

# ตัวแปรแบบโครงสร้างและยูเนียน

1. ตัวแปรแบบโครงสร้าง
2. การเข้าถึงตัวแปรโครงสร้าง
3. โครงสร้างแบบซ้อน
4. ยูเนียน

วันที่ 14 ตุลาคม 2558


## ตัวแปรแบบโครงสร้าง

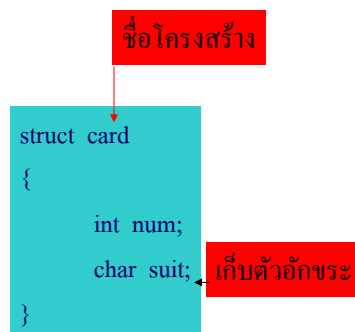
ตัวแปรประเภทอาร์เรย์ เป็นตัวแปรที่สามารถเก็บข้อมูลหลายๆ ค่าไว้ในกลุ่มเดียวกัน แต่ตัวแปรแต่ละตัวจะต้องเป็นข้อมูลชนิดเดียวกัน สำหรับตัวแปรแบบโครงสร้างจะเป็นกลุ่มข้อมูลเช่นกันแต่เป็นข้อมูลต่างประเภทกันได้ ทำให้สามารถเก็บข้อมูลที่มีความสัมพันธ์กันภายใต้ชื่อตัวแปรเดียวกันได้

### การประกาศตัวแปรโครงสร้าง

```
struct [ชื่อโครงสร้าง]
{
    ประเภทของข้อมูล ชื่อตัวแปร1;
    ประเภทของข้อมูล ชื่อตัวแปร2;
    .....
}ชื่อตัวแปร โครงสร้าง;
```

## ตัวอย่างเช่น

- การเก็บข้อมูลโครงสร้างของไฟ  มีตัวแปรที่เป็นตัวอักษรและตัวเลข
- การเก็บข้อมูลรายการหนังสือของห้องสมุด  
มีตัวแปรที่เป็นตัวเลข, ตัวแปรที่เป็นตัวสตริง
- การเก็บข้อมูลของบุคคล  
มีตัวแปรที่เป็นชื่อ, ตัวแปรเก็บเงินเดือน, ตัวแปรเก็บอายุ, ตัวแปรเก็บเพศ



## ตัวอย่างการประกาศตัวแปรโครงสร้างของไฟ

```
struct card
{
    int num;
    char suit;
};
```

การประกาศตัวแปร โครงสร้างทำได้ตามรูปแบบดังนี้

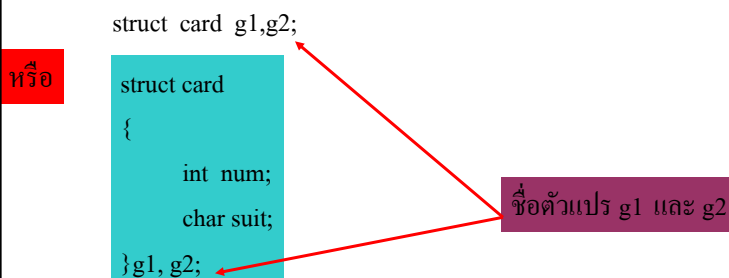
```
struct ชื่อโครงสร้าง รายชื่อตัวแปรโครงสร้าง;
```

ตัวอย่างเช่นถ้าต้องการประกาศตัวแปรชื่อ g1, g2 ให้เป็นตัวแปรประเภทโครงสร้างของไฟทำได้ดังนี้

หรือ

```
struct card g1,g2;

struct card
{
    int num;
    char suit;
} g1, g2;
```



### ตัวอย่างการอ้างถึงสมาชิกในตัวแปรโครงสร้างของไฟ

รูปแบบการอ้างถึงสมาชิกแต่ละตัวทำได้ตามรูปแบบดังนี้

ชื่อตัวแปรโครงสร้าง.ชื่อสมาชิก

↑  
เครื่องหมายจุด

ตัวอย่างเช่นถ้าต้องการกำหนดค่าให้สมาชิกของ g1, g2 ทำได้ดังนี้

```
g1.num = 9;
g1.suit = 'h';
g2.num = 10;
g2.suit = 'd';
```

### ตัวอย่างการประกาศตัวแปรโครงสร้างสำหรับเก็บข้อมูลของบุคคล

```
struct employee
{
    char name[40];
    char address[50];
    long emp_id;
    float salary;
};
```

struct employee emp1; ← ประกาศตัวแปร โครงสร้างชื่อ emp1

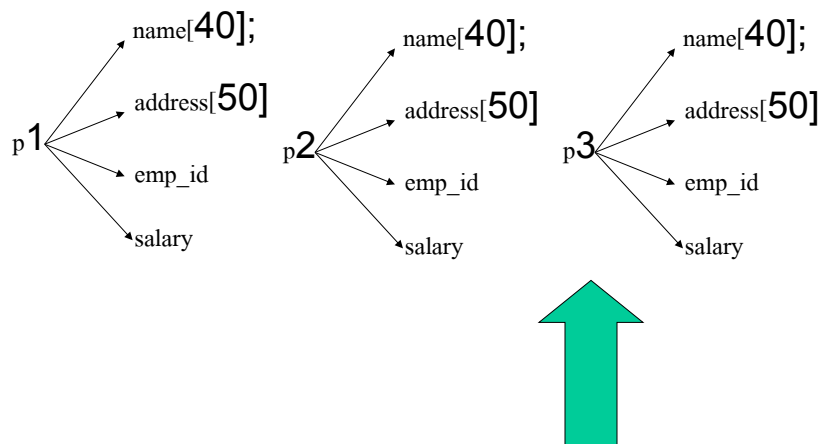
การอ้างถึงสมาชิกของตัวแปร โครงสร้าง emp1 ทำได้ดังนี้

emp1.name , emp1.address, emp1.emp\_id และ emp1.salary

ถ้าหากมีสมาชิกหลายคนอาจเขียนเป็น

struct employee p1,p2,p3;

### ตัวอย่างการประกาศตัวแปร โครงสร้างสำหรับเก็บข้อมูลของบุคคล



ถ้าหากมีสมาชิกหลายคนอาจเขียนเป็น

```
struct employee p1,p2,p3;
```

### ตัวอย่างการอ้างถึงสมาชิกของตัวแปร

```
main()
{
    struct date
    {
        int day;
        char mon_name[4];
        int year;
    };
    struct date d;
```

```
    d.day = 4;
    d.year = 2004;
    d.mon_name[0] = 'A';
    d.mon_name[1] = 'p';
    d.mon_name[2] = 'r';
    d.mon_name[3] = '\0';
    printf("%d %s %d\n",d.day
        ,d.mon_name, d.year);
}
```

ถ้าหากต้องการเปรียบเทียบชื่อเดือนทำได้ดังนี้

```
if(strcmp(d.mon_name,"Aug")== 0)
```

### ตัวอย่างตัวแปรโครงสร้างเก็บรายการหนังสือ

```
struct catalog
{
    char name[30]; /*ชื่อผู้แต่ง*/
    char title[30]; /*ชื่อหนังสือ*/
    char pub[20]; /*ชื่อสำนักพิมพ์*/
    unsigned date; /*วันที่พิมพ์*/
    unsigned char ed; /*ครั้งที่พิมพ์*/
};

struct catalog cat[100];
```

ประกาศตัวแปรเก็บหนังสือไม่เกิน 100 เล่ม  
เล่มแรกคือ cat[0];

ถ้าต้องการใส่ครั้งที่พิมพ์ให้กับ  
หนังสือเล่มที่ 34 ทำดังนี้

↓  
**cat[33].ed = 2;**

ถ้าต้องการพิมพ์ชื่อหนังสือเล่มที่ 20  
ทำได้ดังนี้

↓  
**printf("%s".cat[19].title);**

### การโอนย้ายค่าให้ตัวแปรโครงสร้าง

ตัวแปรโครงสร้างหนึ่งตัวสามารถมองเป็นตัวแปรใด ๆ ได้ ถ้าหากมีการประกาศตัวแปร  
โครงสร้างสองตัว และมีการโอนย้ายข้อมูลระหว่างตัวแปรทั้งสอง สมาชิกทุกตัวจะมีค่า  
เท่ากันหมด

```
struct s_type{
    int a;
    float f;
}var1,var2;

var1.a = 10;
var1.f = 100.23;
var2 = var1; /*ใส่ข้อมูลในตัวแปร var1 ให้ตัวแปร var2*/
```

หลังจากทำคำสั่งนี้ข้อมูลในตัวแปรโครงสร้างจะเหมือนกันทุกประการ

ตัวอย่างการส่งค่าโครงสร้างเข้าไปในฟังก์ชัน

```
#include <stdio.h>
#include <conio.h>
struct s_type{
    int i;
    double d;
}var1;
void f(struct s_type temp);
main()
{
    clrscr();
    var1.i = 99;
    var1.d = 98.6;
    f(var1);
}
```

สร้างฟังก์ชัน และมีตัวแปรชื่อ temp เป็นโครงสร้าง

```
void f(struct s_type temp)
{
    printf("%d %f\n",temp.i,temp.d);
}
```

ส่งค่าตัวแปร โครงสร้างให้กับฟังก์ชัน

## โครงสร้างแบบซ้อน

ดูในหนังสือหน้า 150 ถึง 153

## ยูเนียน (unions)

ตัวแปรแบบยูเนียนเป็นตัวแปรที่ใช้หน่วยความจำจำนวนหนึ่ง ที่สามารถมีสมาชิกได้หลายตัว โดยที่สมาชิกแต่ละตัวสามารถใช้หน่วยความจำร่วมกันได้ ถ้าหากสมาชิกตัวหนึ่งเปลี่ยนแปลง จะทำให้สมาชิกตัวอื่น ๆ เปลี่ยนแปลงด้วย

### การประกาศตัวแปร

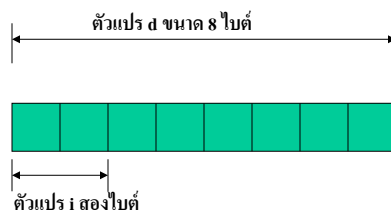
```
union ชื่อประเภทข้อมูล
{
    ชนิดของตัวแปร ชื่อตัวแปร;
    ชนิดของตัวแปร ชื่อตัวแปร;
    .....
}ชื่อตัวแปร,....;
```

## ตัวอย่าง

```
union u_type
```

```
{
    int i;      /* 2 ไบต์ */
    char c[2]; /* 2 ไบต์ */
    double d;   /* 8 ไบต์ */
}sample;
```

ใช้หน่วยความจำทั้งหมด 8 ไบต์  
แต่ถ้าประกาศแบบโครงสร้างจะ  
ใช้หน่วยความจำทั้งหมด 12 ไบต์



ถ้าหากตัวแปร i มีการเปลี่ยนแปลงจะทำให้ตัวแปร d เปลี่ยนไปด้วย