

LAB 10 - Structure

1. ชนิดข้อมูลแบบโครงสร้าง

ในการแก้ปัญหาบางอย่างเราต้องการเขียนโปรแกรมเพื่อจัดการข้อมูลที่ประกอบด้วยข้อมูลย่อย ๆ ทั้งที่เป็นชนิดเดียวกันและต่างชนิดกัน ทว่าข้อมูลเหล่านั้นมักถูกใช้งานร่วมกันอยู่เป็นประจำ ลองพิจารณาลักษณะงานเกี่ยวกับทะเบียนนิสิต ข้อมูลนิสิตหนึ่งคนประกอบด้วยข้อมูลย่อยหลายส่วน อาทิเช่น รหัสประจำตัว ชื่อ นามสกุล ภาควิชา อายุ เกรดเฉลี่ย อาจารย์ที่ปรึกษา ฯลฯ แม้ว่าเราจะสามารถเขียนโปรแกรมโดยกำหนดให้ข้อมูลเหล่านี้ถูกแยกเก็บไว้ในตัวแปรที่แตกต่างกันได้ก็ตาม แต่โปรแกรมที่ได้จะเต็มไปด้วยตัวแปรที่ใช้เก็บทั้งข้อมูลนิสิตปะปนอยู่กับตัวแปรอีกหลายตัวที่ใช้สำหรับจุดประสงค์อื่นในโปรแกรม อันมีผลทำให้โปรแกรมยากต่อการทำความเข้าใจและแก้ไขเพิ่มเติมในภายหลัง

ในภาษา C รวมถึงภาษาโปรแกรมอีกหลายภาษาซึ่งรองรับการใช้งาน โครงสร้าง (structure) ที่อนุญาตให้เรานำข้อมูลย่อยที่อาจประกอบด้วยข้อมูลชนิดต่างมารวมไว้เป็นกลุ่มเดียวกัน โดยที่การอ้างอิงจะกระทำผ่านตัวแปรตัวเดียวกันทั้งหมด การทำเช่นนั้นนอกจากจะทำให้โปรแกรมดูเป็นระเบียบขึ้นแล้ว การใช้ structure ยังมีประโยชน์อย่างมากในการรวมข้อมูลเป็นกลุ่มเพื่อส่งไปประมวลผลใน function อื่น ๆ ผ่านทาง parameter เพียงตัวเดียว

2. นิยามโครงสร้าง

เนื่องจากข้อมูลที่เราจะรวมไว้เป็นกลุ่มเดียวกันอาจมีชนิดข้อมูลที่แตกต่างกันออกไป ซึ่งต่างจาก array ที่ข้อมูลซึ่งถูกมารวมกันจะต้องเป็นข้อมูลประเภทเดียวกัน เราจึงจำเป็นต้องสร้างนิยามโครงสร้างข้อมูลที่ชัดเจนขึ้นมาเสียก่อน ว่าข้อมูลนั้นจะประกอบด้วย สมาชิก (member) ที่มีรูปแบบข้อมูลชนิดใดบ้าง และสมาชิกแต่ละตัวถูกอ้างถึงอย่างไร ในการนิยาม structure ในภาษา C นั้นใช้คีย์เวิร์ด struct (ย่อมาจากคำว่า structure) ซึ่งมีรูปแบบดังนี้

```
struct StructureName {
    DataType1 var1;
    DataType2 var2;
    :
    DataTypeN varN;
};
```

ชื่อ-นามสกุล.....รหัสนิสิต.....หมู่.....

รูปแบบข้างต้น เป็นการนิยาม structure ชื่อ *StructureName* ซึ่งประกอบไปด้วยสมาชิกจำนวน *N* ตัว สมาชิกมีชนิดข้อมูลเป็น *DataType1* และถูกอ้างอิงผ่านชื่อ *var1* สมาชิกตัวถัดมามีชนิดข้อมูลเป็น *DataType2* และถูกอ้างอิงผ่านชื่อ *var2* เช่นนี้เรื่อยไป การนิยาม structure ต้องปรากฏอยู่ภายนอก function ใด ๆ เสมอ

ตัวอย่าง 2.1 การนิยาม structure ชื่อ *StdInfo* เพื่อใช้เก็บข้อมูลนิสิตแต่ละคน ซึ่งประกอบด้วยสมาชิกดังนี้

- รหัสประจำตัวนิสิต เป็นข้อมูลชนิดตัวเลขจำนวนเต็ม อ้างอิงโดยใช้ชื่อ *id*
- ชื่อนิสิต เป็นข้อมูลชนิดข้อความความยาวไม่เกิน 30 อักขระ อ้างอิงโดยใช้ชื่อ *name*
- คะแนนนิสิต เป็นข้อมูลชนิดตัวเลขจำนวนจริง อ้างอิงโดยใช้ชื่อ *score*

```
struct StdInfo{
    int id;
    char name[30];
    float score;
};
```

ตามที่ได้กล่าวไว้ข้างต้น การนิยาม structure ภายในโปรแกรมจะต้องอยู่นอก function อื่น ๆ เสมอ ตัวอย่างเช่น

```
#include<stdio.h>
struct StdInfo{
    int id;
    char name[30];
    float score;
};
int main(){
    :
}
```

แบบฝึกหัดที่ 2.1: ให้นิยาม structure ชื่อ *VehicleInfo* สำหรับเก็บข้อมูลรถยนต์แต่ละคัน ซึ่งประกอบด้วยสมาชิกดังนี้

- ยี่ห้อรถ เป็นข้อมูลชนิดข้อความ ความยาวไม่เกิน 20 อักขระ อ้างอิงโดยใช้ชื่อ *make*
- ทะเบียนรถ เป็นข้อมูลชนิดข้อความ ความยาวไม่เกิน 10 อักขระ อ้างอิงโดยใช้ชื่อ *plate*

- สีรถ เป็นข้อมูลชนิดข้อความ ความยาวไม่เกิน 10 อักขระ อ้างถึงโดยใช้ชื่อ *color*
- ปีที่ผลิต เป็นข้อมูลชนิดตัวเลขจำนวนเต็ม อ้างถึงโดยใช้ชื่อ *year*

```
struct _____ {
    _____;
    _____;
    _____;
    _____;
};
```

การใช้คีย์เวิร์ด `struct` ข้างต้นนั้น เป็นเพียงการนิยามโครงสร้างขึ้นมาเท่านั้น ยังมิได้มีผลทำให้โปรแกรมสร้างเนื้อที่สำหรับเก็บข้อมูลขึ้นมาภายในหน่วยความจำแต่อย่างใด การนำ `structure` มาเก็บข้อมูลจริง ๆ จะต้องมีการประกาศตัวแปรที่ระบุชนิดข้อมูลเป็นชื่อ `structure` นั้น ๆ เสียก่อนดังรูปแบบต่อไปนี้

```
struct StructureName structVar;
```

โดย `StructureName` คือชื่อของ `structure` ที่ได้ทำการนิยามไปแล้ว และ `structVar` คือชื่อตัวแปรที่นำมาใช้อ้างอิงถึงข้อมูลภายใน `structure`

จะเห็นว่าคำสั่งข้างต้นนั้น มีรูปแบบเช่นเดียวกับการประกาศตัวแปรทั่วไปทุกประการ ดังนั้นการใช้งานคีย์เวิร์ด `struct` จึงเปรียบเสมือนการสร้างชนิดข้อมูลขึ้นมาใหม่ นั่นเอง ซึ่งหมายความว่า นอกเหนือจากการประกาศตัวแปรแล้ว เรายังสามารถนำชื่อ `structure` ไปใช้ในส่วนอื่นของโปรแกรมได้อีกด้วย อาทิเช่น ใช้ระบุชนิดของข้อมูลของพารามิเตอร์สำหรับ `function` ระบุชนิดข้อมูลที่ `function` ทำการคืนค่า หรือแม้กระทั่งนำไปนิยามสมาชิกใน `structure` อื่น ๆ

เนื่องจากตัวแปรชนิด `structure` ไม่ได้เป็นตัวแปรที่เก็บค่าเพียงค่าเดียว แต่เป็นเหมือนตัวแทนกลุ่มข้อมูลที่มีรูปแบบตาม `structure` นั้น ๆ การเข้าถึงข้อมูลภายใน `structure` จึงต้องมาระบุให้ชัดเจนว่าข้อมูลชิ้นใดที่ถูกอ้างถึง ซึ่งทำได้โดยการระบุชื่อของสมาชิกต่อท้ายชื่อตัวแปรแบบ `structure` คั่นด้วยเครื่องหมายจุด (`.`)

ดังตัวอย่าง

```
structVar.memberName
```

เช่นเดียวกับ array การอ้างถึงสมาชิกใน structure เช่นนี้จะมีการใช้งานเสมือนเป็นตัวแปรโดดโดยตัวหนึ่งที่มีชนิดข้อมูลตามที่กำหนดให้สมาชิกในระหว่งการนิยาม structure นั่นคือหากเราใช้การอ้างอิงนี้เป็นส่วนหนึ่งของนิพจน์ใด ๆ ค่าของสมาชิกจะถูกดึงออกมาใช้เพื่อประเมินค่าของนิพจน์นั้น ๆ ในทางตรงกันข้าม หากเราวางการอ้างอิงนี้ไว้ทางซ้ายของเครื่องหมาย = ในคำสั่งให้ค่ากับตัวแปร ค่าสมาชิกตำแหน่งนี้จะถูกเปลี่ยนค่าไปตามค่าที่กำหนดให้

ตัวอย่าง 3.1: โปรแกรมด้านล่างทำการนิยาม structure ชื่อ Vector เพื่อใช้แทนข้อมูลแบบเวกเตอร์ 3 มิติ ภายใน structure ประกอบด้วยสมาชิกชื่อ x y และ z ที่มีชนิดข้อมูลเป็น float ใช้สำหรับเก็บค่าในแต่ละแกนของเวกเตอร์ภายในโปรแกรมหลัก จะมีการสร้างตัวแปรแบบ Vector ขึ้นมาหนึ่งตัวและกำหนดให้มีค่าเป็น (3,4,5) จากนั้นจึงนำค่าเหล่านี้มาแสดงบนหน้าจอในรูปแบบเวกเตอร์

```
#include<stdio.h>
struct Vector{
    float x;
    float y;
    float z;
};
int main(){
    struct Vector v1;
    v1.x = 3; v1.y = 4; v1.z = 5;
    printf("Vector v1 = (%.2f,%.2f,%.2f) ",v1.x,v1.y,v1.z);
}
```

ผลลัพธ์
Vector v1 = (3.00,4.00,5.00)

จากด้านบนเราสามารถปรับเปลี่ยนโปรแกรม โดยสร้างฟังก์ชันที่ทำหน้าที่พิมพ์ข้อมูล โดยส่งเวกเตอร์ทั้งชุดซึ่งเก็บอยู่ใน structure v1 จากฟังก์ชันเมนมาพิมพ์ที่ฟังก์ชันนี้ดังตัวอย่าง

```
#include<stdio.h>
struct Vector{
    float x;
    float y;
    float z;
};

void PrintVector(struct Vector v){
    printf("Vector v1 = (%.2f,%.2f,%.2f) ",v1.x,v1.y,v1.z);
}

int main(){
    struct Vector v1;
    v1.x = 3; v1.y = 4; v1.z = 5;
    PrintVector(v1);
}
```

ให้สังเกตภายในวงเล็บ

การส่งเวกเตอร์ v1 ไปให้
ฟังก์ชัน PrintVector

แบบฝึกหัด 3.1 : เราจะทำการนิยาม function ขึ้นมา 2 function เพื่อจัดการข้อมูลเกี่ยวกับเวกเตอร์ และนำไปใช้ต่อไปในแบบฝึกหัดหลายๆ function แรกคือ ReadVector ซึ่งทำการรับข้อมูลเวกเตอร์สามมิติ จากผู้ใช้และส่งค่าคืนกลับมาในรูปแบบโครงสร้าง Vector อีก function หนึ่งคือ PrintVector ใช้สำหรับ แสดงผลข้อมูลในโครงสร้าง Vector ออกทางหน้าจอ จงเติมคำสั่งลงในช่องว่างเพื่อให้โปรแกรมทำงานได้ ผลลัพธ์ตรงกับตัวอย่างที่กำหนด

```
#include<stdio.h>
struct Vector{
    float x,y,z;
};

struct Vector ReadVector(){
    struct Vector v;
    printf("X element: ");
    scanf("%f",_____);
    printf("Y element: ");
    _____;
    _____;
    return v;
}

void PrintVector(struct Vector v){
    printf("(%.2f,%.2f,%.2f)"_____);
}
```

ตัวอย่างผลการทำงาน

```
Enter a vector  
X element: 6  
Y element: 13  
Z element: 1  
You justed enter a vector (6.00,13.00,1.00)
```

ตัวอย่าง 3.2 : โปรแกรมต่อไปนี้แสดงการนิยาม function *VectorSize* เพื่อใช้ในการคำนวณขนาดของเวกเตอร์ ซึ่งสำหรับเวกเตอร์ $v = (v_x, v_y, v_z)$ ขนาดของเวกเตอร์ v มีนิยามดังต่อไปนี้

$$|v| = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

ชื่อ-นามสกุล.....รหัสนิสิต.....หมู่.....

โปรแกรมนี้อาศัยการนิยาม structure *Vector* จากแบบฝึกหัดที่แล้ว และการนิยาม function *ReadVector* และ *PrintVector* เพื่อรับข้อมูลเวกเตอร์จากผู้ใช้และแสดงผลข้อมูลเวกเตอร์ และใช้ *Library math.h* ช่วยในการหาค่ารากที่สอง และค่ายกกำลัง

```
#include<stdio.h>
#include<math.h>
// ในการเขียนโปรแกรม ให้คัดลอกนิยาม structure Vector มาปะในตำแหน่งนี้

// ในการเขียนโปรแกรม ให้คัดลอก function ReadVector มาปะในตำแหน่งนี้

// ในการเขียนโปรแกรม ให้คัดลอก function PrintVector มาปะในตำแหน่งนี้

float VectorSize(struct Vector v){
    return sqrt(pow(v.x,2) + pow(v.y,2) + pow(v.z,2));
}

int main(){
    struct Vector v;
    printf("Enter vector v\n");
    v = ReadVector();
    printf("The size of the vector ");
    PrintVector(v);
    printf(" is %.2f\n",VectorSize(v));
}
```

ตัวอย่างผลการทำงาน

```
Enter vector v
X element: 3
Y element: 7
Z element: -2
The size of the vector (3.00,7.00,-2.00) is 7.87
```

ชื่อ-นามสกุล.....รหัสนิสิต.....หมู่.....

แบบฝึกหัดที่ 3.2: สำหรับเวกเตอร์ $u = (u_x, u_y, u_z)$ และเวกเตอร์ $v = (v_x, v_y, v_z)$ ผลคูณจุด (dot product) ของเวกเตอร์ u และ v มีสัญลักษณ์เป็น $u \cdot v$ ให้ผลลัพธ์เป็นปริมาณสเกลาร์ที่มีค่าตามสูตร

$$u \cdot v = u_x v_x + u_y v_y + u_z v_z$$

จงทำโปรแกรมต่อไปนี้ให้สมบูรณ์เพื่อให้โปรแกรมรับเวกเตอร์สองจำนวนจากผู้ใช้ และแสดงผลคูณจุดที่เกิดจากเวกเตอร์ทั้งสอง

ตัวอย่างผลการทำงาน

```
Enter vector u
X element: -1.2
Y element: 3
Z element: 0.5
Enter vector v
X element: 4
Y element: 2
Z element: 1.8
u * v = 2.10
```


แบบฝึกหัดที่ 3.3: สำหรับเวกเตอร์ $u = (u_x, u_y, u_z)$ และเวกเตอร์ $v = (v_x, v_y, v_z)$ ผลคูณไขว้ (cross product) ของเวกเตอร์ u และ v มีสัญลักษณ์เป็น $u \times v$ ให้ผลลัพธ์เป็นปริมาณเวกเตอร์ที่มีค่าตามสูตร

$$u \times v = (u_y v_z - u_z v_y, u_z v_x - u_x v_z, u_x v_y - u_y v_x)$$

จงทำโปรแกรมต่อไปนี้ให้สมบูรณ์เพื่อให้โปรแกรมรับเวกเตอร์สองจำนวนจากผู้ใช้ และแสดงผลคูณไขว้ที่เกิดจากเวกเตอร์ทั้งสอง

```
#include<stdio.h>
// คัดลอกนิยาม structure Vector มาปะในตำแหน่งนี้
// คัดลอก function ReadVector มาปะในตำแหน่งนี้
// คัดลอก function PrintVector มาปะในตำแหน่งนี้

_____ CrossVectors(struct Vector u, struct Vector v){
    struct Vector product;
    _____;
    _____;
    _____;
    _____; }

int main(){
    struct Vector u,v;
    printf("Enter vector u\n");
    u = ReadVector();
    _____;
    _____;
    printf("u x v = ");
    PrintVector(CrossVectors(u,v));
}
```

ตัวอย่างผลการทำงาน

```

Enter vector u
X element: 1.5
Y element: -5
Z element: 20
Enter vector v
X element: 0
Y element: 1
Z element: 3
u x v = (-35.00,-4.50,1.50)

```

4. การใช้งาน structure ร่วมกับ array

ในหัวข้อที่ผ่านมาเรารู้จักการประกาศตัวแปรแบบ structure สำหรับสิ่งของ หรือวัตถุเพียงชิ้นเดียว เช่น นิสิตหนึ่งคน หรือเวกเตอร์หนึ่งเวกเตอร์ ในบางครั้งเราจำเป็นต้องประมวลผลข้อมูลเกี่ยวกับวัตถุต่าง ๆ เหล่านี้ มากกว่าหนึ่งชิ้น ดังเช่น ตัวอย่างงานด้านระเบียบนิสิตที่ได้กล่าวมาในตอนต้นที่โปรแกรมต้องสามารถประมวลผล ข้อมูลนิสิตซักันหลายๆคนได้ โดยที่ข้อมูลของนิสิตแต่ละคนแม้จะแตกต่างกันแต่ก็มีโครงสร้างข้อมูลที่เหมือนกัน ดังนั้น array ของ structure จึงเหมาะสมที่สุดสำหรับการจัดการกับข้อมูลในลักษณะนี้

ตามที่ได้กล่าวมาข้างต้นแล้วว่า structure ที่นิยามเรียบร้อยแล้วสามารถนำมาใช้เหมือนกับเป็นชนิดข้อมูล ชนิดหนึ่งได้ทันที ดังนั้นการประกาศตัวแปรแบบ array of structure จึงมีรูปแบบเหมือนกับการประกาศ array ทั่ว ๆ ไปที่ระบุชนิดข้อมูลให้ตรงกับชื่อของ structure นั้น ๆ เท่านั้น ตัวอย่างเช่น หากเราต้องการประกาศตัวแปร ชื่อ student เพื่อเป็น array หนึ่งมิติของ structure StdInfo ที่นิยามไว้ในตัวอย่าง 2.1 จำนวน 10 คน

เราสามารถใช้อ้างอิง

```
struct StdInfo student[10];
```

การเข้าถึงข้อมูลใน array ของ structure จะมีรูปแบบเหมือนการเข้าถึงข้อมูลใน array ของข้อมูลพื้นฐาน ทั่ว ๆ ไป ต่างกันตรงที่พจน์ที่เกิดจากการอ้างอิงตำแหน่งใน array ด้วย index จะไม่ได้ให้ค่าโดด ๆ อีกต่อไป แต่เป็นนิพจน์ที่อ้างถึง structure เช่นนิพจน์

```
student[5]
```

ชื่อ-นามสกุล.....รหัสนิสิต.....หมู่.....

เป็นนิพจน์ที่แทน structure ของนิสิตหมายเลข 5 (นิสิตคนที่ 6 ใน array) ดังนั้นการเข้าถึงข้อมูลภายใน structure จึงทำได้โดยระบุชื่อสมาชิกของ structure ต่อท้ายนิพจน์ข้างต้นเท่านั้น ตัวอย่างเช่น ถ้าเราต้องการอ้างอิงชื่อของนิสิตคนนี้ เราจะใช้นิพจน์

```
student[5].name
```

ตัวอย่าง 4.1: โปรแกรมด้านล่างรับเวกเตอร์เข้ามา 3 เวกเตอร์ แล้วทำการเปรียบเทียบเวกเตอร์ที่มีขนาดใหญ่ที่สุด

ดังตัวอย่างผลลัพธ์

```
Enter vector 1
X element: 1
Y element: 0
Z element: 10
Enter vector 2
X element: 9
Y element: 9
Z element: 0
Enter vector 3
X element: -9
Y element: 8
Z element: 9
The largest vector is (-9.00,8.00,9.00)
```

โปรแกรมอาศัย function *ReadVector* และ *PrintVector* จากแบบฝึกหัด 3.1 และ function *VectorSize*

จากตัวอย่าง 3.2

```
#include<stdio.h>
#include<math.h>
struct Vector{
    float x,y,z;
};
struct Vector ReadVector(){
    struct Vector v;
    printf("X element: ");
    scanf("%f",&v.x);
    printf("Y element: ");
    scanf("%f",&v.y);
    printf("Z element: ");
    scanf("%f",&v.z);
    return v;
}
void PrintVector(struct Vector v){
    printf("(%.2f,%.2f,%.2f)",v.x,v.y,v.z);
}
float VectorSize(struct Vector v){
    return sqrt(pow(v.x,2) + pow(v.y,2) + pow(v.z,2));
}
```

ชื่อ-นามสกุล.....รหัสนิสิต.....หมู่.....

5. ผีโปรแกรมจากโจทย์

โจทย์ 5.1: จงเขียนโปรแกรมจัดการระเบียบนิสิตจำนวน 5 คน ที่สามารถค้นหาข้อมูลนิสิตตามรหัสประจำตัวได้ โดยข้อมูลของนิสิตแต่ละคนประกอบด้วย

- รหัสประจำตัว เป็นตัวเลข 8 หลัก
- ชื่อ-นามสกุล
- เกรดเฉลี่ยสะสม

โปรแกรมจะทำการอ่านข้อมูลนิสิตเข้ามาก่อน แล้วเข้าสู่โหมดการค้นหาโดยระบุรหัสประจำตัว และจบโปรแกรมเมื่อผู้ใช้ป้อนรหัสที่ต้องการค้นหาเป็น -1 ดังตัวอย่าง

```
Enter student #1's information
ID: 48050613
Name: Somchai Jaikra
GPA: 2.25
```

```
Enter student #2's information
ID: 48006578
Name: Johny Smith
GPA: 3.36
```

```
Enter student #3's information
ID: 48050944
Name: Somying Narak
GPA: 3.98
```

ชื่อ-นามสกุล.....รหัสנית.....หมู่.....

จากนั้นคัดลอกโปรแกรมลงในช่องว่าง