาม (?) และตามด้วยค่าที่อยากให้เป็น อาจจะ 1 หรือ 2 ค่าก็ได้ ซึ่งการทำงานคือจะมีการเช็คเงื่อนไขใน condition ก่อน หากเป็นจริงจะได้ค่าด้านหน้า หากเป็นเท็จจะได้ค่าด้านหลัง เช่น

```c
int a = 10, b = 20, c;
c = (a < b) ? a : b;
printf ("%d", c);
```

```
10
```

ในตัวอย่างจะมีการเช็คว่า a < b หรือไม่ ซึ่งเป็นจริงทำให้ c = 10

## Loop

เป็นคำสั่งเพื่อควบคุมให้โปรแกรมดำเนินการภายในบล็อกหลาย ๆ ครั้ง ซึ่งโดยทั่วไปจะมีลักษณะการใช้งานอยู่ 2 แบบ คือ สั่งให้โปรแกรมการทำงานวนซ้ำ ๆ แบบเดิมเป็นจำนวนครั้งตามที่เราต้องการ กับสั่งให้โปรแกรมทำงานเดิม ๆ ซ้ำเรื่อย ๆ จนกว่าเงื่อนไขบางอย่างจะเป็นจริง

## while loop

คำสั่ง while loop เป็นคำสั่งวนซ้ำที่เป็นพื้นฐานและเรียบง่ายที่สุด คำสั่งนี้ใช้สำหรับควบคุมจนกว่าเงื่อนไขที่กำหนดจะเป็นเท็จ โดยรูปแบบการใช้งานคำสั่ง while loop เป็น ดังนี้

```
while (condition) {
   // statements
}
```

ในคำสั่ง while loop นั้นประกอบไปด้วยสองส่วนคือ ส่วนของ condition เป็นส่วนที่ใส่เงื่อนไขที่จะทำให้โปรแกรมทำการวน Loop ต่อ และส่วน statement ที่จะใส่คำสั่งต่าง ๆ ที่เราต้องการให้โปรแกรมทำงานในขณะที่อยู่ระหว่างการวน Loop ซึ่งส่วนนี้กำหนดขอบเขตด้วยบล็อค ({ })

```
printf ("Prime Number Checker Program\n");
printf ("Enter a number: ");

int Num;
scanf ("%d", &Num);

int divisible = 0;
int i = 1;
while (i <= Num) {
   if (Num % i == 0) {
      divisible++;
   }
   i++;
}

if (divisible == 2) {
  printf ("%d is prime number", Num);
} else {
```

```
  printf ("%d is not prime number", Num);
}
```

Input: 7

```
Prime Number Checker Program
Enter a number: 7
7 is prime number
```

Input: 24

```
Prime Number Checker Program
Enter a number: 24
24 is not prime number
```

ในตัวอย่าง หลังจากที่รับค่าจากผู้ใช้งานแล้ว เราได้ประกาศให้ตัวแปร i มีค่าเท่ากับ 1 จากนั้นจะวนใน while loop จนกว่า i จะมีค่ามากกว่าตัวเลขที่ผู้ใช้งานป้อน โปรแกรมจะวน Loop ไปเรื่อย ๆ โดยทุกครั้งที่ทำงานจะมีการเช็คว่าตัวเลขที่ผู้ใช้งานป้อนหารด้วย i ลงตัวหรือไม่ ถ้าหารลงก็เพิ่มค่าของตัวแปร divisible ด้วย 1 และสุดท้ายก่อนถึง } มีการเพิ่มค่า i ขึ้นอีก 1

## for loop

คำสั่ง for loop เป็นคำสั่งวนซ้ำที่ใช้สำหรับควบคุมให้โปรแกรมทำงานซ้ำ ๆ ในจำนวนรอบที่แน่นอน รูปแบบการใช้งานคำสั่ง for loop เป็น ดังนี้

```
for (initial; condition; update) {
   // statements
}
```

ในการใช้งาน เริ่มต้นด้วยการเขียนคำสั่ง for ตามด้วยวงเล็บ ( ) ที่เป็นส่วนสำหรับกำหนดลักษณะการทำงานของคำสั่ง ซึ่งประกอบไปด้วย 3 ส่วน คือ

- **Initial:** เป็นส่วนที่ใช้กำหนดค่าเริ่มต้นให้กับ Loop ทำงานครั้งแรกก่อน Loop เริ่มทำงาน และทำแค่ครั้งเดียวไม่ทำซ้ำ
- **condition:** เป็นการกำหนดเงื่อนไขที่จะให้ Loop ทำงานต่อไป ซึ่งทุกครั้งที่การทำงานใน Loop เสร็จสิ้น โปรแกรมจะตรวจสอบเงื่อนไขเพื่อดูว่าจะทำต่อหรือไม่
- **update:** เป็นส่วนสำหรับอัพเดทค่าในตัวแปรและอื่น ๆ เมื่อ Loop จบการทำงานในแต่ละรอบ

โดยทั้ง 3 ส่วนจะคั่นด้วยเครื่องหมาย Semicolon (;) นอกจากนี้ทั้ง 3 ส่วนยังเป็นทางเลือก เราสามารถปล่อยให้บางส่วนว่างได้ แต่ยังต้องมีเครื่องหมาย Semicolon (;) อยู่ และในบล็อค { } ของคำสั่ง for จะเป็นส่วนที่โปรแกรมมีการทำซ้ำ

ตัวอย่างการใช้งาน เช่น

```
int N;
printf ("Enter a number: ");
scanf ("%d", &N);
printf ("Multiplication table of %d\n", N);

for (int i = 1; i <= 12; i++) {
   int Product = N * i
   printf ("%d x %d = %d\n", N, i, Product);
}
```

Input: 5

```
Enter a number: 5
Multiplication table of 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
5 x 12 = 60
```

ตัวอย่างจะเป็นโปรแกรมแสดงตารางสูตรคูณ ซึ่งขั้นตอนการทำงานเป็นดังนี้ หลังจากรับค่าจากผู้ใช้งาน เราใช้คำสั่ง for loop สำหรับนับ 1-10 แล้วนำเลขที่นับนี้มาใช้งาน โดยจากตัวอย่างจะเห็นส่วนต่าง ๆ ของ for loop ดังนี้

- **initial:** ก่อน Loop เริ่มทำงาน เราได้ประกาศตัวแปร i และกำหนดให้มีค่าเป็น 1
- **condition:** เงื่อนไขของ Loop คือ i <= 10 หมายความว่าโปรแกรมนี้จะทำงานเรื่อย ๆ จนกว่าค่าของ i จะน้อยกว่าหรือเท่ากับ 10

- **update:** เมื่อจบการทำงานของ Loop ในแต่ละรอบ เราได้เพิ่มค่าของ i ขึ้นไปหนึ่งด้วยคำสั่ง i++

## do-while loop

คำสั่ง do-while loop เป็นคำสั่งวนซ้ำที่ใช้สำหรับควบคุมเพื่อให้โปรแกรมทำงานซ้ำภายใต้เงื่อนไขที่กำหนด แต่จะแตกต่างจากคำสั่ง while loop ตรงที่โปรแกรมจะทำงานใน Loop 1 รอบก่อนเสมอ รูปแบบของคำสั่งเป็นดังนี้

```
do {
   // statements
} while (condition);
```

ในการเขียน เราจะเริ่มต้นด้วยคำสั่ง do ตามด้วยบล็อคของวงเล็บ ({ }) ที่จะประกอบไปด้วยคำสั่งที่ต้องการให้โปรแกรมทำงานในขณะที่กำลัง Loop และ condition คือ เงื่อนไขที่จะให้โปรแกรมทำงานต่อ ซึ่งจะเห็นได้ว่าเงื่อนไขที่กำหนดจะอยู่ส่วนท้ายของ Loop นั่นหมายความว่า โปรแกรมจะทำงานในบล็อกก่อนหนึ่งรอบเสมอ และสำหรับคำสั่ง do-while loop ให้ระวังไว้ว่าหลังเครื่องหมายปีกกาปิดตรง condition จะต้องมีเครื่องหมาย Semicolon (;) ด้วย

```
printf ("Program for finding sums of Integer\n");
int Sum = 0;
int Temp;

do {
   printf ("Enter number to continue or 0 to exit program: ");
   scanf ("%d", &Temp);
   sum += Temp;
} while (Temp != 0);

  printf ("Sum of all entered numbers is %d", Sum);
```

```
Input: 2 4 7 -1 8 0
```

```
Program for finding sums of Integer
Enter number to continue or 0 to exit program: 2
Enter number to continue or 0 to exit program: 4
Enter number to continue or 0 to exit program: 7
Enter number to continue or 0 to exit program: -1
Enter number to continue or 0 to exit program: 8
Enter number to continue or 0 to exit program: 0
Sum of all entered numbers is 20
```

จากตัวอย่างจะเป็นโปรแกรมคำนวณหาผลบวกของเลขที่ผู้ใช้งานใส่เข้ามา ซึ่งจะเห็นได้ว่าผู้ใช้งานสามารถใส่เลขได้เรื่อย ๆ จนกว่าจะใส่เลข 0 เพื่อจบการทำงานและหาผลรวม

และเหมือนกันกับคำสั่ง if else คำสั่งเกี่ยวกับ Loop ก็สามารถซ้อนทับกันได้ เช่น

```c
for (int i = 1; i <= 3; i++) {
   for (int j = 1; j <= 3; j++) {
      printf ("i = %d, j = %d\n", i, j);
   }
}
```

```
i = 1, j = 1
i = 1, j = 2
i = 1, j = 3
i = 2, j = 1
i = 2, j = 2
i = 2, j = 3
i = 3, j = 1
i = 3, j = 2
i = 3, j = 3
```

จากตัวอย่างจะเห็นว่าแต่ละครั้งที่ Loop ด้านนอกทำงาน จะมีการทำงานของ Loop ด้านในจนครบเซ็ต 1 เซ็ต ซึ่งทั้ง 2 Loop เป็นการเพิ่มค่า i หรือ j ไปเรื่อย ๆ จาก 1 จนมีค่าเท่ากับ 3 ดังนั้นเมื่อนำทั้ง 2 ลูปมาซ้อนกันจึงเกิดการทำงานภายในลูปซ้อนนี้ 9 ครั้ง

## break

คือ คำสั่งที่ใช้เพื่อควบคุมให้จบการทำงานของ Loop หรือวนซ้ำในทันที โดยที่ไม่สนใจว่าเงื่อนไขจะยังคงเป็นจริงอยู่หรือไม่ เรามักใช้ร่วมกับคำสั่ง if เพื่อตรวจสอบเงื่อนไขก่อน และถ้าหากเงื่อนไขบางอย่างเป็นจริง เราจึงจะเรียกใช้คำสั่ง break เพื่อจบการทำงานของ Loop เช่น

```
for (int i = 1; i <= 10; i++) {
    if (i == 7) {
        break;
    }
    printf("%d ", i);
}
```

```
1 2 3 4 5 6
```

ในตัวอย่างจะเห็นว่า Loop หยุดทำงานเมื่อ i มีค่าเท่ากับ 7 ทำให้ไม่มีการนับเลขต่อ สังเกตว่าด้านหลังคำว่า break จะมีเครื่องหมาย Semicolon ด้วย (;)

## Continue

คำสั่ง continue เป็นคำสั่งควบคุมที่ใช้เพื่อข้ามการทำงานของ Loop ในรอบปัจจุบันไปยังรอบใหม่ในทันที โดยที่จะข้ามคำสั่งที่ปรากฏหลังจากคำสั่ง continue ทั้งหมด และไม่เหมือนกับคำสั่ง break คำสั่ง continue จะยังคงให้ Loop ทำงานอยู่ถ้าหากเงื่อนไขยังเป็นจริง เพียงแต่จะข้ามการทำงานในรอบนี้ไป เช่น

```
for (int i = 1; i <= 10; i++) {
    if (i % 3 == 0) {
        continue;
    }
    printf("%d ", i);
}
```

```
1 2 4 5 7 8 10
```

ในตัวอย่าง เป็นการนับ 1 ถึง 10 แต่จะข้ามเลขที่ 3 หารลงตัวไป สังเกตว่าด้านหลังคำว่า continue จะมีเครื่องหมาย Semicolon ด้วย (;) เหมือนกับคำสั่ง break

## Array

คือ ตัวแปรที่เก็บข้อมูลเป็นชุดข้อมูลหลาย ๆ ตัว โดยข้อมูลนั้นจะต้องเป็นประเภทเดียวกัน รูปแบบของการประกาศ Array ดังนี้

```
Type VariableName[Size];
```

จะเห็นว่าการประกาศ Array นั้น มีรูปแบบเหมือนกันกับการประกาศตัวแปรธรรมดา เพียงแต่จะมีการใส่วงเล็บ ([ ]) และภายในวงเล็บจะมีค่า size เพื่อบ่งบอกขนาดของ Array หรือก็คือค่าที่บอกว่าชุดข้อมูลนี้จะเก็บข้อมูลได้สูงสุดกี่ตัว สำหรับการตั้งชื่อก็มีกฎแบบเดียวกันกับการประกาศตัวแปรแบบธรรมดา

ตัวอย่างการประกาศ Array

```
int Set[5];
char Name[20];
float Arr[10];
```

ในตัวอย่างเราได้ประกาศ Array โดยที่ยังไม่ได้มีการกำหนดค่า จากตัวอย่างตัวแปร Set จะเก็บข้อมูลประเภท int ได้สูงสุด 5 ตัว ตัวแปร Name จะเก็บข้อมูลประเภท char ได้ 20 ตัว และตัวแปร Arr จะเก็บข้อมูลประเภท float 10 ตัว

และเหมือนกันกับตัวแปรปกติ ตัวแปรแบบ Array สามารถประกาศพร้อมกำหนดค่าได้ ซึ่งหากประกาศตัวแปร Array พร้อมกำหนดค่า จะไม่ต้องใส่ขนาดของ Array ปล่อยให้ด้านในวงเล็บว่างเพราะคอมไพเลอร์จะกำหนดให้เอง โดยจะกำหนดให้ขนาดพอดีกับจำนวนข้อมูลที่ใส่ตอนประกาศพร้อมกำหนดค่า หรือประกาศผสมกับแบบปกติก็ได้ โดยคั่นแต่ละตัวด้วยเครื่องหมายจุลภาค (,) เช่นกัน

ตัวอย่างการประกาศตัวแปรในรูปแบบต่าง ๆ

```
int Num = 5, Rating, Set[5];
char Signal = 'E', Name[20], Vowel[7] = {'a', 'E', 'i', 'o', 'u'};
float Arr[] = {1.0, 2, 3.2, 4.1, 5.8};
```

ในตัวอย่างได้ประกาศและกำหนดค่าให้กับ Array ในหลาย ๆ แบบ สังเกตว่าตัวแปร Arr ที่เป็น Array ของ float ไม่ต้องใส่ขนาดก็ได้ ซึ่งในกรณีนี้คอมไพเลอร์จะกำหนดให้ Arr มีขนาดเท่ากับ 5 เอง และจากตัวอย่างเราจะเห็นได้อีกว่าเมื่อเราประกาศตัวแปรแบบ Array พร้อมกำหนดค่า มีรูปแบบคือ มีการใช้เครื่องหมายวงเล็บปีกกา ({ }) และด้านในวงเล็บ จะมีการใส่ลำดับของค่าข้อมูลที่จะเก็บในตัวแปร Array นี้

## การเข้าถึงค่าใน Array

ใน Array เราจะใช้ index ในการบอกตำแหน่ง โดยตำแหน่งของข้อมูลในอาเรย์จะเริ่มจาก 0 เสมอ นั่นคือ index ของอาเรย์จะมีค่าอยู่ระหว่าง 0 ถึง size-1 ข้อมูลใน Array เราจะใช้รูปแบบนี้

```
VariableName[Index];
```

ตัวอย่างการเข้าถึงข้อมูลใน Array เช่น

```
int Array[8] = {5, 10, 15, 20, 25, 30, 35, 40};

printf ("Array[0] = %d\n", Array[0]);
printf ("Array[5] = %d\n", Array[5]);

Array[5] = 99;
printf ("Array[5] = %d", Array[5]);
```

```
Array[0] = 5
Array[5] = 30
Array[5] = 99
```

ในตัวอย่างเราได้สร้างตัวแปร Array ที่มีขนาด 8 และกำหนดค่าเริ่มต้นให้ จะเห็นว่าค่าของ Array[0] คือ เลขตัวแรก และค่าของ Array[5] คือเลขตัวที่ 6 สำหรับการใช้งานจะเห็นได้ว่ามีการใช้งานเหมือนกันกับตัวแปรปกติเพียงแต่มีการระบุ index ของตัวแปรเพิ่มเติม

## การใช้คำสั่ง for loop กับ Array

เนื่องจาก Array เป็นลำดับของตัวแปรและใช้ index ในการเข้าถึงข้อมูล ดังนั้น Array จึงมักใช้คู่กับ Loop โดยเฉพาะกับ for loop เพื่อที่จะเข้าถึงค่าใน Array เช่น

```
int X[100];

for (int i = 0; i < 100; i++) {
   X[i] = (i + 1) * 7;
}
```

ในตัวอย่าง เราจะได้ Array ของตัวแปร int ชื่อ X มีชุดของข้อมูลด้านในเป็นพหุคูณของ 7 แบบนี้ X[100] = {7, 14, 21, 28, 35, 42, ... , 700}

## Array 2 มิติ

ที่ผ่านมา คือ Array 1 มิติ ซึ่งหากจินตนาการว่า Array 1 มิติ คือ List ของข้อมูล Array 2 มิติก็คือ ตารางของข้อมูลนั่นเอง ตัวอย่างการใช้งาน เช่น

```
int Table[3][3] =
   {
      {35, 6, 1},
      {4, 5, 16},
      {9, 7, 92}
```

```
    };

int i, j;
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        printf ("%d ", Table[i][j]);
    }
    printf ("\n");
}
```

```
35 6 1
4 5 16
9 7 92
```

ในตัวอย่างจะเป็นการประกาศตัวแปร Array ใน 2 มิติแบบกำหนดค่า และการเข้าถึงตำแหน่งต่าง ๆ ใน Array 2 มิติ ซึ่งจะเห็นว่าการจะใช้ Array 2 มิติ จะต้องมีความรู้และความชำนาญในการใช้ Loop ซ้อน Loop ก่อน

นอกจากนี้ เรายังสามารถสร้าง Array ที่มีมากกว่า 2 มิติ ได้อีก คือ Array 3 มิติ Array 4 มิติ หรือมากว่านั้น เช่น

```
float Cube[1][5][2];
int Map[4][3][2][1];
```

## String

คือ ข้อมูลประเภทข้อความที่ประกอบจากหลายตัวอักษร ซึ่งจริง ๆ แล้ว string ก็คือ Array ของ character เพราะอย่างที่รู้ว่าข้อมูลประเภท char ใช้สำหรับเก็บข้อมูลหนึ่งตัวอักษร และ Array จะใช้สำหรับเก็บชุดของข้อมูลหลายตัว เมื่อนำมารวมกันเราจึงได้เป็นชุดของข้อมูลตัวอักษร หรือก็คือข้อความนั่นเอง

ตัวอย่างการประกาศ string

```
char Name1[] = {'C', 'P', 'E', '\0'};
char Name2[] = {"CPE"};
```

ในการประกาศ string เราสามารถทำได้ทั้ง 2 แบบ แบบแรกคือการประกาศ Array ของ char ซึ่งก็คือ string โดยที่ตัวสุดท้ายเราจะใส่ \0 เพิ่มเข้าไปด้วยเสมอหากประกาศแบบนี้ แต่สำหรับแบบที่สองเราจะใช้เครื่องหมาย double quotes (") ครอบข้อความ และถึงแม้จะไม่ใส่ \0 เราก็จะได้ผลลัพธ์เดียวกับแบบแรก เพราะโปรแกรมจะเติมให้อัตโนมัติ

## ตัวอักษร '\0'

คือ ตัวอักษรพิเศษที่เรียกว่า null-terminated character ซึ่งใช้สำหรับระบุจุดสิ้นสุดของ String ดังนั้นเราจำเป็นต้องใส่ตัวอักษรนี้เสมอเมื่อเราประกาศ string

สิ่งที่เกิดขึ้นในการทำงาน คือ เมื่อคอมไพเลอร์พบกับตัวอักษร \0 ใน string มันจะพิจารณาว่าตัวอักษรก่อนหน้าทั้งหมดเป็นค่าของ String และเพิกเฉยต่อตัวอักษรที่เหลือทั้งหมด หมายความว่าหากเราไม่ใส่ตัวอักษรนี้เพื่อระบุจุดสิ้นสุดของ string คอมไพเลอร์ก็จะไม่รู้ว่าตรงไหนคือจุดจบของ string และจะทำให้ String ของเรามีตัวอักษรแปลก ๆ ติดมาด้วย

## การรับค่าและแสดงผลของ string

เราจะใช้ %s ตัวอย่างการใช้งาน เช่น

```
char Name[10];
printf ("What's your name\n");
scanf ("%s", Name);
printf ("You are %s", Name);
printf ("The first letter of your name is %c", Name[0]);
```

Input: Sugar

```
What's your name
Sugar
You are Sugar
The first letter of your name is S
```

จากตัวอย่าง จะเห็นว่าเมื่อเราต้องการแสดงข้อความหรือค่าข้อมูลทั้ง string เราจะไม่ใส่วงเล็บ {[ ]} แต่ถ้าหากต้องการแสดงหรือเข้าถึงข้อมูลตัวเดียวที่อยู่ข้างใน เราจะใส่วงเล็บ {[ ]} นอกจากนี้ตรงฟังก์ชัน scanf() หากใช้กับ string เราจะไม่มีการใส่สัญลักษณ์ ampersand (&) นั่นเพราะสำหรับตัวแปรแบบ string การพิมพ์ชื่อเฉย ๆ ก็ถือเป็นการบอกตำแหน่งที่ข้อมูลถูกเก็บภายในหน่วยความจำแล้ว

และข้อควรระวังสำหรับ string คือ เราไม่สามารถกำหนดค่าหรือเปลี่ยนค่าของ string ภายหลังจากที่ประกาศแล้ว เช่น

```
char Name1[5];
Name1 = {"CPE"};
```

แบบนี้ทำไม่ได้ หากจะเปลี่ยนหรือใส่ค่าจริง ๆ จะต้องทำทีละตัว ดังนี้

```
char Name1[5];
Name1[0] = 'C';
Name1[1] = 'P';
Name1[2] = 'E';
Name1[3] = '\0';
```

จะเห็นว่าการจัดการกับ string นั้นยากมาก แต่อย่างไรก็ตามในภาษา C มี string.h Library ที่จะเพิ่มความสะดวกอย่างมากในการจัดการกับข้อมูลแบบ string ดังนี้

## string.h Library

ฟังก์ชันสำคัญใน string Library มีดังนี้

- ฟังก์ชันช่วยหาความยาวของ **strlen()**
- ฟังก์ชันช่วยคัดลอก **strcpy()**
- ฟังก์ชันช่วยเชื่อมต่อ **strcat()**
- ฟังก์ชันช่วยเปรียบเทียบ **strcmp()**

## การหาความยาวของ string

ใช้ strlen() เพื่อหาความยาวของ string รูปแบบการใช้งาน คือ

```
strlen (string);
```

การทำงานของฟังก์ชัน คือ ส่งค่าความยาวของ string ออกมาเป็นเลขจำนวนเต็ม

ตัวอย่างการใช้งาน เช่น

```
char Text[] = {"C Programing"};
int Length = strlen (Text);
printf ("%d", Length);
```

```
12
```

## การคัดลอก string ไปยังตัวแปรใหม่

เราใช้ strcpy() (string copy) โดยรูปแบบการใช้งาน คือ

```
strcpy (PasteString, CopyString);
```

การทำงานของฟังก์ชัน คือ ทำการคัดลอกข้อความของตัวแปรทางขวาไปยังตัวแปรทางซ้าย หากจำไม่ได้หรือกลัวจำสลับ ให้นึกถึงเครื่องหมายเท่ากับ ( = ) ที่นำผลลัพธ์ทางขวาไปใส่ตังแปรทางซ้าย

ตัวอย่างการใช้งาน เช่น

```
char Context[] = {"Kitty Cat"};
char AnotherContext[10];

strcpy (AnotherContext, Context);
printf ("%s", AnotherContext);
```

```
Kitty Cat
```

## การเชื่อมต่อ string

เราจะใช้ strcat() เพื่อเชื่อมต่อ string สองตัวเข้าด้วยกัน รูปแบบการใช้งานเป็น ดังนี้

```
strcat (String1, String2);
```

การทำงานของฟังก์ชัน คือ นำ string2 ไปต่อท้าย string1 ซึ่งหลังจากการทำงาน string1 จะเปลี่ยนไป แต่ string2 จะเหมือนเดิม

ตัวอย่างการใช้งาน เช่น

```
char Sitename[] = {"comcamp"};
char TLD[] = {".io"};

strcat (Sitename, TLD);
printf ("%s", Sitename);
```

```
comcamp.io
```

## การเปรียบเทียบ String

ในการเปรียบเทียบความเท่ากันของ string สองตัวเราไม่สามารถใช้ตัวดำเนินการเปรียบเทียบ (==) ได้ ทำให้เราจะต้องใช้ฟังก์ชัน strcmp() แทน ซึ่งมีรูปแบบการใช้งาน ดังนี้

```
strcmp (String1, String2);
```

การทำงานของฟังก์ชัน คือ การนำ string ทั้งสองตัวมาเทียบกัน และส่งเลขจำนวนเต็มออกมา ซึ่งแต่ละเลขก็จะมีความหมายต่างกัน ดังนี้

- มีค่า 1 เมื่อ**ค่าของอักษรตัวแรกที่ต่างกัน**ของ **string1** มีค่า**มากกว่า string2** ตาม ASCII TABLE
- มีค่า -0 เมื่อ string1 เหมือนกันกับ string2
- มีค่า -1 เมื่อ**ค่าของอักษรตัวแรกที่ต่างกัน**ของ **string1** มีค่า**น้อยกว่า string2** ตาม ASCII TABLE

ตัวอย่างการใช้งาน เช่น

```
char Name1[] = {"Martin"};
char Name2[] = {"Mark"};

if (strcmp (Name1, Name2) == 0) {
   printf("Equal");
} else {
   printf ("Not equal");
}
```

```
Not equal
```

## Function

คือ ส่วนของโปรแกรมที่ถูกเขียนขึ้นเพื่อทำหน้าที่เฉพาะของมัน โดยฟังก์ชันจะถูกแบ่งเป็น 2 แบบคือ แบบแรกเป็นฟังก์ชันที่มีอยู่แล้วเราสามารถนำมาใช้ได้ และแบบที่สองที่เราจะสร้างขึ้นมาเอง ตัวอย่างฟังก์ชันที่มีอยู่แล้ว เช่น printf(), scanf(), strlen(), strcmp() เป็นต้น ซึ่งเราจะมาดูวิธีสร้างฟังก์ชันเองกัน

## การประกาศฟังก์ชัน

รูปแบบการประกาศฟังก์ชันเป็นดังนี้

```
Type Name (Paramiter1, Paramiter1, ...) {
   // statements
}
```

**Type** คือ เป็นประเภทของฟังก์ชันที่ต้องการประกาศ ขึ้นอยู่กับค่าที่ฟังก์ชันจะส่งออกมา ซึ่งประเภทของฟังก์ชันก็จะเหมือนกันกับประเภทของตัวแปร เช่น int, float, char และสำหรับฟังก์ชันที่ไม่มีการส่งค่าออกมาจะมี Type เป็น void

**Name** เป็นชื่อของฟังก์ชัน ซึ่ง มีกฎการตั้งชื่อเหมือนกันกับการตั้งชื่อตัวแปร เมื่อต้องการใช้งานฟังก์ชันก็ใช้ชื่อของฟังก์ชันเป็นตัวแทน

**Parameters** เป็นตัวแปรที่จะส่งเข้ามาในฟังก์ชัน โดยพารามิเตอร์สามารถมีหรือไม่ก็ได้

**statement** เป็นคำสั่งของโปรแกรมเพื่อที่จะให้ฟังก์ชันทำงานและได้ผลลัพธ์ที่ต้องการ ซึ่งฟังก์ชันที่เขียนขึ้นเอง จะต้องมาก่อนฟังก์ชัน main เสมอ

## คำศัพท์

ฟังก์ชันพารามิเตอร์ - เป็นค่าที่ส่งเข้ามาในฟังก์ชัน โดยจะเป็นลำดับของตัวแปรที่ส่งเข้ามา

ฟังก์ชันอาร์กิวเมนต์ - เป็นค่าของตัวแปรที่จะส่งเข้าไปในฟังก์ชัน ซึ่งจะต้องสอดคล้องกับฟังก์ชันพารามิเตอร์

ค่า return ของฟังก์ชัน - ในการสร้างฟังก์ชันนั้น ส่วนมากจำเป็นต้องมีการส่งค่าออกมา โดยการใช้คำสั่ง return โดยในการประกาศฟังก์ชัน ประเภทของฟังก์ชันจะบ่งบอกถึงประเภทของค่าที่ส่งออกมา

ตัวอย่างเรื่องฟังก์ชัน เช่น

```
#include <stdio.h>

int Factorial (int Number) {
   int Result = 1;

   for (int i = 1; i <= Number; i++) {
      Result = Result * i;
   }

   return Result
}

int main () {
   int Num1 = Factorial(4);
   int Num2 = Factorial(3);

   printf ("%d", Num1 – Num2);
   return 0;
}
```

```
18
```

ในตัวอย่างจะเป็นฟังก์ชันในการหาค่า Factorial ของตัวเลขที่เราสร้างขึ้นเอง ในภาษา C เราจะใช้งานฟังก์ชันด้วยชื่อของมัน สังเกตในตัวอย่าง เราได้สร้างฟังก์ชันชื่อ Factorial ตอนใช้งานก็ใช้คำว่า Factorial() ด้านในวงเล็บ ( ) ก็จะใส่ค่าที่ต้องการส่งเข้าไป และเมื่อฟังก์ชันทำงานเสร็จก็จะส่งค่าบางอย่างออกมา ในกรณีตัวอย่างก็จะมีการส่งค่าของ Result ออกมาเป็นผลลัพธ์ก่อนที่จะนำผลลัพธ์นี้ไปเขียนทับตัวแปร Num1

โดยในตัวอย่างฟังก์ชันพารามิเตอร์ก็คือ Number ส่วนฟังก์ชันอาร์กิวเมนต์ก็คือ 4 กับ 3 นั่นเอง