ใบความรู้ที่ 1 โครงสร้างของโปรแกรมภาษาซี

โครงสร้างของโปรแกรมภาษาซี

การเขียนโปรแกรมคอมพิวเตอร์แต่ละภาษา จะมีโครงสร้างของโปรแกรมภาษา แตกต่างกัน ผู้เขียนโปรแกรมจะต้องรู้และเข้าใจ รูปแบบโครงสร้างของภาษานั้น ๆ ว่าประกอบด้วยอะไรบ้าง เพื่อจะได้เขียนได้ถูกต้อง โปรแกรมภาษาซีก็เช่นเดียวกัน โครงสร้างของโปรแกรมภาษาซี ประกอบด้วย 3 ส่วนใหญ่ ๆ ดังนี้

1. ส่วนหัวของโปรแกรม (Preprocessor Directives)
2. ส่วนของฟังก์ชั่นหลัก (The main () function)
3. ส่วนของคำอธิบายโปรแกรม (Comment)

จากส่วนประกอบทั้ง 3 ส่วน แสดงได้ดังรูปที่ 1.1

```
#include<stdio.h>   ------> ส่วนหัวของโปรแกรม (Preprocessor Directives)
main()
{                                  ส่วนของฟังก์ชั่นหลัก (Main function)
 printf("Hello C"); // show c  ------> ส่วนของคำอธิบายโปรแกรม (Comment)
}
```

รูปที่ 1.1 แสดงโครงสร้างของโปรแกรมภาษาซี

ในการเขียนโปรแกรมภาษาซี จะมีส่วนหลัก ๆ ที่สำคัญ 2 ส่วน คือ ส่วนหัวของโปรแกรม (Preprocessor Directives) และส่วนของฟังก์ชั่นหลัก (Main function) โดยรายละเอียดรูปแบบของส่วนต่าง ๆ มีดังนี้

1. ส่วนหัวของโปรแกรม (Preprocessor Directives)

ส่วนหัวของโปรแกรมนี้เรียกว่า Preprocessor Directives ใช้ระบุเพื่อบอกให้คอมไพเลอร์กระทำการใด ๆ ก่อนการแปลผลโปรแกรม ในที่นี้คำสั่ง #include<stdio.h> ใช้บอกกับคอมไพเลอร์ให้นำเฮดเดอร์ไฟล์ที่ระบุ คือ stdio.h เข้าร่วมในการแปลโปรแกรมด้วย โดยการกำหนดจะต้องขึ้นต้นด้วยเครื่องหมาย # (pound sign) เสมอ สำหรับไดเร็กทีฟ (Directive) ที่ใช้กันบ่อย ๆ ได้แก่

1.1 #include เป็นคำสั่งที่ระบุให้คอมไพเลอร์นำเฮดเดอร์ไฟล์เข้าร่วมในการแปลโปรแกรม สามารถเขียนได้ 2 รูปแบบ คือ

1.1.1 #include <ชื่อไฟล์> คอมไพเลอร์จะทำการค้นหาเฮดเดอร์ไฟล์ที่ระบุจากไดเรกทอรีที่ใช้สำหรับเก็บเฮดเดอร์ไฟล์โดยเฉพาะ (ปกติคือไดเรกทอรีชื่อ include)

1.1.2 #include "ชื่อไฟล์" คอมไพเลอร์จะทำการค้นหาเฮดเดอร์ไฟล์ที่ระบุจากไดเรกทอรีเดียวกันกับไฟล์ source code นั้น แต้ถ้าไม่พบก็จะไปค้นหาไดเรกทอรีที่ใช้เก็บเฮดเดอร์ไฟล์โดยเฉพาะ

ตัวอย่างเฮดเดอร์ไฟล์ที่มีการเรียกใช้งาน เช่น

- Stdio.h เป็นไฟล์ที่เก็บฟังก์ชั่นที่เกี่ยวกับการติดต่ออินพุตและเอาต์พุตที่เป็นมาตรฐาน เช่น scanf () (ใช้ในการรับค่าข้อมูลอินพุตผ่านทางคีย์บอร์ดเข้ามาเก็บไว้ในตัวแปร) และ printf() (ใช้ในการแสดงผลเอาต์พุตออกทางจอภาพ)
- Math.h เป็นไฟล์ที่เก็บฟังก์ชั่นเกี่ยวกับการคำนวณทางคณิตศาสตร์ เช่น pow () (ใช้หาค่าเลขยกกำลัง) และ sqrt() (ใช้หาค่ารากที่สอง)
- Conio.h เป็นไฟล์ที่เก็บฟังก์ชั่นที่ใช้ในการเรียกใช้หน่วยอินพุตหรือเอาต์พุตของดอส (DOS) เช่น clrscr() (clear screen ใช้ในการล้างหน้าจอเอาต์พุตและเลื่อนเคอร์เซอร์ไปยังตำแหน่งบนสุดของของจอภาพ) และ gotoxy() (ใช้ในการกำหนดตำแหน่งของเคอร์เซอร์บนจอภาพ)

ตัวอย่างที่ 1.1 #include directive

```
#include <stdio.h>          //หมายถึงการอ่านไฟล์จากไลบารี่ stdio.h เข้ามาด้วย
#include "stdio.h"          //หมายถึงการอ่านไฟล์จากไลบารี่ stdio.h เข้ามาด้วย
#include <sample.c>         //หมายถึงการอ่านไฟล์ sample.c เข้ามาด้วย
```

1.2 #define เป็นคำสั่งที่ใช้ในการกำหนดค่าคงที่หรือมาโคร (เป็นส่วนของคำสั่งหรือการกำหนดค่าที่เขียนแทนด้วยชื่อ เพื่อเรียกใช้ในส่วนคำสั่งของโปรแกรม) ซึ่งการกำหนดมาโคร มีรูปแบบดังนี้

```
#define Name Value
```

โดยที่

#define	เป็นคำที่จะต้องประกาศเพื่อบอกให้รู้ว่าบรรทัดที่มีคำนี้เป็นการประกาศมาโคร
Name	เป็นชื่อที่ตั้งตามหลักการตั้งชื่อบ่งชี้ ชื่อนี้จะถูกเรียกใช้ในส่วนของคำสั่งในโปรแกรม
Value	เป็นส่วนของค่าคงที่

ตัวอย่างที่ 1.2 define directive

```
#define PI 3.1417        // กำหนดค่า PI มีค่าเท่ากับ 3.1417
#define NUM 10           // กำหนดค่า NUM มีค่าเท่ากับ 10
```

2. ส่วนของฟังก์ชั่นหลัก (Main Function)

ฟังก์ชั่นหลักของภาษาซี คือ ฟังก์ชั่น main() ซึ่งโปรแกรมภาษาซีทุกโปรแกรมจะต้องมีฟังก์ชั่นนี้อยู่ในโปรแกรมเสมอ ขาดฟังก์ชั่นนี้ไปไม่ได้ เป็นส่วนที่ใช้เขียนคำสั่งต่าง ๆ เพื่อให้โปรแกรมทำงานตามที่ต้องการ โดยขอบเขตของฟังก์ชั่นจะเริ่มต้นที่ เครื่องหมาย { (ปีกกาเปิด) และสิ้นสุดที่เครื่องหมาย } (ปีกกาปิด) การเขียนคำสั่งต่าง ๆ จะอยู่ภายในเครื่องหมาย { และ } เมื่อสิ้นสุดคำสั่งใด ๆ จะต้องลงท้ายด้วย ; (Semicolon) เสมอ

ฟังก์ชั่น main() สามารถเขียนในรูปแบบของ void main ก็ได้ มีความหมายเหมือนกันว่า ฟังก์ชั่น main() จะไม่มีอาร์กิวเมนต์ (argument) คือไม่มีการรับค่าใด ๆ เข้ามาประมวลผลภายในฟังก์ชั่น และจะไม่มีการคืนค่ากลับออกไปจากฟังก์ชั่นด้วย

คืนค่าใด ๆ กลับออกไปจากฟังก์ชั่น

main() เทียบเท่ากับ void main(void)

ไม่รับค่าใด ๆ เข้ามาในฟังก์ชั่น

argument คือ ตัวรับค่าเข้ามาในฟังก์ชั่น

parameter คือ ค่าที่ส่งไปยังฟังก์ชั่น

ค่าของ argument และ parameter ต้องเป็นข้อมูลชนิดเดียวกัน เช่น หากกำหนดให้ argument เป็นข้อมูลชนิดตัวอักษรแล้ว parameter ที่ส่งไปก็ต้องเป็นชนิดตัวอักษรด้วย

ตัวอย่างที่ 1.3 argument และ parameter

```
#include<stdio.h>
void show(char a)      → argument(รับ)รับตัวอักษร 'z' เข้ามาในฟังก์ชั่น
{
        printf("%c",a);
}
void main(void)
{
        show('z');     → parameter(ส่ง) ส่งตัวอักษร 'z' ไปยังฟังก์ชั่น show()
}
```

ซึ่งคำสั่งต่าง ๆ ที่เขียนจะภายในฟังก์ชั่น main() ประกอบด้วย

- **การประกาศตัวแปร (Variable Declaration)** เป็นส่วนที่บอกให้คอมพิวเตอร์รู้ว่าตัวแปรที่ใช้ในโปรแกรมนี้มีตัวแปรชนิดใด ชื่ออะไร เพื่อเป็นที่เก็บข้อมูลและใช้อ้างอิงในการคำนวณค่า
- **อินพุต (Input)** เป็นส่วนที่ทำการอ่านค่าของตัวแปรเข้ามา ซึ่งการอ่านค่าอาจจะอ่านเข้ามาทางคีย์บอร์ดก็ได้
- **การกำหนดค่าหรือการคำนวณ (Assignment or Computation)** เป็นส่วนที่ใช้ในการกำหนดค่าให้กับตัวแปร และการคำนวณ ซึ่งในสองอย่างนี้อาจจะเกิดขึ้นในเวลาพร้อมๆ กัน หรือเพียงอย่างใดอย่างหนึ่งก็ได้ ขึ้นอยู่กับขั้นตอนในการเขียนโปรแกรม
- **เอาต์พุต (Output)** เป็นส่วนที่ทำการแสดงผลข้อมูลออกทางจอภาพ หรือทางสื่ออื่น ๆ เช่น ทางเครื่องพิมพ์ จอภาพ เป็นต้น

3. ส่วนของคำอธิบายโปรแกรม (Comment)

ส่วนของคำอธิบายโปรแกรม (Comment) หรือคอมเมนต์ คือส่วนที่เป็นหมายเหตุของโปรแกรม มีไว้เพื่อให้ผู้เขียนโปรแกรมใส่ข้อความอธิบายกำกับลงไปใน source code ซึ่งคอมไพเลอร์จะข้ามการแปลผลในส่วนที่เป็นคอมเมนต์นี้ ส่วนของคอมเมนต์อาจจะมีหรือไม่มีก็ได้ในโปรแกรม คอมเมนต์ในโปรแกรมภาษาซีมี 2 แบบ คือ

- คอมเมนต์แบบบรรทัดเดียว ใช้เครื่องหมาย //
- คอมเมนต์แบบหลายบรรทัด ใช้เครื่องหมาย /* และ */

ตัวอย่างที่ 1.4 การเขียนคอมเมนต์ในโปรแกรม

```
// Comment only one line
#include <stdio.h>
main ()
{
    printf ("hello") ;
/* comment
            many  line */
}
```

ข้อควรระวังในการใช้คอมเมนต์ คือ ในกรณีที่ใช้คอมเมนต์แบบหลายบรรทัด จะไม่สามารถใช้คอมเมนต์ซ้อนคอมเมนต์ได้ ดังรูปที่ 1.2 มิฉะนั้นจะก่อให้เกิดข้อผิดพลาดในการคอมไพล์

```
/* comment1 */        /* comment2 */        /* comment3 */    ✓

/* comment1          /* comment2 */          comment3 */     ✗
```

รูปที่ 1.2 การใช้คอมเมนต์แบบหลายบรรทัด

จะเห็นว่าในกรณีที่ต้องการใส่คอมเมนต์หลาย ๆ บรรทัดติดกันนั้น คอมเมนต์แบบหลายบรรทัดจะช่วยประหยัดเวลาในการใส่คอมเมนต์ได้มากกว่าการใช้คอมเมนต์แบบบรรทัดเดียว แต่ก็ควรระมัดระวังในการใช้งานด้วย

ตัวอย่างที่ 1.5 โปรแกรมภาษาซีที่มีส่วนประกอบครบตามโครงสร้าง

```
/* This is the first program with C Language*/ → คำอธิบายโปรแกรม
#include<stdio.h> → ส่วนหัวของโปรแกรม (Preprocessor Directives)
main()
{
        float answer;
        int dividend, divisor;      } การประกาศตัวแปร

        printf("Enter Dividend Number: "); → เอาต์พุต
        scanf("%d",&dividend);      → อินพุต
        printf("Enter Divisor Number:   ");
        scanf("%d",&divisor);
        answer = dividend/divisor;  → การคำนวณและการกำหนดค่า
        printf("The answer is %f",answer);
}
                                    } ส่วนของฟังก์ชั่น
```