บทที่ 1
เริ่มต้นเขียนโปรแกรมต้องรู้จักอะไรบ้าง

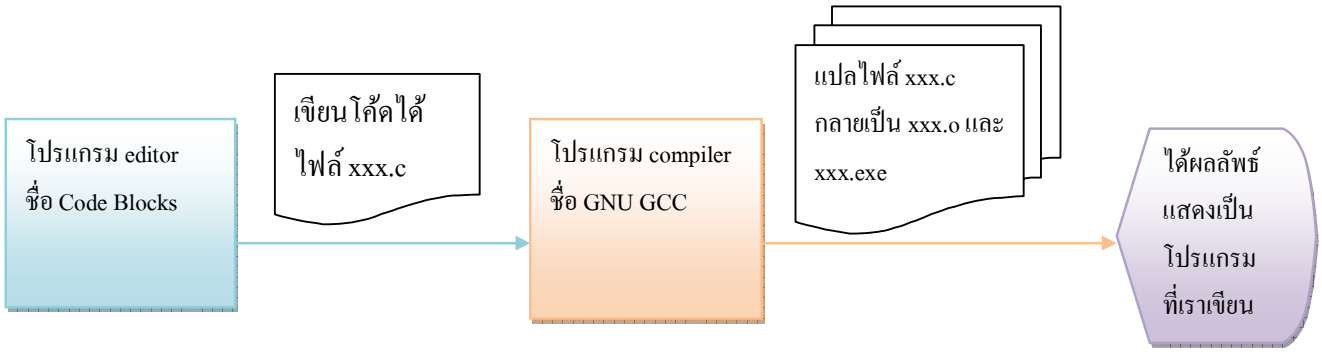
1. จะเขียนภาษาอะไรล่ะ

การที่จะเขียนโปรแกรมภาษานึงๆ จะต้องมีเครื่องมือคือ

- ตัวแปรภาษา (compiler / intepretor)
- โปรแกรมสำหรับเขียนโค้ด (editor)

เราก็ต้องไปทำการดาวโหลดโปรแกรม 2 อย่างนี้มาติดตั้งในเครื่องคอมพิวเตอร์ก่อน ถึงจะเขียนโค้ดด้วยภาษาซีได้ ตัวอย่างดังตาราง

ชื่อภาษา	ตัวแปรภาษา	ตัวอย่างโปรแกรม editor
ภาษาซี	GNU GCC Borland Turbo C MinGW	Turbo C Borland C++ Visual C++ Visual Studio.net Code Blocks Notepad
ภาษาซีพลัสพลัส	GNU G++ Microsoft Visual C++	Turbo C++ Borland C++ Visual C++ Visual Studio.net Code Blocks Notepad
ภาษาจาวา	Java Development kit (Oracle) GNU GCJ Eclipse Compiler for Java ECJ	JCreator Netbeans Eclipse Notepad



2. รู้จักวิธีการเขียนโปรแกรมก่อน

1. เฉพาะ ฟังก์ชันหลัก(main function) เท่านั้นที่ถูกทำงานเป็นอันดับแรกเสมอในโปรแกรม
2. ลักษณะการทำงาน โปรแกรมเริ่มทำงานตั้งแต่ชุดคำสั่งบรรทัดแรกในฟังก์ชันหลักและเรียงลำดับตามชุดคำสั่งไปจนจบ
3. เมื่อเจอชื่อฟังก์ชันใดๆในฟังก์ชันหลัก คอมไพเลอร์ก็จะกระโดดไปทำงานที่ฟังก์ชันชื่อนั้นๆ และเมื่อทำงานเสร็จ ก็จะกลับมายังตำแหน่งก่อนหน้าที่จะไป
4. ไฟล์โค้ดของโปรแกรม นามสกุล .c
5. เมื่อทำการแปลชุดคำสั่งแล้วจะได้ ไฟล์นามสกุล .o
6. เมื่อทำการรันโปรแกรมแล้วจะได้ไฟล์นามสกุล .exe ซึ่งไฟล์นามสกุล .exe คือผลลัพธ์ของโปรแกรมสำหรับนำไปใช้งานต่อไป

ตัวอย่าง จะเห็นว่าบรรทัดที่ 13-20 ทำงานทีละบรรทัด output

```
 8
 9  #include <stdio.h>
10
11
12
13  int main()
14  {
15      printf("Hello World\n");
16      printf("1\n");
17      printf("-5\n");
18      printf("2\n");
19      return 0;
20  }
21
```

เรียกว่าฟังก์ชันหลัก หรือ main มีขอบเขตตั้งแต่บรรทัดที่ 13-20

```
Hello World
1
-5
2
```

กรณีไม่มี ฟังก์ชัน main ก็จะไม่มีผลลัพธ์อะไรออกมา (มี error)

```
 8
 9  #include <stdio.h>
10
11
12  void print(){
13  printf("Hello World");
14  }
15
```

Compilation failed due to following error(s).

```
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info): relocation 0 has invalid symbol index 11
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info): relocation 1 has invalid symbol index 12
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info): relocation 2 has invalid symbol index 2
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info): relocation 3 has invalid symbol index 2
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info): relocation 4 has invalid symbol index 11
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info): relocation 5 has invalid symbol index 13
/usr/bin/ld: /usr/lib/debug/usr/lib/x86_64-linux-gnu/crt1.o(.debug_info): relocation 6 has invalid symbol index 13
```

ตัวอย่างกรณีมีฟังก์ชัน จะทำที่บรรทัด 15 - 18 เสร็จแล้ว

ที่บรรทัดที่ 19 ก็กระโดดมาทำบรรทัดที่ 11-13

แล้วกลับไปที่บรรทัด 19 ต่อให้เสร็จ แล้วก็ทำบรรทัดที่20-22

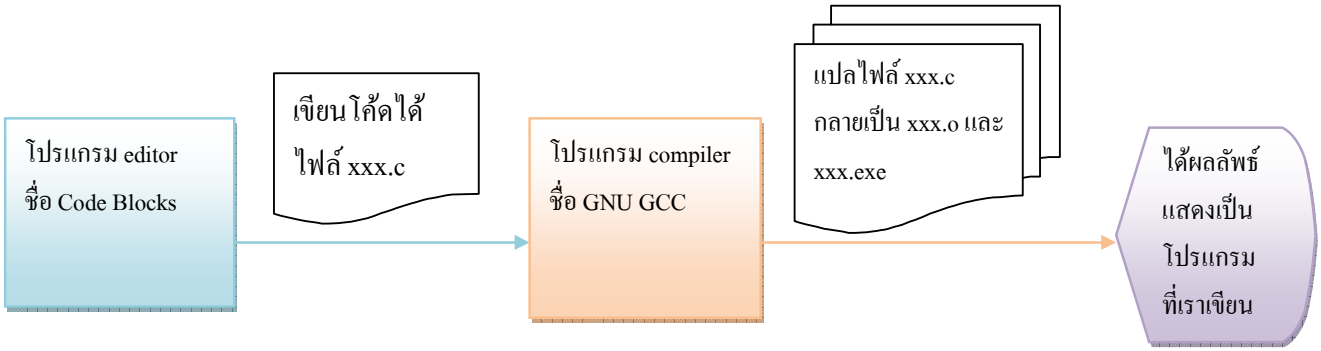
```
 9  #include <stdio.h>
10
11  void print(){
12  printf("-5\n");
13  }
14
15  int main()
16  {
17      printf("Hello World\n");
18      printf("1\n");
19      print();
20      printf("2\n");
21      return 0;
22  }
```

เรียกว่าฟังก์ชัน ชื่อ print มีขอบเขตตั้งแต่บรรทัด 11-13

output

```
Hello World
1
-5
2
```

ลำดับกระบวนการตั้งแต่เขียนโค้ดไปจนถึงรันผลลัพธ์



บทที่ 2
เริ่มต้นเรียนรู้ภาษาซี

1. สิ่งที่ห้ามลืม

การเขียนโปรแกรมในภาษาซี เมื่อจบหนึ่งชุดคำสั่งจะต้องลงท้ายด้วยเครื่องหมาย ; เสมอ

โค้ดโปรแกรมจะถูกคอมไพเลอร์แปลคำสั่งเพื่อไปทำงานเสมอ แต่โค้ดส่วนที่อยู่ภายในเครื่องหมาย คอมเมนท์(comment)จะไม่ถูกนำไปคอมไพล์ ดังนั้นนักเขียนโปรแกรมนิยมเขียนคอมเมนท์ไว้เพื่อเตือนความจำ ฯลฯ ซึ่งวิธีการเขียนคอมเมนท์มี 2 รูปแบบ ดังนี้

1.2.1 ใช้เครื่องหมาย // ตามด้วยข้อความ หนึ่งบรรทัด รูปแบบนี้สามารถคอมเมนท์ได้เพียงบรรทัดเดียว

1.2.2 ใช้เครื่องหมาย /* ตามด้วยข้อความที่ต้องการคอมเมนท์ แล้วตามด้วยเครื่องหมาย */ รูปแบบนี้สามารถคอมเมนท์ได้หลายบรรทัด ซึ่งข้อความที่อยู่ภายในเครื่องหมาย /* */ จะไม่ถูกแปล

การเขียนโปรแกรมในภาษาซีพลัสพลัส จะต้องมีการบอกขอบเขตของชุดคำสั่งเสมอ ซึ่งขอบเขตของแต่ละชุดคำสั่ง จะอยู่ในเครื่องหมาย { }

2. รู้จักคำสั่ง แสดงผลทางหน้าจอ

สามารถทำได้โดยใช้ฟังก์ชัน ที่ชื่อว่า printf (ถ้าใช้คำสั่งนี้เมื่อไหร่ ต้องมี include<stdio.h>)

ฟังก์ชันนี้รับอินพุต คือ ข้อความ ที่อยู่ในเครื่องหมาย " "

ดังนั้น คิดง่ายๆว่า อยากจะพิมพ์ข้อความอะไรออกทางหน้าจอ ใครมาก่อน มาหลัง ก็ต้องเรียงลำดับตามนั้น ใส่ไว้ในเครื่องหมาย " "

ส่วน "\n" เป็นอักขระพิเศษ สั่งให้ขึ้นบรรทัดใหม่

ดังนั้น ถ้าแทรก "\n" ไว้ตรงไหน ก็จะขึ้นบรรทัดใหม่ตรงนั้น

```
 9  #include <stdio.h>
10
11
12
13  int main()
14  {
15      printf("Hello World\n");
16      printf("1\n");
17      printf("-5\n");
18      printf("2\n");
19      return 0;
20  }
21
```

```
Hello World
1
-5
2
```

3. ชนิดข้อมูลพื้นฐาน

ภาษาซีพลัสพลัสมีชนิดข้อมูลพื้นฐานหลายชนิดดังนี้

ชื่อเรียก	คุณสมบัติ	ขนาด	ตัวอย่าง
char	ตัวอักษรหนึ่งตัว หรือ เลขจำนวนเต็มขนาดน้อยๆ	1byte	'a' 'B' '9' จะถูกเขียนอยู่ในเครื่องหมาย ' '
int	เลขจำนวนเต็ม	4bytes	-50 2 390
_Bool	ค่าตรรกศาสตร์เป็นได้แค่สองอย่าง คือ true หรือfalse	1byte	1 (คือ true) 0 (คือ false)
float	เลขทศนิยม	4bytes	3.45 -4.1 1.0
double	เลขทศนิยมขนาดมากกว่า float	8bytes	3567.4512 -4.1556432 5642341.0

4. เมื่อไหร่จึงจะมีการประกาศตัวแปรเพื่อใช้เก็บค่าต่างๆ

เมื่อเรา ต้องมีการจำค่า

ยกตัวอย่างเช่น

- รับค่าจากผู้ใช้ (รับจากคีย์บอร์ด รับจากไฟล์ ฯลฯ)
- ทำการคำนวณต่างๆ แล้วต้องเก็บค่าไว้เพื่อใช้ในการคำนวณครั้งต่อๆไป

4.1 วิธีการประกาศตัวแปร

```
        ชนิดข้อมูล  ชื่อตัวแปร
เช่น      int              x    ;
```

ถ้าตัวแปรหลายๆตัวมีชนิดข้อมูลเดียวกันสามารถประกาศในชุดคำสั่งเดียวกันได้ โดยใช้เครื่องหมาย , คั่นระหว่างแต่ละตัวแปร เช่น int x , y ;

นอกจากจะประกาศตัวแปรแล้วยังกำหนดค่าเริ่มต้นให้กับตัวแปรด้วยก็ได้ ด้วยการเขียนเครื่องหมาย = แล้วตามด้วยค่าที่กำหนดให้ เช่น int x = 10; หรือ char y = ‘A’ , z = ‘B’ ;

4.2 การตั้งชื่อตัวแปร

ต้องขึ้นต้นด้วยตัวอักษร หรือเครื่องหมาย _ เท่านั้น แล้วตามด้วย ตัวอักษร หรือ ตัวเลข หรือเครื่องหมาย _ โดยไม่สามารถเว้นช่องว่างระหว่างตัวแปรหนึ่งชื่อได้ และไม่สามารถมีเครื่องหมาย _ มากกว่า 1 ตัวได้ (ทั้งนี้ขึ้นอยู่กับแต่ละคอมไพเลอร์)

4.3 รู้จักคำสั่ง แสดงผลทางหน้าจอ (ต่อ)

การใช้ฟังก์ชัน ที่ชื่อว่า printf ซึ่งปกติ ฟังก์ชันนี้รับอินพุต คือ ข้อความ ที่อยู่ในเครื่องหมาย " "

ทีนี้ ถ้าเราอยากจะพิมพ์ ตัวเลข(ชนิด int / float / double) ตัวอักษรอื่น (char) หรือ ค่า จริงเท็จ (_Bool) ลงไปในประโยคเดียวกัน พร้อมกันกับข้อความที่อยู่ใน เครื่องหมาย " " จะทำอย่างไร

ชนิดข้อมูลในข้อ 3 สามารถใช้คู่กับอักขระพิเศษได้ดังนี้

ชนิดข้อมูล	อักขระพิเศษ ที่ใช้กับ printf
int	%d
float	%f
double	%f
char	%c
_Bool	%d
String (คือ char ติดกันหลายๆตัว)	%s

ดังนั้น เรามาวิเคราะห์ประโยค output ที่ต้องการพิมพ์กันก่อน

ตัวอย่างเช่น

```
My name is Jiraporn.
GPA = 3.95
```

เราจะเริ่มเขียนโค้ดทีละแบบ (กรุณาทำความเข้าใจทีละขั้นตอน)

แบบแรก สมมติว่าเรามองว่า ทั้งหมดที่ต้องการจะพิมพ์คือข้อความ ก็แค่ให้ทุกอย่างอยู่ใน " "

จะเห็นว่า หลัง My name is Jiraporn. จะต้องขึ้นบรรทัดใหม่ ก็ต้องมีการแทรก \n ด้านหลัง Jiraporn.

ก็จะเขียนโค้ดได้ว่า

```
 9  #include <stdio.h>
10
11
12  int main()
13  {
14      printf("My name is Jiraporn.\n");
15      printf("GPA = 3.95");
16      return 0;
17  }
```

แบบที่ 2 ถ้ามองตัวหนังสือเป็นข้อความ ส่วน 3.95 ก็เป็นตัวเลข
อะไรที่เป็นข้อความก็อยู่ใน " " ส่วนตรงตำแหน่งของตัวเลข ก็แทนด้วยอักขระพิเศษ (%d หรือ %f)
ในที่นี้ ตัวเลขเป็นเลขทศนิยม ซึ่งต้องแทนด้วย %f
แล้ว เราก็ต้องระบุว่า %f ตัวนั้น จะต้องแทนค่าด้วยเลขอะไร ซึ่งในที่นี้คือ 3.95 เราก็ใส่มันด้านหลัง ,
เราก็สามารถเขียนโค้ดได้ว่า

```
 8
 9  #include <stdio.h>
10
11
12  int main()
13  {
14      printf("My name is Jiraporn.\n");
15      printf("GPA = %f",3.95);
16      return 0;
17  }
18
```

%f จะถูกแทนที่ด้วยตัวเลข 3.95 เวลาแสดงผลลัพธ์

โค้ดแบบที่สองนี้ สามารถเขียนรวมกันใน ฟังก์ชัน printf ครั้งเดียวก็ได้เช่นกัน ดังนี้

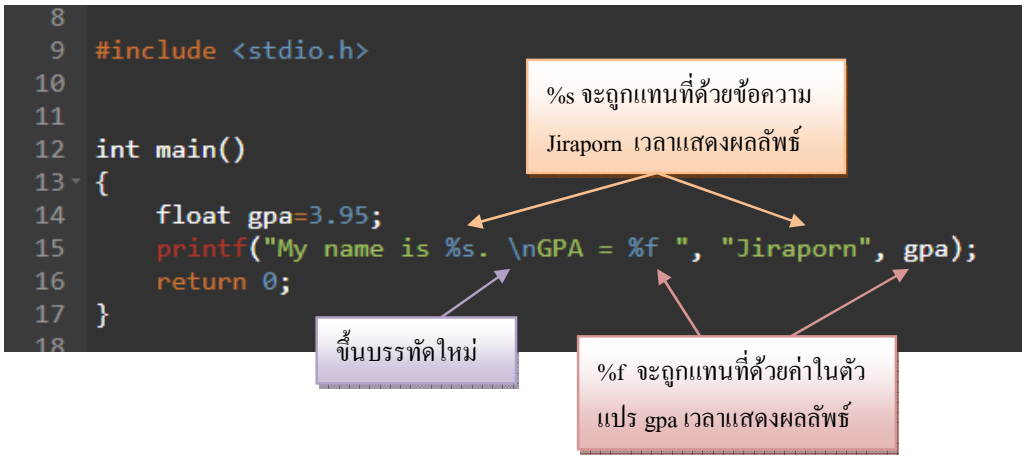
```
 8
 9  #include <stdio.h>
10
11
12  int main()
13  {
14      printf("My name is %s. \nGPA = %f ", "Jiraporn", 3.95);
15      return 0;
16  }
```

%s จะถูกแทนที่ด้วยข้อความ Jiraporn เวลาแสดงผลลัพธ์

ขึ้นบรรทัดใหม่

%f จะถูกแทนที่ด้วยตัวเลข 3.95 เวลาแสดงผลลัพธ์

แบบที่ 3 ถ้ามองตัวหนังสือเป็นข้อความ ส่วน 3.95 ก็เป็นตัวเลขที่ถูกเก็บอยู่ในตัวแปร
อะไรที่เป็นข้อความก็อยู่ใน " " ส่วนตรงตำแหน่งของตัวเลข ก็แทนด้วยอักขระพิเศษ (%d หรือ %f)
ในที่นี้ ตัวเลขเป็นเลขทศนิยม ซึ่งต้องแทนด้วย %f
แล้ว เราก็ต้องระบุว่า %f ตัวนั้น จะต้องแทนค่าด้วยเลขอะไร ซึ่งในที่นี้คือ 3.95 แต่ว่า เลข 3.95 นั้นอยู่ในตัวแปรชื่อ gpa เราก็ใส่ชื่อตัวแปรนั้นด้านหลัง ,
เราก็สามารถเขียนโค้ดได้ว่า

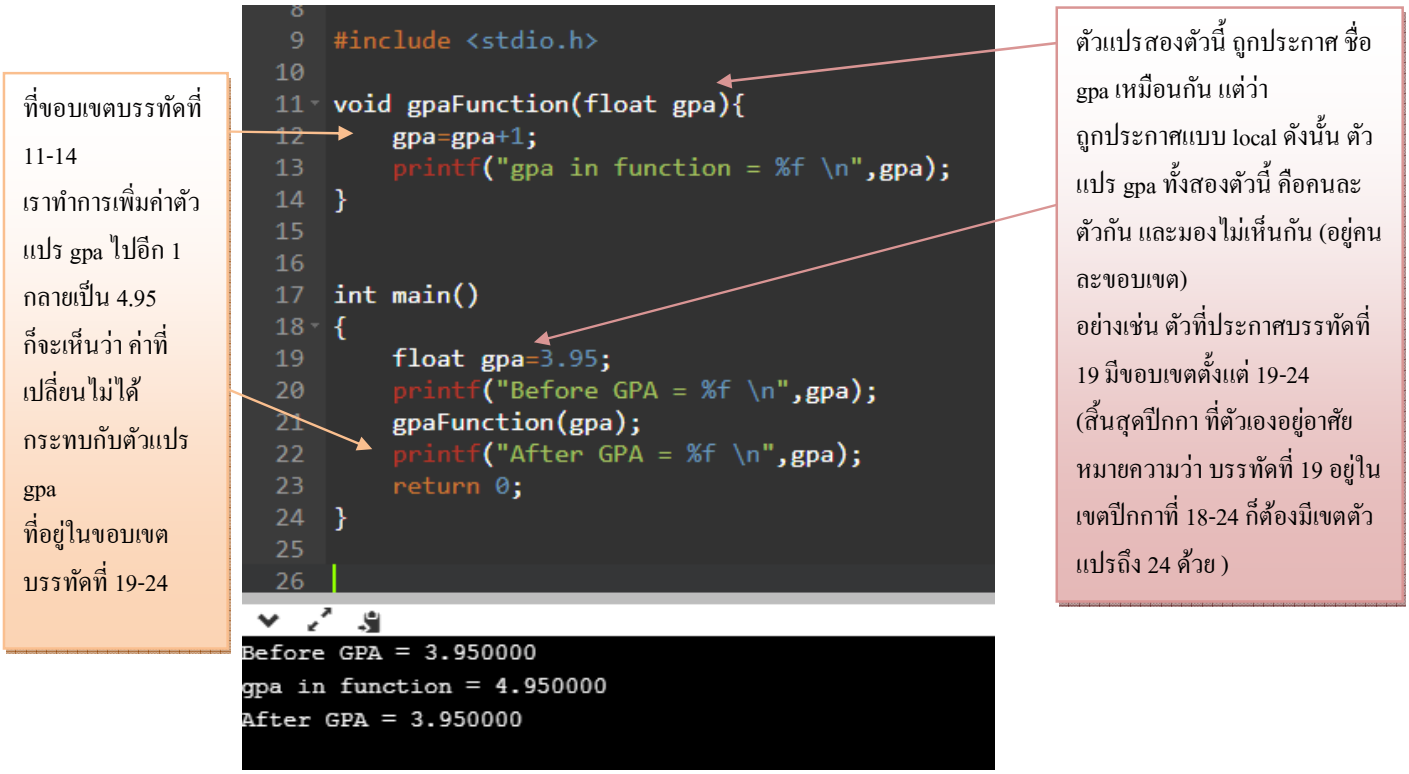


4.4 รูปแบบตัวแปร มี2 รูปแบบคือ

4.4.1 ตัวแปรแบบสาธารณะ (global variable) ตัวแปรชนิดนี้จะเป็นที่รู้จักไปทั่วทั้งไฟล์ โค้ดโปรแกรม วิธีการประกาศตัวแปรชนิดนี้จะต้องประกาศไว้ที่ต้นไฟล์โค้ดโปรแกรม ก่อนที่จะถูกเอามาเรียกใช้เท่านั้น

4.4.2 ตัวแปรแบบท้องถิ่น (local variable) ตัวแปรชนิดนี้จะเป็นที่รู้จักเฉพาะในขอบเขตของ { } ที่ตัวแปรถูกประกาศไว้เท่านั้น วิธีการประกาศตัวแปรชนิดนี้ ประกาศไว้ในฟังก์ชันที่ใช้งานเท่านั้น

ตัวอย่าง การประกาศตัวแปรแบบ local variable



ตัวอย่าง การประกาศตัวแปรแบบ local variable

```c
#include <stdio.h>

void gpaFunction(float gpa){
    int x =5;
    gpa=gpa+1;
    printf("gpa in function = %f \n",gpa);
    printf("x = %d \n",x);
}


int main()
{
    float gpa=3.95;
    printf("Before GPA = %f \n",gpa);
    gpaFunction(gpa);
    printf("After GPA = %f \n",gpa);
    printf("x = %d \n",x);
    return 0;
}
```

ตัวแปร x ประกาศที่บรรทัดที่ 12 ซึ่งอยู่ในขอบเขตปีกกา 11-16 ดังนั้น ขอบเขตของตัวแปร x คือ 12-16

ดังนั้น ที่บรรทัด 25 จึง error ว่าไม่รู้จักตัวแปร x เนื่องจาก ขอบเขตมาไม่ถึงบรรทัดที่ 25

input

Compilation failed due to following error(s).

```
main.c: In function 'main':
main.c:25:24: error: 'x' undeclared (first use in this function)
     printf("x = %d \n",x);
                        ^
```

ที่นี้สมมติว่า เราต้องการให้ตัวแปร gpa ที่เปลี่ยนค่าแล้วใน ฟังก์ชัน gpaFunction มีผลมาถึงตัวแปร gpa ที่อยู่ใน main ด้วย จะทำอย่างไร (พูดง่ายๆก็คือ เราต้องการให้ gpa เป็นตัวแปรตัวเดียวกัน)

ตัวอย่าง การประกาศตัวแปรแบบ global variable

```c
#include <stdio.h>

float gpa=3.95;
void gpaFunction(){
    gpa=gpa+1;
    printf("gpa in function = %f \n",gpa);
}

int main()
{
    printf("Before GPA = %f \n",gpa);
    gpaFunction();
    printf("After GPA = %f \n",gpa);
    return 0;
}
```

ประกาศ gpa ไว้บรรทัด ด้านนอกทุกๆ ฟังก์ชัน จะทำให้มีขอบเขต ตั้งแต่บรรทัดที่ประกาศจนถึงจบไฟล์ 11-23

```
Before GPA = 3.950000
gpa in function = 4.950000
After GPA = 4.950000
```

ยนเรศวร

5. ฟังก์ชัน

ในภาษาซีพลัสพลัส ฟังก์ชันที่ถูกทำงานคือ ฟังก์ชันหลัก ซึ่งจะต้องตั้งชื่อฟังก์ชันนี้ว่า main เท่านั้น เพราะฉะนั้นการสร้างฟังก์ชันอื่นๆมีข้อกำหนดดังนี้

โครงสร้างการเขียนฟังก์ชัน มีดังนี้

```
ชนิดข้อมูลของ output   ชื่อฟังก์ชัน  (  ชนิดข้อมูลของ input )
{


}
```

หมายเหตุ ชื่อของฟังก์ชันต้องติดกันเป็นคำๆเดียวเท่านั้น

ฟังก์ชันมี สองชนิดหลักๆคือ

1. ฟังก์ชันที่มีการคืนค่า
2. ฟังก์ชันที่ไม่มีการคืนค่า

ถ้าเขียนแบบที่ 1 จะได้

```
int  functionTest ( int x )
{
return  x;
}
```

จะพบว่าถ้าเขียนแบบนี้ มีการคืนค่า เพราะฉะนั้นจะต้องมีคำว่า return ซึ่งจะตามด้วยค่าที่ต้องการคืนให้เพื่อเป็นคำตอบของฟังก์ชันนั้นๆ

ถ้าเขียนแบบที่ 2 จะได้

```
void  functionTest ()
{
}
```

จะพบว่าถ้าเขียนแบบนี้ ไม่มีการคืนค่า เพราะฉะนั้นไม่ต้องมีคำว่า return และชนิดข้อมูลของ output จะต้องเป็นคำว่า void เท่านั้น

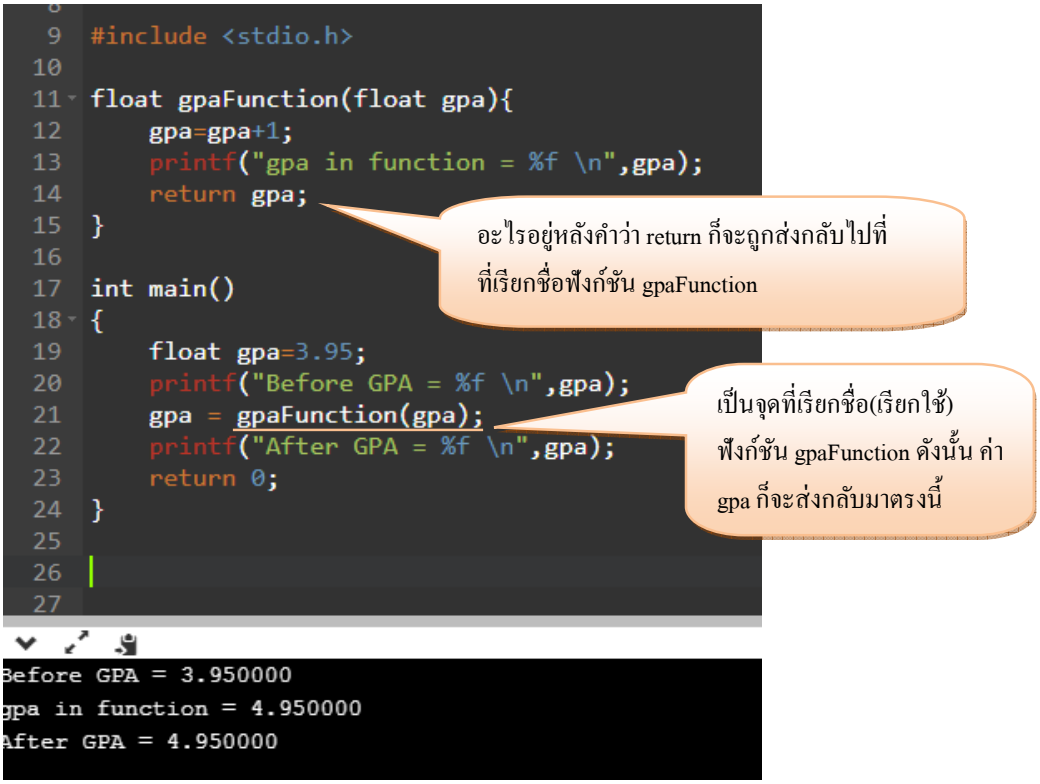
จากตัวอย่างที่แล้ว จะเห็นว่าเราประกาศฟังก์ชันแบบไม่มีการ return ค่า
ทำให้ ค่าตัวแปร gpa ที่อยู่ใน main ไม่ได้เปลี่ยนเป็น 4.95 ตามไปด้วย

```c
#include <stdio.h>

void gpaFunction(float gpa){
    gpa=gpa+1;
    printf("gpa in function = %f \n",gpa);
}


int main()
{
    float gpa=3.95;
    printf("Before GPA = %f \n",gpa);
    gpaFunction(gpa);
    printf("After GPA = %f \n",gpa);
    return 0;
}
```

```
Before GPA = 3.950000
gpa in function = 4.950000
After GPA = 3.950000
```

ถ้าหากว่าเราต้องการจะทำให้ค่าตัวแปร gpa ในฟังก์ชัน main เปลี่ยนไปตาม ค่าที่อยู่ใน gpaFunction จะทำยังไง

เราก็เปลี่ยนให้ฟังก์ชัน return ค่ากลับมา แล้วเราก็เอา output ที่ได้จากฟังก์ชัน gpaFunction มากำหนดให้กับตัวแปร gpa ที่อยู่ใน main ดังโค้ดต่อไปนี้



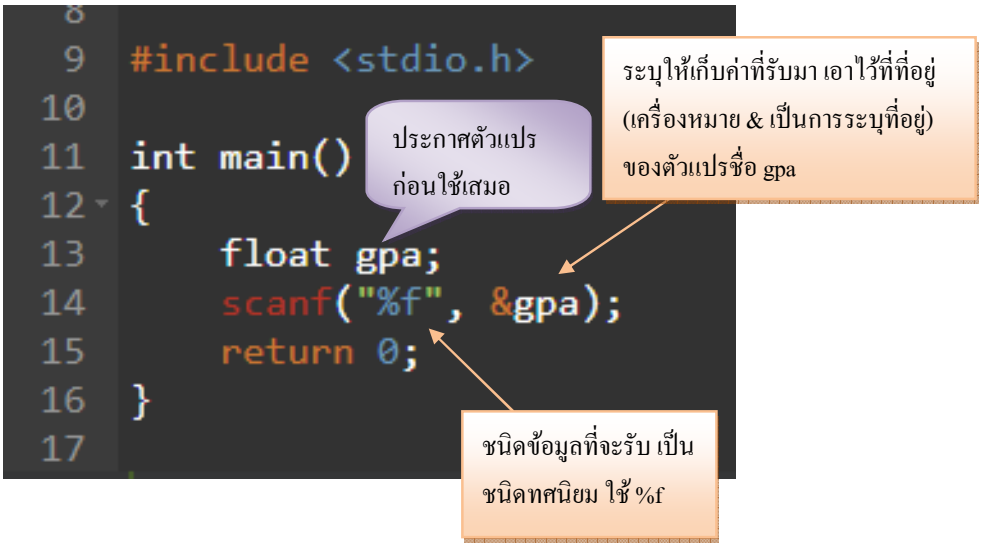
จะเห็นว่า ฟังก์ชัน gpaFunction มีการ return ค่า gpa (ตรงบรรทัด 14) กลับมาให้ ตรงที่บรรทัด 21 ทำให้ตัวแปร gpa ในmain ที่เดิม มีค่า 3.95 กลายเป็น 4.95 ไปด้วย (เพราะบรรทัดที่ 21 กำหนด ให้ gpa(ใน main) = ค่าที่ได้จากฟังก์ชัน gpaFunction

6. การรับค่าอินพุตจากคีย์บอร์ด

การใช้ฟังก์ชัน ที่ชื่อว่า scanf (ถ้าใช้ฟังก์ชันนี้ จะต้องมี include<stdio.h>) ซึ่งปกติ ฟังก์ชันนี้รับอินพุตสองค่า คือ ชนิดข้อมูลของค่าที่จะรับมาจากคีย์บอร์ด กับตำแหน่งที่อยู่ของตัวแปรที่จะใช้เก็บค่า (ดังนั้นจะต้องมีการประกาศตัวแปรไว้ก่อนแล้ว ถึงจะอ้างตำแหน่งที่อยู่ของตัวแปรนั้น เพื่อใช้เก็บค่าที่รับจากคีย์บอร์ดได้)

ทีนี้ ถ้าเราอยากจะรับข้อมูลตัวเลข(ชนิด int / float / double) ตัวอักษรอื่น (char) หรือ ค่า จริงเท็จ (_Bool) เราจะทำอย่างไร

สมมติ เราต้องการรับค่า ตัวเลขทศนิยมจากคีย์บอร์ด ก็เขียนโค้ดได้ว่า



จะเห็นว่า พอรันโปรแกรม เราก็สามารถกดคีย์บอร์ดได้ แต่ว่าเนื่องจากเราไม่ได้แสดงผลด้วยการพิมพ์อะไรออกมาเลย ทำให้ผู้ใช้ก็ไม่สามารถรู้ได้ว่า ตัวเลขนั้นเก็บเข้าไปแล้วจริงๆหรือไม่

เราก็นำฟังก์ชัน printf มาประยุกต์ใช้ คู่กับ scanf ดังนี้

วิเคราะห์ ผลลัพธ์ที่อยากได้ก่อน

```
Enter your GPA:
3.95 (รับค่าเข้ามาจากคีย์บอร์ด)
My name is Jiraporn.
GPA = 3.95
```

ก็ง่ายๆ ตรงไหนเป็นการแสดงผล(พิมพ์)ทางหน้าจอ เราก็ใช้ printf ตรงไหนเป็นการรอให้ผู้ใช้กดคีย์บอร์ด เราก็ใช้ scanf

ดังนั้นจะได้โค้ดดังนี้

แบบฝึกหัด

1. เขียนโปรแกรมเพื่อรับชื่อ-นามสกุล(ให้ใช้ตัวอักษรย่อ แทนชื่อ นามสกุล) และอายุของผู้ใช้ และแสดงผลออกทางหน้าจอ

ตัวอย่างผลรัน

```
Please enter your name: J P
Please enter your age: 25
Hello my name is J P
I am 25 years old.
```

2. เขียนโปรแกรมเพื่อคำนวนหาพื้นที่และเส้นรอบวงของวงกลม โดย ให้รับค่าจำนวนจริงของรัศมีจากแป้นพิมพ์ และแสดงผลออกทางหน้าจอ (ให้กำหนดค่า Pi เป็นค่าคงที่ มีค่าเท่ากับ 3.14)

ตัวอย่างผลรัน

```
Please input radius of circle: 3.2
Area =  32.1536
Circumference = 20.096
```

3. ให้นักศึกษาเขียนโปรแกรมเพื่อคำนวณหาดัชนีมวลกาย โดยให้รับค่า จำนวนเต็มของ น้ำหนัก(Kg) และส่วนสูง (cm) และแสดงผลการคำนวนออกทางหน้าจอ (BMI = weight(kg) / height(m)^2)

ตัวอย่างผลรัน

```
Please input weight and height : 68 180
BMI =  20.9877
```

จัดทำโดย อาจารย์จิราพร พุกสุข
ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

บทที่ 3

การเขียนโปรแกรมแบบมีเงื่อนไข

จากบทเรียนก่อนหน้า คำตอบของโปรแกรมที่ได้มักจะเป็นคำตอบที่มาจากสมการเดียวหรือมาจากรูปแบบๆเดียว ซึ่งในความเป็นจริงแล้ว คำตอบของโปรแกรมที่ได้ อาจจะไม่ได้มาจากสมการเดียวกันหรือมาจากรูปแบบเดียวกันก็ได้ เพราะฉะนั้นเมื่อคำตอบของโปรแกรมมาจากสมการหรือรูปแบบที่มากกว่า 1 รูปแบบ ต้องใช้เทคนิคการเขียนโปรแกรมแบบมีเงื่อนไข

การเขียนโปรแกรมแบบมีเงื่อนไขจะเข้ามาช่วยในการตัดสินใจว่า
ถ้า input นี้ จะต้องใช้สมการนี้หรือรูปแบบคำตอบนี้
ถ้า input อีกอันนึงก็ใช้อีกรูปแบบนึง
เริ่มต้น เราจะเรียนรู้การเขียนโปรแกรมแบบมีเงื่อนไขสำหรับคำตอบสองรูปแบบก่อน

โครงสร้างภาษาซีพลัสพลัสที่ใช้ก็คือ

```
if( เงื่อนไข )
{
สิ่งที่จะให้ทำ ฯลฯ
}else
{
สิ่งที่จะให้ทำ ฯลฯ
}
```

ความหมายก็คือ
ถ้า เป็นอย่างนี้
{
จะให้ทำอะไร
}
ถ้าไม่ใช่กรณีแรก แล้ว(พูดง่ายๆก็คือไม่ใช่ตามเงื่อนไข)
{
จะให้ทำอะไรอีก
}

หมายเหตุ โจทย์ที่มักจะใช้การเขียนโปรแกรมแบบมีเงื่อนไขคือ โจทย์ที่มักจะมีคำว่า หรือไม่ , ตรวจสอบ , แบ่งประเภท ฯลฯ

เครื่องหมายที่ใช้กับการเปรียบเทียบเงื่อนไขต่างๆมีดังนี้

= =	เท่ากัน
!=	ไม่เท่ากัน
>	มากกว่า
<	น้อยกว่า
> =	มากกว่าหรือเท่ากับ
< =	น้อยกว่าหรือเท่ากับ

เครื่องหมายที่เกี่ยวข้องกับประพจน์ และค่าประพจน์

เครื่องหมาย && (AND)

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

เครื่องหมาย || (OR)

a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

เครื่องหมาย ! (NOT)

a	!a
true	false
false	true

ตัวอย่าง เขียนโปแกรมรับค่าจำนวนเต็ม 1 จำนวนเพื่อตรวจสอบว่าเป็นจำนวนเต็มบวก หรือไม่ พร้อมแสดงผลออกทางหน้าจอ

ตัวอย่างผลรัน

```
Please input a number : 8
8 is positive number.
```

```
Please input a number : 0
0 is not positive number.
```

```
Please input a number : -2
-2 is not positive number
```

คำอธิบาย

โจทย์ข้อนี้ให้ "รับค่า" จำนวนเต็ม 1 จำนวน เพื่อ "ตรวจสอบ" ว่า "เป็นจำนวนเต็มบวกหรือไม่" (คำว่าหรือไม่ แสดงว่ามีแค่สองแบบ) ให้สังเกตุว่า ผลการรันโปรแกรมนี้ มีได้สองแบบ คือ เป็นจำนวนเต็มบวก หรือ ไม่เป็นจำนวนเต็มบวก

กรณีแบบนี้ แสดงว่าเรามีผลลัพธ์ที่เกิดจากสองเงื่อนไข คือ

ถ้าตัวเลขมากกว่า 0 ผลลัพธ์คือ เป็นจำนวนเต็มบวก

ถ้าตัวเลขไม่มากกว่า 0 ผลลัพธ์คือ ไม่เป็นจำนวนเต็มบวก

ถ้าหากมี 2 เงื่อนไข ให้ผลลัพธ์สองแบบอย่างนี้ เราจะใช้ if -else ในการแยกเงื่อนไขแต่ละแบบ

การใช้คำสั่ง if-else

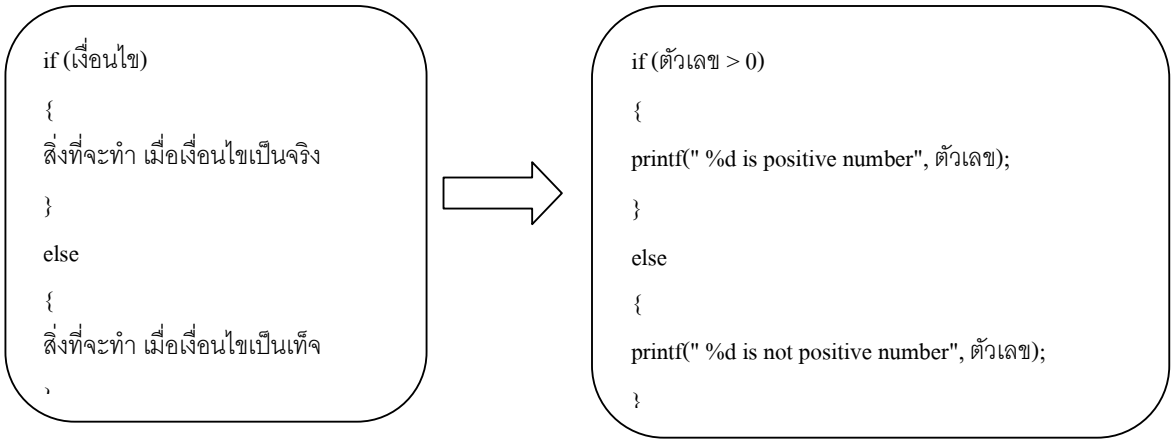
มีรูปแบบคือ

```
if (เงื่อนไข)
{
สิ่งที่จะทำ เมื่อเงื่อนไขเป็นจริง
}
else
{
สิ่งที่จะทำ เมื่อเงื่อนไขเป็นเท็จ
}
```

จำไว้ว่า เมื่อไหร่ก็ตามที่มีคำว่า if ภาษาซี จะทำการเช็คว่า เงื่อนไขที่อยู่ในวงเล็บ เป็นจริงหรือไม่ ถ้าจริง จะทำคำสั่งต่างๆ ที่อยู่ใน เครื่องหมายปีกกา ต่อจาก if แต่ถ้าไม่จริง ก็จะดูว่ามี else ไหม ถ้ามี else ก็จะไปทำคำสั่งใน else แทน

สำหรับการเขียนเงื่อนไข สามารถใช้เครื่องหมายเปรียบเทียบ พวก น้อยกว่ามากกว่า รวมถึง and , or (และ หรือ) จากตารางในหน้า 15-16 ได้

อย่างโจทย์ข้อนี้ เราจะ map จากรูปแบบ ไปเป็นโค้ดได้



ซึ่ง ตัวเลข ในเงื่อนไขของเราก็คือ ตัวเลขที่รับค่าจากคีย์บอร์ด ดังนั้น code ของโจทย์ข้อนี้ ก็จะได้ว่า

```c
#include <stdio.h>

int main()
{
   int number;

    printf("Please input a number : ");
    scanf("%d",&number);

    if(number>0)
    {
    printf(" %d is a positive number.",number);
    }
    else
    {
    printf(" %d is not a positive number.",number);
    }
    return 0;
}
```

```
lease input a number : 5
5 is a positive number.

..Program finished with exit code 0
ress ENTER to exit console.
```

ซึ่งสามารถที่จะเขียนให้อยู่ในรูปแบบของฟังก์ชันที่ไม่มีการ return ค่าได้ดังต่อไปนี้

ไม่มีการ return ค่า จึงเป็น void

```c
#include <stdio.h>

void checkNumber(int number){
    if(number>0)
    {
    printf(" %d is a positive number.",number);
    }
    else
    {
    printf(" %d is not a positive number.",number);
    }
}

int main()
{
   int number;

    printf("Please input a number : ");
    scanf("%d",&number);

    checkNumber(number);
    return 0;
}
```

เพราะฟังก์ชัน checkNumber ไม่มีการ return ค่า จึงเรียกใช้ชื่อฟังก์ชันเฉยๆ

```
ease input a number : 5
 is a positive number.
```

หรือจะเขียนให้อยู่ในรูปแบบของฟังก์ชันที่มีการ return ค่า ดังนี้

ไม่เป็น void ต้องมีคำว่า return ข้างในฟังก์ชัน

```c
#include <stdio.h>

_Bool checkNumber(int number){
    if(number>0)
    {
    return 1;
    }
    else
    {
    return 0;
    }
}

int main()
{
   int number;
    printf("Please input a number : ");
    scanf("%d",&number);

    printf("Is %d a positive number?\n");
    printf("Answer: %d",number,checkNumber(number));
    return 0;
}
```

ค่าที่อยู่หลังคำว่า return ชนิดต้องเป็น boolean ให้ตรงกับชนิดข้อมูลที่ประกาศว่า _Bool

```
ease input a number : 5
 1 a positive number?
swer: 5
```

เพราะฟังก์ชัน checkNumber มีการ return ค่า จึงสามารถนำมาใช้กับ printf ได้

แบบฝึกหัด

1. ระบบเวลาของเรา เริ่มตั้งแต่ 0.00 นาฬิกา จนถึง 23.59 นาฬิกา

ให้เขียนโปรแกรมรับค่าจำนวน เต็ม 1 จำนวน ซึ่งก็คือค่าบอกชั่วโมง เป็นนาฬิกา เพื่อตรวจสอบว่าตัวเลขนั้นอยู่ในช่วง 0-23 นาฬิกาถูกต้องหรือไม่ พร้อมแสดงผลออกทางหน้าจอ

ตัวอย่างผลรัน

```
Please input hour: 8
8 is a correct number.
```

```
Please input hour: 24
24 is a not correct number.
```

ตัวอย่างพร้อมคำอธิบาย

จากโจทย์ข้อที่ 3 หน้า 14 ให้ทำการแสดงผลของดัชนีมวลกาย โดยมีเกณฑ์ดังนี้

Underweight	< 18.50
Normal Range	18.50 - 24.99
Overweight	25 - 29.99
Obese	>= 30

ตัวอย่างผลรัน

```
Please input weight and height : 68 180
BMI =  20.9877
Your BMI is at Normal Range.
```

ตัวอย่างผลรัน

```
Please input weight and height : 100 180
BMI =  30.8642
Your BMI is at Obese.
```

โจทย์ข้อนี้ เป็นโจทย์ ที่ให้ "รับค่า" จำนวนเต็ม 2 จำนวน เพื่อนำมาคำนวณค่า BMI เมื่อเราได้ค่า BMI แล้ว ก็นำค่ามา "ตรวจสอบ" ว่าผลของค่านั้นอ่านได้ว่าอย่างไร ซึ่งมีได้ 4

ให้สังเกตุว่า ผลการรันโปรแกรมนี้ มีได้ 4 แบบ คือ underweight, normal range, overweight, obese

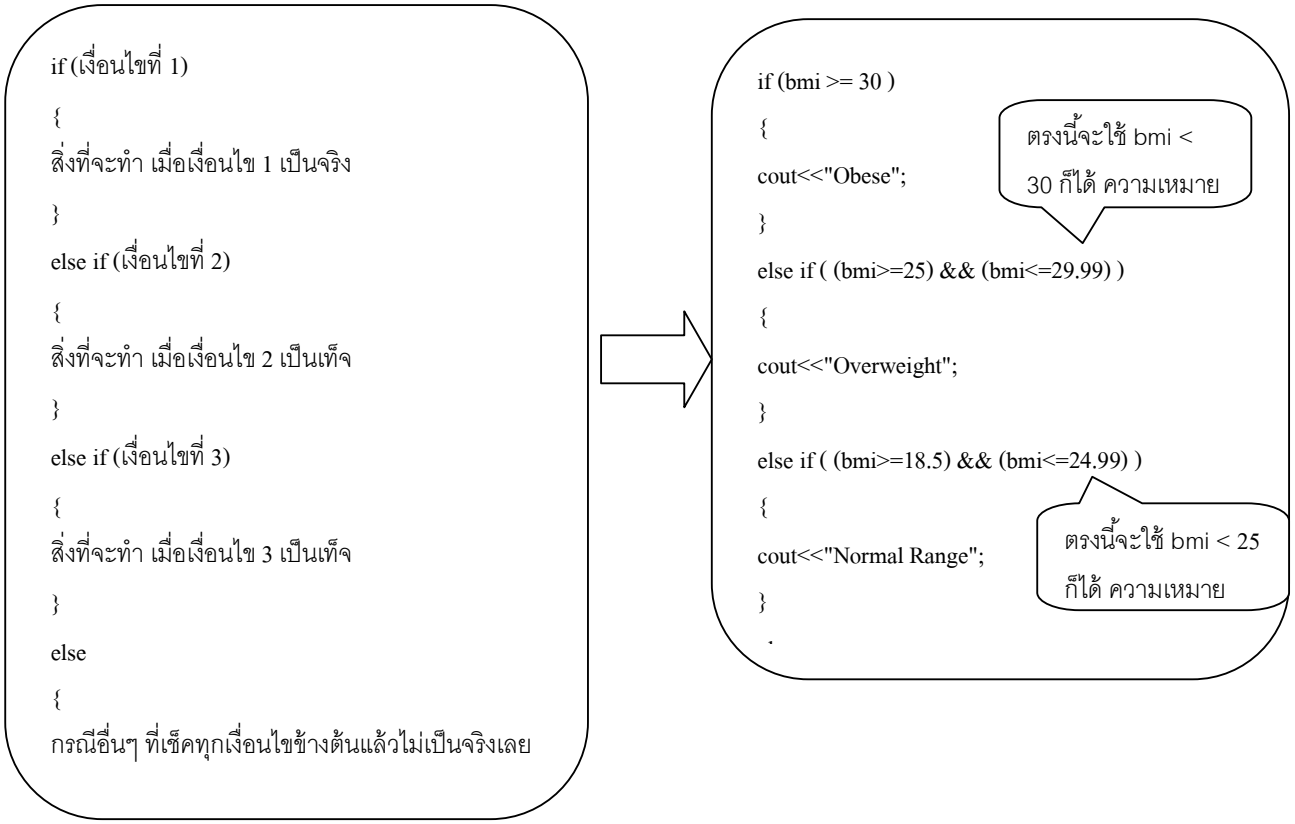
กรณีแบบนี้ แสดงว่าเรามีผลลัพธ์ที่เกิดจาก 4 เงื่อนไข

ถ้าหากมีมากกว่า 2 เงื่อนไข ให้ผลลัพธ์ หลายๆแบบอย่างนี้ เราจะใช้ if -else if (เรียกอีกชื่อว่า block if) ในการแยกเงื่อนไขแต่ละแบบ

การใช้คำสั่ง if-elseif มีรูปแบบดังนี้

```
if (เงื่อนไขที่ 1)
{
สิ่งที่จะทำ เมื่อเงื่อนไข 1 เป็นจริง
}
else if (เงื่อนไขที่ 2)
{
สิ่งที่จะทำ เมื่อเงื่อนไข 2 เป็นเท็จ
}
else if (เงื่อนไขที่ 3)
{
สิ่งที่จะทำ เมื่อเงื่อนไข 3 เป็นเท็จ
}
..... (จะมีกี่ else if ก็ได้เรื่อยๆ ตามจำนวนเงื่อนที่ต้องการเช็ค)
else
{
กรณีอื่นๆ ที่เช็คทุกเงื่อนไขข้างต้นแล้วไม่เป็นจริงเลย
```

อย่างโจทย์ข้อนี้ เราจะ map จากรูปแบบ ไปเป็นโค้ดได้



ซึ่ง ตัวเลข bmi ในเงื่อนไขของเราก็คือ ค่าที่คำนวณจากในข้อ 3 โดยรับค่า น้ำหนัก ส่วนสูง จากคีย์บอร์ด ดังนั้น code ของโจทย์ข้อนี้ จะเป็นโค้ดที่เขียนต่อจาก ข้อ 3 ก็จะได้ว่า

```
22  int main()
23  {
24     int weight, height;
25      printf("Please input weight and height: ");
26      scanf("%d",&weight);
27      scanf("%d",&height);
28
29      float bmi;
30      float height_m =(height/100.0);
31      bmi = weight/( height_m * height_m);
32      printf("BMI =  %f\n",bmi);
33
34      if (bmi>=30.0 )
35      {
36      printf("Your BMI is at Obese");
37      }
38      else if ( (bmi>=25.0) && (bmi<=29.99) )
39      {
40      printf("Your BMI is at Overweight");
41      }
42      else if ( (bmi>=18.5) && (bmi<=24.99) )
43      {
44      printf("Your BMI is at Normal Range");
45      }
46      else
47      {
48      printf("Your BMI is at Underweight");
49      }
50
51      return 0;
52  }
```

หรือจะเขียนเป็นฟังก์ชัน ได้ดังนี้

```c
#include <stdio.h>

void checkBMI(float bmi){
    if (bmi>=30.0 )
    {
    printf("Your BMI is at Obese");
    }
    else if ( (bmi>=25.0) && (bmi<=29.99) )
    {
    printf("Your BMI is at Overweight");
    }
    else if ( (bmi>=18.5) && (bmi<=24.99) )
    {
    printf("Your BMI is at Normal Range");
    }
    else
    {
    printf("Your BMI is at Underweight");
    }

}
```

```c
int main()
{
   int weight, height;
    printf("Please input weight and height: ");
    scanf("%d",&weight);
    scanf("%d",&height);

    float bmi;
    float height_m =(height/100.0);
    bmi = weight/( height_m * height_m);
    printf("BMI =  %f\n",bmi);

    checkBMI(bmi);
    return 0;
}
```

```
lease input weight and height: 68
80
MI =  20.987656
our BMI is at Normal Range
```

แบบฝึกหัด

เขียนโปรแกรมเพื่อคำนวนภาษีรายได้บุคคลโดยมีเกณฑ์ดังนี้

เงินรายได้ (ปี)	**ภาษี (%)**
0 - 150,000	0
150001 - 250,000	5
250001 - 450,000	10
มากกว่า 450,000	15

โดยอัตราภาษีจะคิดจากส่วนที่เกินจากช่วงราคาก่อนหน้า เช่น รายได้ 200,000 บาทจะมีการคิดภาษี 5% จากส่วนที่เกินจาก 150,000 บาทซึ่งคือ 50,000 บาท

ตัวอย่างผลรัน

```
Input income:  140000
tax = 0
```

```
Input income:  200000
tax = 2500
```

```
Input income:  500000
tax = 32500
```

คำอธิบายสั้นพร้อมเฉลย

โจทย์ข้อนี้มี 4 กรณี เพราะมีการคิดเงินภาษีแตกต่างกันไป 4 แบบ ดังนั้นก็จะต้องใช้ if elseif สำหรับเรื่องภาษี มีวิธีการคำนวณที่แตกต่างไปนิดว่า การคำนวณ จะต้องเอาเงินได้ มาหักคำนวณเปอร์เซ็นต์ภาษีเป็นช่วงๆ อย่างเช่น ถ้าเงินได้ 200,000 บาท ก็จะต้องเอา 150,000 บาทแรก มาคิดภาษี ที่ 0% ก่อน แล้วหักลบที่เหลืออีก 50,000 ถึงจะเอาไปคิดภาษี 5%

หรืออย่าง ถ้าเงินได้ 300,000 บาท

เราก็ต้องหัก 150,000 บาทแรกออก ไม่ต้องคิดภาษี

แล้วก็เอา เงินได้ ตั้งแต่ 150,000 ถึง 250,000 บาท มาคิดภาษี 5% ก็จะได้ (250,000-150,000)*0.05

หรือก็คือ 100,000 *0.05

แล้วก็เอา เงินได้ที่เหลือ คือ 300,000 ถึง 250,000 บาท มาคิดภาษี 10% ก็จะได้ (300,000-250,000)*0.05

หรือก็คือ 50,000 *0.1

สุดท้ายก็นำทั้งหมดมารวมกัน ได้เป็น ภาษีที่ต้องจ่าย คือ (250,000-150,000)*0.05 + (300,000-250,000)*0.05

```c
#include <stdio.h>

float taxPayment(float income){
    if( ( income >=0 ) && (income <=150000) )
    {
        return 0;
    }
    else   if( ( income >= 150001 ) && (income <=250000) )
    {
        return (income-150000 )*0.05 ;
    }
    else    if( ( income >=250001 ) && (income <=450000) )
   {
       return ((income-250000 )*0.1) + ((250000 - 150000)*0.05) ;
    } else
   {
     return ((income-450000)*0.15) + ((450000-250000 )*0.1) + ((250000 - 150000)*0.05) ;
   }

}
```

```c
int main()
{
        float income;

    printf("Enter your income ");
    scanf("%f",&income);

    printf("Tax = %f\n",taxPayment(income));

    return 0;
}
```

บทที่ 8

การเขียนโปรแกรมแบบมีการทำซ้ำ (Loop programming)

ชุดคำสั่งประเภทที่ทำให้เกิดการเขียนโปรแกรมแบบมีการทำซ้ำได้ มีทั้งหมดสามโครงสร้างด้วยกัน คือ

1. while
2. do while
3. for

โครงสร้าง while มีดังนี้

```
while(เงื่อนไข)
{
สิ่งที่จะทำซ้ำ
}
```

ซึ่งโครงสร้าง while จะทำซ้ำก็ต่อเมื่อ เงื่อนไขเป็นจริง

โครงสร้าง do while มีดังนี้

```
do
{
สิ่งที่จะทำซ้ำ
} while(เงื่อนไข) ;
```

ซึ่งโครงสร้าง do while จะทำก่อนหนึ่งรอบ และค่อยเช็คเงื่อนไขจะทำซ้ำรอบใหม่ก็ต่อเมื่อ เงื่อนไขเป็นจริง

โครงสร้าง for มีดังนี้

```
for(ค่าเริ่มต้น ; เงื่อนไข ; การเพิ่ม/ลดจำนวนรอบ)
{
สิ่งที่จะทำซ้ำ
}
```

ซึ่งโครงสร้าง for จะทำซ้ำก็ต่อเมื่อ เงื่อนไขเป็นจริง

การเขียนโปรแกรมด้วย 3 โครงสร้างนี้จะประกอบไปด้วยส่วนสำคัญ 4 จุดด้วยกันคือ

1. กำหนดตัวนับรอบเริ่มต้น (จะถูกใช้เพียงครั้งแรกครั้งเดียวเท่านั้น)
2. เงื่อนไขที่จะใช้ในการนับรอบ
3. สิ่งที่จะทำซ้ำ (จะมีกี่บรรทัดก็ได้ อยากให้อะไรซ้ำบ้างก็สั่งๆไป)
4. การเพิ่ม / ลด จำนวนรอบ

ดังนั้นถ้านำโครงสร้างทั้งสามมาเขียนจะพบว่ามีโครงสร้างเหมือนกัน ดังนี้

```
1. กำหนดตัวนับรอบเริ่มต้น
while(2. เงื่อนไข)
{
3. สิ่งที่จะทำซ้ำ
4. เพิ่ม/ลดจำนวนรอบ
}
```

```
1. กำหนดตัวนับรอบเริ่มต้น
do
{
3. สิ่งที่จะทำซ้ำ
4. เพิ่ม/ลดจำนวนรอบ
} while(2. เงื่อนไข);
```

```
for(1. กำหนดตัวนับรอบ ;2. เงื่อนไข ;4. เพิ่ม/ลดจำนวนรอบ )
{
3. สิ่งที่จะทำซ้ำ
}
```

สำหรับลำดับการทำงานของคอมไพเลอร์ จะเป็นดังนี้

Loop while กับ loop for จะมีการทำงานเหมือนกันคือ เรียงตามลำดับข้อได้เลย

อันดับแรก เริ่มต้นโดยกำหนดตัวนับรอบก่อนว่าให้เริ่มต้นมีค่าเท่าไหร่

อันดับสอง เช็คเงื่อนไขว่าจริงหรือไม่

ถ้าเงื่อนไขเป็นจริง ก็จะทำงานในสิ่งที่ทำซ้ำ

สุดท้ายก็เพิ่ม / ลดจำนวนรอบ

สังเกตว่า ใน loop while และ do while ตัวเพิ่มลดจำนวนรอบอยู่ใน { } ซึ่งจะถูกทำซ้ำด้วย จะมาก่อนหน้า ข้อ 3 คือสิ่งที่จะทำซ้ำก็ได้หรือข้างหลังก็ได้ ขึ้นอยู่กับจุดประสงค์ของการเขียนโปรแกรม

ไม่เหมือนกับ loop for ที่จะกำหนดตายตัวเลยว่าตัวเพิ่มลดจำนวนรอบจะถูกทำงานตอนสุดท้ายของแต่ละรอบ

เมื่อไหร่จึงจะใช้ loop

1. เมื่อมีการทำคำสั่งอะไรซ้ำๆ เหมือนกันทุกครั้ง

```c
#include <stdio.h>

int main()
{
    float income;

    printf("Hello World\n");
    printf("Hello World\n");
    printf("Hello World\n");

    return 0;
}
```

```
Hello World
Hello World
Hello World
```

ตรงนี้ทำซ้ำเหมือนกันทั้ง 3 รอบ

เราก็แค่นับยังไงก็ได้ ให้ได้ 3 รอบ

ตัวอย่างเช่น

นับ 1, 2, 3 หรือ นับ 3, 2, 1

นับ 5, 6, 7 หรือ นับ 23, 22, 21

นับ 0, 2, 4 หรือ นับ 7, 5, 3

เราก็แทนที่โค้ดบริเวณที่ซ้ำบรรทัด 15-17 ด้วย โค้ด loop ใดก็ได้ ใน 3 แบบ for / while / do...while

เช่น แทนด้วย

```c
int i;
for( i = 5; i <= 7; i = i + 1)
{
printf("Hello World\n");
}
```

2. เมื่อมีการทำคำสั่งอะไรซ้ำๆ โดยที่มีจุดที่เปลี่ยนแปลงแบบเป็น pattern จำพวก เพิ่มขึ้นหรือลดลง เท่าๆกัน

```c
#include <stdio.h>

int main()
{
    float income;

    printf("Hello World %d \n",1);
    printf("Hello World %d \n",2);
    printf("Hello World %d \n",3);

    return 0;
}
```

```
Hello World 1
Hello World 2
Hello World 3
```

ตรงนี้ทำซ้ำเป็น pattern ที่ต้องเปลี่ยนจาก เลข 1 -> 2 -> 3
ซึ่งจะมีวิธีนับแบบเดียวคือ
นับเพิ่มขึ้นทีละ 1 นับ 1, 2, 3
เราก็แทนที่โค้ดบริเวณที่ซ้ำบรรทัด 15-17 ด้วย โค้ด loop ใดก็ได้ ใน 3 แบบ for / while / do...while
เช่น แทนด้วย

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

```c
#include <stdio.h>

int main()
{
    float income;

    printf("Hello World %d \n",4*10);
    printf("Hello World %d \n",3*10);
    printf("Hello World %d \n",2*10);

    return 0;
}
```

```
Hello World 40
Hello World 30
Hello World 20
```

ตรงนี้ทำซ้ำเป็น pattern ที่ต้องเปลี่ยนจาก เลข 4-> 3 ->2
ซึ่งจะมีวิธีนับแบบเดียวคือ
นับลดลงทีละ 1 นับ 4,3,2
เราก็แทนที่โค้ดบริเวณที่ซ้ำบรรทัด 15-17 ด้วย โค้ด loop ใดก็ได้ ใน 3 แบบ for / while / do...while
เช่น แทนด้วย

```c
int i;
for( i = 4; i >= 2; i = i - 1)
{
printf("Hello World  %d  \n ", i*10);
}
```

3. เมื่อมีการทำคำสั่งอะไรซ้ำๆ หลายๆรอบ

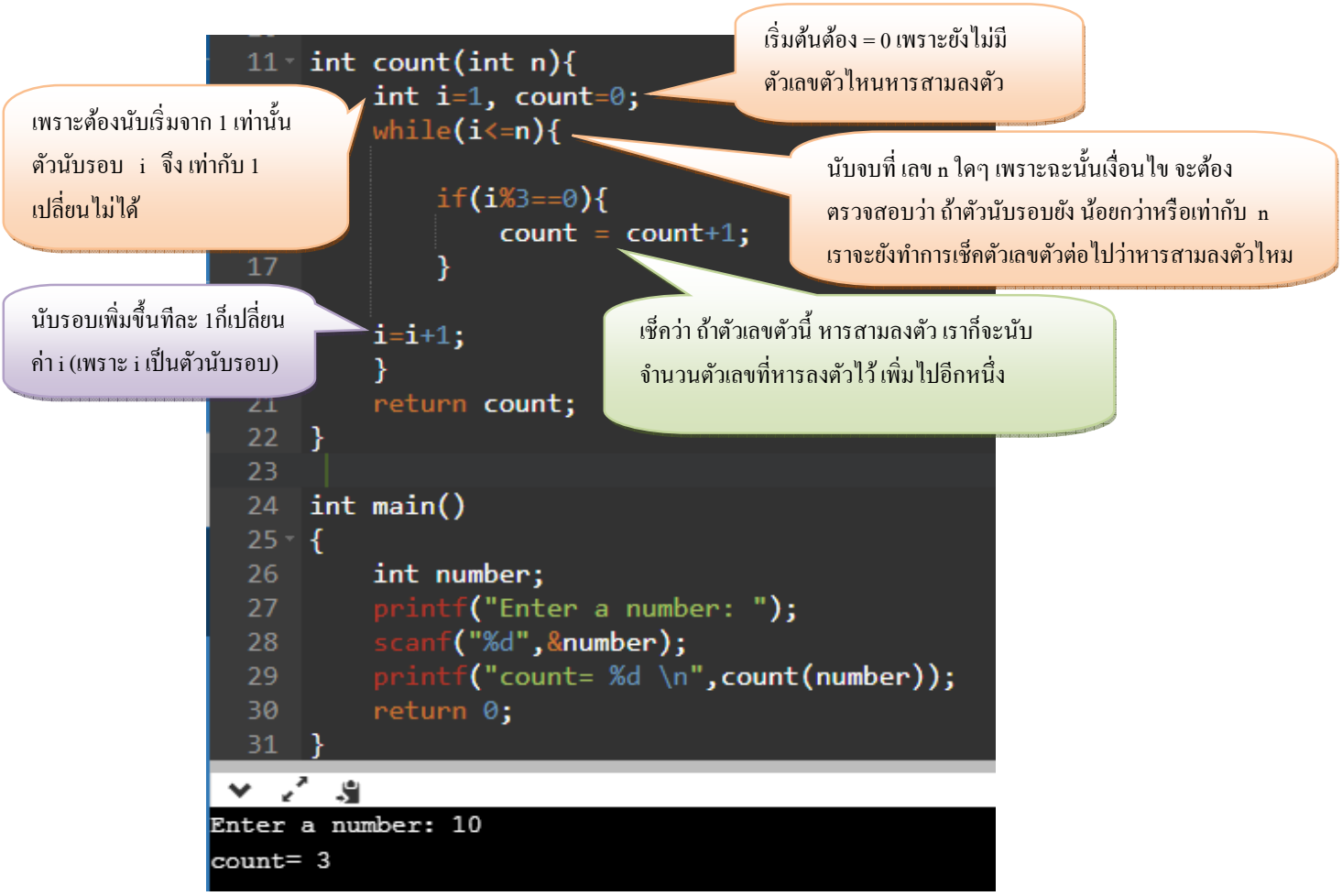
ยกตัวอย่างเช่น

โปรแกรม ให้บวกเลข จาก 1 ถึง n ใดๆ

สิ่งที่ทำซ้ำคือ การบวกตัวเลขทีละตัวๆ

เวลานับรอบ ก็ต้องนับจาก 1, 2,3,..., n-1, n นับเพิ่มขึ้นทีละหนึ่ง นับเริ่มจาก 1

โปรแกรม ให้นับเลขที่หารด้วยสามลงตัว ของตัวเลขจาก 1 ถึง n ใดๆ
สิ่งที่ทำซ้ำคือ การเช็คตัวเลขทีละตัวๆ ว่ามันหารด้วยสาม ลงตัวหรือไม่
ถ้ามันหารลงตัว เราก็บันทึกจำนวนตัวเลขที่หารลงตัวเพิ่มไปอีก 1
ถ้าหารไม่ลงตัวก็ไม่ต้องทำอะไร
เวลานับรอบ ก็ต้องนับจาก 1, 2,3,..., n-1, n นับเพิ่มขึ้นทีละหนึ่ง นับเริ่มจาก 1



นอกจากนี้ เรายังสามารถเขียนลูปซ้อนกันหลายๆชั้นได้ ในที่นี้ จะยกตัวอย่างลูปแค่สองชั้น

เมื่อไหร่จะใช้ลูปซ้อนกัน 2 ชั้น
เมื่อ คำสั่งที่ทำซ้ำชุดนี้ จะต้องถูกทำซ้ำอีก
ยกตัวอย่างเช่น
(จาก รูปด้านซ้ายมือ เราค่อยๆเปลี่ยนโค้ดเป็นตามด้านขวามือ จนได้ลูปซ้อนกันสองชั้น)
จะเห็นว่า สิ่งทำซ้ำชุดแรกคือ การพิมพ์ตัวเลข 1 2 3
และชุดการพิมพ์ {1,2,3} นี้ก็ถูกทำซ้ำ สองครั้ง

```c
#include <stdio.h>
int main()
{
    int number;

    printf("Hello World %d \n",1);
    printf("Hello World %d \n",2);
    printf("Hello World %d \n",3);

    printf("Hello World %d \n",1);
    printf("Hello World %d \n",2);
    printf("Hello World %d \n",3);

    return 0;
}
```

```
Hello World 1
Hello World 2
Hello World 3
Hello World 1
Hello World 2
Hello World 3
```

```c
#include <stdio.h>
int main()
{
    int number;

    int i,j;
    for(i=1;i<=2;i++){
    printf("Hello World %d \n",1);
    printf("Hello World %d \n",2);
    printf("Hello World %d \n",3);
    }

    return 0;
}
```

ดังนั้นจะเห็นว่าชุดการพิมพ์ 1,2,3 อยู่ด้านในลูปที่ทำสองรอบ

```c
#include <stdio.h>
int main()
{
    int number;

    int i,j;
    for(i=1;i<=2;i++){
        for(j=1;j<=3;j++){
    printf("Hello World %d \n",j);
        }
    }

    return 0;
}
```

แล้วเราก็เปลี่ยนการพิมพ์ 1,2,3 ให้เป็นลูปที่ทำ 3 รอบแทน

ยกตัวอย่างเช่น โปรแกรมที่ทำการคูณเลขทุกตัวคล้ายๆกับ แม่สูตรคูณ แต่ตัดมาแค่สามตัว

เช่น เอาเซต {5,6,7} คูณกับ { 1,2,3} แบบพบกันหมด ก็จะได้เป็น

5* 1 = 5, 5*2 = 10, 5*3 = 15

6* 1 = 6, 6*2 = 12, 6*3 = 18

7* 1 = 7, 7*2 = 14, 7*3 = 21

```c
#include <stdio.h>
int main()
{
    int number;

    int i,j;
    for(i=5;i<=7;i++){
        for(j=1;j<=3;j++){
    printf("%d x %d = %d \n",i,j,i*j);
        }
    }

    return 0;
}
```

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
```

คือตัวนับรอบ ของลูปนอกสุด ไล่จากเลข 5, 6, 7

คือตัวนับรอบของลูปใน ไล่จากเลข 1,2,3 ดังนั้น เลข 1,2,3 นี้จะต้องถูก ทำซ้ำ 3 ครั้ง(เพราะลูปนอกครอบมันไว้ 3 รอบ นับจาก 5,6,7)

บทที่ 9

โครงสร้างข้อมูลแบบ Array

จากบทเรียนที่ผ่านมาเรารู้จักกับชนิดข้อมูลแล้ว ก็คือแบ่งข้อมูลออกเป็นประเภทต่างๆ
แต่ว่าคำศัพท์ใหม่ที่เราจะเรียนคือ โครงสร้างข้อมูลนี้ ไม่เหมือนกันกับชนิดข้อมูล เพราะฉะนั้นอย่าสับสน
โครงสร้างข้อมูล คือแนวความคิดที่จะจัดรูปแบบข้อมูลให้เป็นแบบต่างๆ
ซึ่งโครงสร้างแบบ Array นี้ มีแนวคิดที่จะรวมเอาข้อมูลที่มีชนิดเดียวกัน อยู่รวมกันเป็นกลุ่มภายใต้ตัวแปรชื่อเดียวกัน แต่เพิ่มตัวบอกตำแหน่งเข้ามาเท่านั้นเอง

ตัวแปร ชื่อหนึ่งจะเก็บค่าข้อมูลได้หนึ่งค่า แต่ว่าพอตัวแปรตัวนั้นมีโครงสร้างเป็น array แล้วจะสามารถเก็บข้อมูลได้หลายค่า จะกี่ค่าก็ขึ้นอยู่กับว่าตัวแปรตัวนั้นมีขนาดเท่าไหร่ แต่ว่าถึงแม้ว่าหนึ่งตัวแปรเก็บได้หลายค่า แต่ก็ยังคงต้องมีตัวบอกตำแหน่งเพิ่มขึ้นมาเพื่อที่จะบอกว่า ค่าไหน อยู่ตรงไหน
ซึ่งตัวบอกตำแหน่งนี้ จะมีค่าเริ่มต้นตั้งแต่ 0 ถึง N-1 เมื่อให้ N คือขนาดตัวแปรที่มีโครงสร้างเป็น array
อธิบายดังรูป

ตัวแปร ชื่อ X

ตำแหน่ง 0	ตำแหน่ง 1	ตำแหน่ง 2	ตำแหน่ง 3

จาก 0 -> N -1 เมื่อ N = ขนาด อาร์เรย์ ซึ่งตอนนี้ ขนาด N = 4
เพราะฉะนั้นแสดงว่าตัวแปร X สามารถเก็บข้อมูลได้ 4 ค่า

อาเรย์หนึ่งมิติ

9.1 การประกาศตัวแปรที่มีโครงสร้างเป็นอาเรย์
ปกติ การประกาศตัวแปร ทำได้โดย
ชนิดข้อมูล ชื่อตัวแปร ;
การประกาศให้เป็นอารเรย์ ก็แค่เพิ่ม สัญญลักษณ์ [ ] ซึ่งข้างในวงเล็บคือขนาดอารเรย์นั่นเอง
จะได้
ชนิดข้อมูล ชื่อตัวแปร [ขนาดอารเรย์] ;
เช่น int x[4] ;
char y [10];
หรืออาจจะประกาศตัวแปรและกำหนดค่าให้เลยก็ได้ทำได้โดย
ชนิดข้อมูล ชื่อตัวแปร [ขนาดอารเรย์] = { ค่าของข้อมูลเรียงตามลำดับของตำแหน่งในอาเรย์ } ;
เช่น int x[4] = {1,2,3,4};

9.2 การเข้าถึงค่าข้อมูลในตัวแปรอารเรย์

ถ้าตัวแปรปกติ ก็ใช้ ชื่อตัวแปร แล้วจะกำหนดค่าให้หรือนำค่าแสดงมาดู ก็ตามแต่

แต่พอเป็นตัวแปรแบบอารเรย์ ก็เพิ่มตำแหน่งที่ต้องการเข้าไป

เช่น กำหนดค่าให้กับตัวแปรอารเรย์

```
int x[4];
x[0] = 1;
x[1]=2;
x[2]=3;
x[3]=4;
```

หรือแสดงค่าออกมา

```
printf("%d",x[0]);
```

จะเห็นว่าไม่แตกต่างจากตัวแปรปกติ เพียงแค่ต้องแสดงสัญญลักษณ์ [ ] และบอกตำแหน่งข้อมูลเท่านั้นเอง

การเขียนโปรแกรมกับอารเรย์

ถ้าเราต้องการบวกค่าตัวเลขที่อยู่ในอารเรย์ ทั้งหมด ก็สามารถทำได้โดย

```
int main()
{
int x[4];
x[0] = 1;
x[1]=2;
x[2]=3;
x[3]=4;
printf("%d", x[0]+x[1]+x[2]+x[3]);
return 0;
}
```

ถ้ากรณีที่อาเรย์ของเรามีขนาดมากๆ อาจจะ 100 หรือ 1000 ก็จะต้องเขียนยาวมาก เพราะฉะนั้นการเขียนโปรแกรมจะไม่สะดวก และยากต่อการแก้ไข ถ้าสังเกตดูจะพบว่า ตัวแปรอารเรย์ มีตำแหน่งเริ่มจาก 0 ถึง N-1 เมื่อ N คือขนาดอารเรย์ ดังนั้นจะเห็นว่าตำแหน่งของอารเรย์เพิ่มขึ้นทีละ 1 ซึ่งตรงนี้สามารถนำการเขียนโปรแกรมแบบมีการทำซ้ำเข้ามาใช้ได้

เมื่อไหร่จึงจะต้องใช้โครงสร้างข้อมูลแบบอารเรย์

เมื่อ มีการเก็บข้อมูลเป็นชุดๆ มากกว่า 1 ค่า (เพราะชนิดข้อมูลพื้นฐาน int, float , double, _Bool, char สามารถเก็บได้ค่าเดียว)

ยกตัวอย่างเช่น

การรับค่าจากคีย์บอร์ด 5 จำนวน แล้วต้องนำทั้ง 5 จำนวนนี้มาคำนวณในฟังก์ชันเดียวกัน

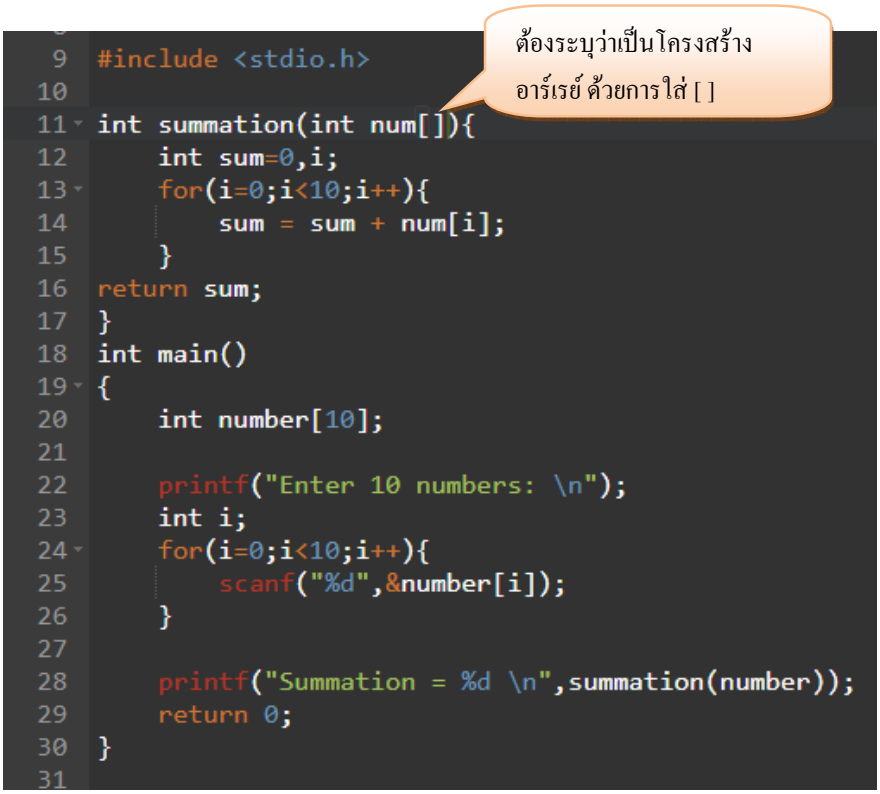
เช่น โปรแกรมหาผลรวมของตัวเลข 10 จำนวนที่รับจากคีย์บอร์ด

ถ้าสมมติเราไม่มีโครงสร้างอาร์เรย์ เราต้องประกาศตัวแปร 10 ตัว มาบวกกันแบบนี้

```c
#include <stdio.h>

int summation(int a,int b, int c,int d,int e, int f, int g, int h, int i, int j){

return a+b+c+d+e+f+g+h+i+j;
}
int main()
{
    int a,b,c,d,e,f,g,h,i,j;

    printf("Enter 10 numbers: \n");
    scanf("%d",&a);
    scanf("%d",&b);
    scanf("%d",&c);
    scanf("%d",&d);
    scanf("%d",&e);
    scanf("%d",&f);
    scanf("%d",&g);
    scanf("%d",&h);
    scanf("%d",&i);
    scanf("%d",&j);

    printf("Summation = %d \n",summation(a,b,c,d,e,f,g,h,i,j));
    return 0;
}
```

ซึ่งมันไม่สะดวก ยิ่งถ้าจำนวนตัวเลขมากขึ้น กว่า 10 เป็น 100 ตัว การประกาศตัวแปร 100 ชื่อเป็นเรื่องลำบาก

ถ้าเปลี่ยนเป็นโครงสร้างแบบอาร์เรย์ เราก็ใช้ตัวแปรชื่อเดิม ชื่อเดียว แค่เปลี่ยน index ที่เรียก สะดวกกว่าเยอะ เพราะในกรณีนี้ ตัวแปร number มีทั้งหมด 10 index ก็แค่อ้างอิงตำแหน่ง number[0] ถึง number[9]




```c
#include <stdio.h>

int summation(int num[]){
    int sum=0,i;
    for(i=0;i<10;i++){
        sum = sum + num[i];
    }
return sum;
}
int main()
{
    int number[10];

    printf("Enter 10 numbers: \n");
    int i;
    for(i=0;i<10;i++){
        scanf("%d",&number[i]);
    }

    printf("Summation = %d \n",summation(number));
    return 0;
}

```

แบบฝึกหัด

1. นับจำนวนตัวเลขที่เป็นเลขคี่ จากค่าที่รับมาจากคีย์บอร์ดทั้งหมด 10 จำนวน

อาเรย์สองมิติ

จากเดิมที่ตัวแปรเก็บค่าได้มากขึ้นแล้ว การทำเป็นสองมิติ คือเพิ่มมิติที่สองให้เก็บค่าได้อีก หมายความว่าจากเดิมตัวแปรเก็บได้ N ค่า ในพื้นที่ N ตำแหน่งนั้นก็จะเก็บได้อีก อันละ M ค่า

เพราะฉะนั้นตัวแปรจะเก็บค่าได้ทั้งหมด N * M ค่า เมื่อ N คือขนาดอาเรย์มิติที่หนึ่ง และ M คือขนาดอาเรย์มิติสอง

ตัวแปร ชื่อ X

ตำแหน่ง 0	ตำแหน่ง 1	ตำแหน่ง 2	ตำแหน่ง 3
ตำแหน่งที่ 0	ตำแหน่งที่ 0	ตำแหน่งที่ 0	ตำแหน่งที่ 0
ตำแหน่งที่ 1	ตำแหน่งที่ 1	ตำแหน่งที่ 1	ตำแหน่งที่ 1
ตำแหน่งที่ 2	ตำแหน่งที่ 2	ตำแหน่งที่ 2	ตำแหน่งที่ 2

มิติที่ 1 จาก 0 -> N -1 เมื่อ N = ขนาด อาร์เรย์มิติที่ 1 ซึ่งตอนนี้ ขนาด N = 4

มิติที่ 2 จาก 0 -> M -1 เมื่อ M = ขนาด อาร์เรย์มิติที่ 2 ซึ่งตอนนี้ ขนาด M = 3

เพราะฉะนั้นแสดงว่าตัวแปร X สามารถเก็บข้อมูลได้ 4 * 3 ค่า

9.3 การประกาศตัวแปรที่มีโครงสร้างเป็นอาเรย์

จากการประกาศของอารเรย์หนึ่งมิติ

การประกาศให้เป็นอารเรย์ ก็แค่เพิ่ม สัญญลักษณ์ [ ] ซึ่งข้างในวงเล็บคือขนาดอารเรย์ของมิตินั้นๆนั่นเอง จะได้

ชนิดข้อมูล ชื่อตัวแปร [ขนาดอารเรย์มิติที่ 1] [ขนาดอารเรย์มิติที่ 2] ;

เช่น int x[4] [3];

char y [10][5];

หรืออาจจะประกาศตัวแปรและกำหนดค่าให้เลยก็ได้ทำได้โดย

ชนิดข้อมูล ชื่อตัวแปร [ขนาดอารเรย์มิติที่ 1] [ขนาดอารเรย์มิติที่ 2] = { {ค่าของข้อมูลเรียงตามลำดับของตำแหน่งในอาเรย์มิติที่สอง } , {ค่าของข้อมูลเรียงตามลำดับของตำแหน่งในอาเรย์มิติที่สอง } } ;

ซึ่งคู่ { } ข้างในแต่ละคู่คือตำแหน่งในอาเรย์มิติที่หนึ่งเรียงตามลำดับ

เช่น int x[4][3] = {{1,2,3} , {4,5,6} ,{7,8,9} , {10,11,12} };

วงเล็บ {1,2,3} คืออาเรย์มิติที่หนึ่ง ตำแหน่งที่ 0

และค่าข้างใน 1, 2, 3 คือค่าในอาเรย์มิติที่สอง เรียงตำแหน่งตามลำดับ จะได้ 1 คือตำแหน่งที่ 0 และ 2 คือตำแหน่งที่ 1 และ 3 คือตำแหน่งที่ 2

* *เปรียบเทียบง่ายๆ เหมือน ตำแหน่งมิติที่ 1 คือ บ้านเลขที่ และตำแหน่งในมิติที่สอง คือ / นั่นเอง เช่น 123/7

9.4 การเข้าถึงค่าข้อมูลในตัวแปรอารเรย์

ก็เพิ่มตำแหน่งที่ต้องการเข้าไปแต่ต้องอ้างถึงตำแหน่งทั้งในมิติที่ 1 และในมิติที่สอง

เช่น กำหนดค่าให้กับตัวแปรอารเรย์

int x[4][3];

x[0][0] = 1;

x[0][1] = 2;

x[0][2] = 3;

x[1] [0] =4;

x[1][1] = 5;

x[1][2] = 6;

x[2] [0] =7;

x[2][1] = 8;

x[2][2] = 9;

x[3] [0] =10;

x[3][1] = 11;

x[3][2] = 12;

หรือแสดงค่าออกมา

printf("%d ", x[0][2]);

ก็จะเห็นว่าเพียงแค่ต้องแสดงสัญญลักษณ์ [ ] ของทั้งสองมิติ และบอกตำแหน่งข้อมูลในทั้งสองมิติเท่านั้นเอง

ยกตัวอย่างเช่น การคำนวณพวก เมทริกซ์

เช่น โปรแกรมหาผลรวมเฉพาะในแต่ละแถว ของเมทริกซ์ขนาด 3*4

แบบฝึกหัด

1. คำนวณ เมทริกซ์ เขียน โปรแกรมหาผลรวมเฉพาะในแต่ละคอลัมน์ ของเมทริกซ์ขนาด 3*4
2. คำนวณเมทริกซ์ เขียนโปรแกรม นับว่าในแต่ละแถว มีจำนวนตัวเลขติดลบ กี่ตัว ของเมทริกซ์ขนาด 3*4