

## Computer Programming II

### การเขียนโปรแกรมคอมพิวเตอร์2

#### LECTURE#9 กลุ่มข้อมูลชนิดโครงสร้าง(Structure)

อ.สถิตย์ ประสมพันธ์

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ

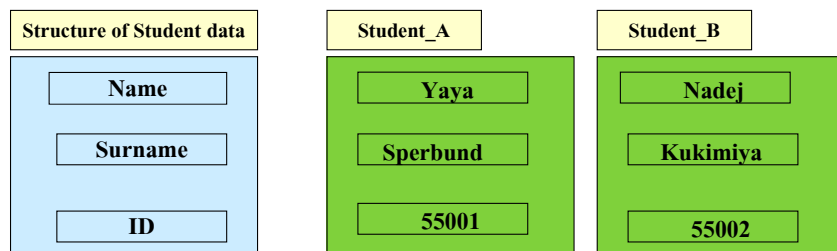
KMUTNB

## กลุ่มข้อมูลชนิดโครงสร้าง(Structure)

- สตริคเจอร์ หรือกลุ่มข้อมูลชนิดโครงสร้าง (Structure / struct )
  - เป็นการประกาศหน่วยของข้อมูลใหม่ที่เกิดจากการรวมกลุ่มของข้อมูล เป็นโครงสร้าง โดยข้อมูลซึ่งเป็นสมาชิกของโครงสร้างใหม่ อาจมีหลายตัว และเป็นชนิดเดียวกันหรือต่างชนิดกันก็ได้ เช่น มีสมาชิกเป็นจำนวนเต็ม ทศนิยม และอักขระ ได้
  - สตริคเจอร์ อาจมีสมาชิกที่เป็นอาร์เรย์ หรือแม้แต่สตริคเจอร์ด้วยก็ได้
- ข้อมูลพื้นฐาน (simple data)
  - int, unsigned int, char, float, double, long เป็นต้น
- ข้อมูลซับซ้อน (complex data)
  - array, structure & union

## กลุ่มข้อมูลชนิดโครงสร้าง(Structure)

- ประโยชน์สตริคเจอร์ หรือกลุ่มข้อมูลชนิดโครงสร้าง (Structure)
  - เพื่อกำหนดหน่วยข้อมูลใหม่ ให้เหมาะสมกับข้อมูลที่ต้องการเก็บ เช่น การกำหนดหน่วยข้อมูลนักเรียน ที่ประกอบด้วยสมาชิกเป็น ชื่อและนามสกุลที่เป็น string (char [ ]) กับรหัสนักศึกษาที่เป็นตัวเลข(int) ซึ่งเราสามารถกำหนดให้ตัวแปรนักเรียน A กับ B มีโครงสร้างแบบหน่วยข้อมูลที่สร้างขึ้นนี้ได้ดังรูป



## การนิยามและประกาศกลุ่มข้อมูลชนิดโครงสร้าง

- การประกาศตัวแปรของกลุ่มข้อมูลชนิดโครงสร้าง structure
  - 1. Structure definition
    - การนิยามกลุ่มข้อมูลที่สร้างใหม่ ว่ามีสมาชิกอะไรบ้าง เป็นชนิดใด
    - ข้อมูลย่อยในสตริคเจอร์เรียกว่า ฟิลด์ (Field)
  - 2. Structure declaration
    - ประกาศตัวแปรสำหรับกลุ่มข้อมูลที่สร้างขึ้นมา

## การนิยามและประกาศกลุ่มข้อมูลชนิดโครงสร้าง

### 1. การนิยามกลุ่มข้อมูลที่สร้างใหม่ว่ามีสมาชิกข้อมูลชนิดใดบ้าง

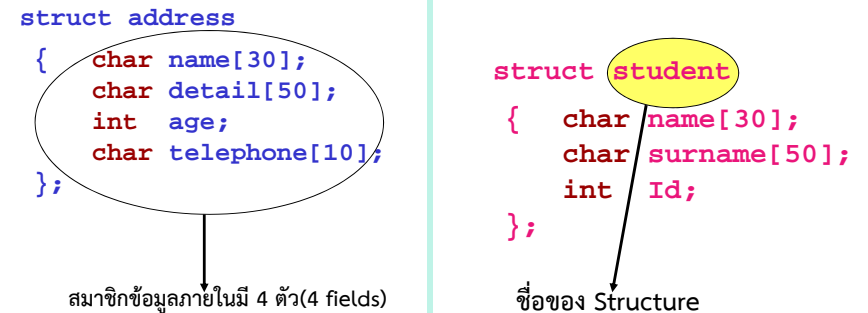
รูปแบบ

**struct** ชื่อของสตรัคเจอร์

```
{   ชนิดของตัวแปร ชื่อตัวแปร[, ชื่อตัวแปร, ...];  
    ชนิดของตัวแปร ชื่อตัวแปร[, ชื่อตัวแปร, ...];  
    .....  
    ชนิดของตัวแปร ชื่อตัวแปร[, ชื่อตัวแปร, ...];  
}
```

## การนิยามและประกาศกลุ่มข้อมูลชนิดโครงสร้าง (2)

- ตัวอย่างการระบุกลุ่มข้อมูลชนิดโครงสร้างที่สร้างขึ้นใหม่



## การนิยามและประกาศกลุ่มข้อมูลชนิดโครงสร้าง (3)

- จงนิยามสตรัคเจอร์ชื่อว่า date เพื่อใช้ในการเก็บข้อมูลของวันที่โดยประกอบด้วยสมาชิก 3 ตัวชื่อว่า day, month และ year ซึ่งสมาชิกทั้งสามเป็นจำนวนเต็ม

```
struct date  
{  
    int day;  
    int month;  
    int year;  
};
```

หรือ

```
struct date {  
    int day,month,year;  
};
```

## การนิยามและประกาศกลุ่มข้อมูลชนิดโครงสร้าง (4)

### 2. ประกาศตัวแปรสำหรับกลุ่มข้อมูลที่สร้างขึ้นมา มีวิธีการประกาศได้ 2 ลักษณะคือ

- 1. การประกาศโดยตรง (ภายหลังจากมีการนิยามสตรัคเจอร์แล้ว) โดยมีรูปแบบ

```
struct ชื่อแบบของสตรัคเจอร์ ชื่อตัวแปร[, ชื่อตัวแปร, ...];
```

เช่น `struct student stdA, stdB;`

- 2. การประกาศตัวแปรพร้อมการสร้างกลุ่มข้อมูลโดยการระบุต่อท้ายก่อนเครื่องหมาย ;

## การนิยามและประกาศกลุ่มข้อมูลชนิดโครงสร้าง (5)

2.1 การประกาศโดยตรง โดยมีรูปแบบ

struct ชื่อแบบของสตรัคเจอร์ ชื่อตัวแปร[, ชื่อตัวแปร, ...];

ตัวอย่าง

struct address input1, input2; ✓

address input1, input2; ✗

## การนิยามและประกาศกลุ่มข้อมูลชนิดโครงสร้าง (6)

2.2 การประกาศตัวแปรพร้อมการสร้างกลุ่มข้อมูลโดยการระบุต่อท้ายก่อนเครื่องหมาย ;

ตัวอย่าง

```
struct address
{
    char name[30];
    char detail[50];
    int age;
    char telephone[10];
} input1, input2 ;
```

หลังจากนี้อาจมีการประกาศตัวแปรเพิ่มได้ เช่น

```
struct address input3, input4 ;
```

## การเข้าถึงสมาชิกในสตรัคเจอร์

- สำหรับ Array เราใช้เลขดัชนีหรือ index(subscript) ในการอ้างถึงสมาชิกแต่ละตัวใน array (เช่น a[5], b[1][0] เป็นต้น) แต่รูปแบบการเข้าถึงข้อมูลสมาชิกของ struct ทำได้โดยใช้โอเปอเรเตอร์จุด (.) แล้วตามด้วยชื่อสมาชิก

รูปแบบ:

ชื่อตัวแปรสตรัคเจอร์.ชื่อสมาชิก

structure-variable.field

## การเข้าถึงสมาชิกในสตรัคเจอร์ (2)

จากรูปแบบ:

ชื่อตัวแปรสตรัคเจอร์.ชื่อสมาชิก

- ตัวอย่าง

```
struct complex {
    double real;
    double image;
} num0 ;

struct complex num1;
```

การกำหนดค่า เช่น

```
num1.real = 1;
num1.image = 2.5;
```

```
Num0.real = num1.image;
num0.image = num1.real;
```

## ตัวอย่างการเขียนโค้ดภาษาซีเพื่อจัดการกับตัวเลขเชิงซ้อน

### 1. นิยามกลุ่มข้อมูลชนิดเชิงซ้อน

```
struct complex
{ double real;
  double image;
};
```

### 2. ประกาศตัวแปร

```
struct complex a,b,c;
```

### 3. ตัวอย่างการใช้งานตัวแปร

```
c.real=a.real+b.real;
c.imag=a.imag+b.image;
```

### Structure Assignment

การกำหนดค่าด้วยสตรัคเจอร์

a = b; มีความหมายเช่นเดียวกับ

a.real=b.real; a.imag=b.image;

คือ copy ทุกข้อมูลจาก b ให้กับ a

แต่ไม่มีการเปรียบเทียบโดยตรงระหว่างสตรัคเจอร์

เช่น a == b หรือ a > b หรือ a < b ใช้ไม่ได้

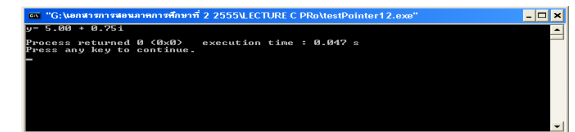
## ตัวอย่างการเขียนโค้ดภาษาซีเพื่อจัดการกับตัวเลขเชิงซ้อน (2)

```
#include <stdio.h>
int main()
{
    struct complex {
        double real;
        double image;
    } x,y;
    x.real = 4;
    x.image = 0.5;
    y.real = x.real + 1;
    y.image = x.image + 0.25;
    printf("y= %.2f + %.2fi\n",y.real,y.image);
    return 0;
}
```

	x	
real		4
image		0.5

	y	
real		5
image		0.75



## การกำหนดค่ากับสมาชิก array แบบ char

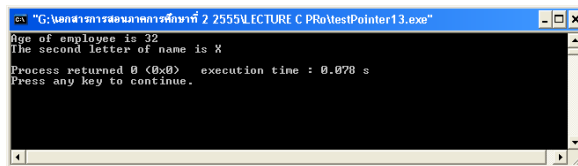
ตัวอย่าง การสร้าง struct ของพนักงาน

```
#include <stdio.h>
int main()
{
    struct employ {
        char name[25];
        int age;
        int pay;
    } employee;

    employee.age=32;
    employee.name[2] = 'X';

    printf("Age of employee is %d\n",employee.age);
    printf("The second letter of name is %c\n", employee.name[2]);

    return 0;
}
```

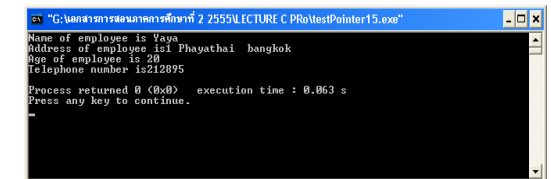


## การกำหนดค่าstring ให้ struct

```
#include <stdio.h>
int main()
{
    struct address{
        char name[30];
        char detail[50];
        int age;
        char telephone[10];
    };
    struct address input;
    strcpy(input.name, "Yaya");
    strcpy(input.detail, "1 Phayathai bangkok");
    input.age=20;
    strcpy(input.telephone, "212895");

    printf("Name of employee is %s\n",input.name);
    printf("Address of employee is %s\n", input.detail);
    printf("Age of employee is %d\n",input.age);
    printf("Telephone number is %s\n", input.telephone);

    return 0;
}
```



## การคัดลอกค่าของตัวแปรสตรัคเจอร์

- การคัดลอก (Copy) ค่าของตัวแปรสตรัคเจอร์ชนิดเดียวกัน สามารถทำได้โดยใช้เครื่องหมาย " = " เช่น

```
struct date {  
    int day, month, year;  
} d1, d2;
```

- แทนที่จะกำหนดค่าสมาชิกทีละตัว

```
d2.day = d1.day;  
d2.month = d1.month;  
d2.year = d1.year;
```

- สามารถกำหนดค่าเป็นตัวแปรสตรัคเจอร์ได้เลย

```
d2 = d1;
```

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

```
int main()  
{
```

```
    struct date {  
        int day, month, year;  
    } ;
```

```
    struct date d1={12,01,2013};  
    struct date d2;
```

```
    d2=d1;
```

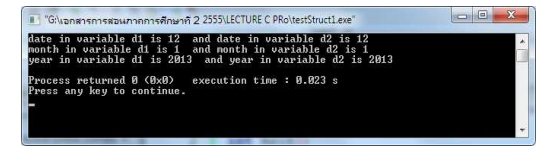
```
    printf("date in variable d1 is %d and date in variable d2 is %d \n",  
           d1.day,d2.day );
```

```
    printf("month in variable d1 is %d and month in variable d2 is %d \n",  
           d1.month,d2.month );
```

```
    printf("year in variable d1 is %d and year in variable d2 is %d \n",  
           d1.year,d2.year );
```

```
    return 0;
```

```
}
```



## สตรัคเจอร์ที่มีสมาชิกเป็นสตรัคเจอร์

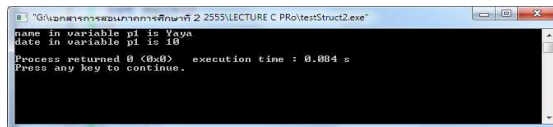
```
#include <stdio.h>  
int main()  
{
```

```
    struct date  
    {  
        int day;  
        int month;  
        int year;  
    };  
    struct person  
    {  
        char name[30];  
        struct date birthday;  
    };
```

```
    struct person p1;
```

```
    strcpy(p1.name,"Yaya");  
    p1.birthday.day = 10;  
    p1.birthday.month = 3;  
    p1.birthday.year = 1992;  
    printf("name in variable p1 is %s\n",p1.name );  
    printf("date in variable p1 is %d\n",p1.birthday.day );  
    return 0;
```

```
}
```



## การกำหนดค่าเริ่มต้นให้กับตัวแปรสตรัคเจอร์

- ใช้เครื่องหมาย { } ครอบค่าเริ่มต้นทั้งหมด และใช้เครื่องหมาย , คั่นระหว่างค่าของสมาชิกแต่ละตัว ตัวอย่างเช่น

```
struct date
```

```
{  
    int day;  
    int month;  
    int year;  
} ;
```

```
struct date d1 = {10,5,1992};
```

d1	
day	10
month	5
year	1992

## การกำหนดค่าเริ่มต้นให้กับตัวแปรสตรัคเจอร์

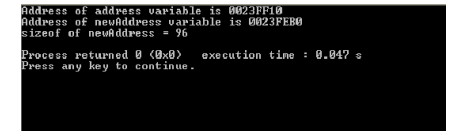
```
struct subject
{
    char name[20];
    int credit;
    char grade;
} s1 = {"Physics I",3,'A'};
```

s1	
name	Physics I
credit	3
grade	A

## การกำหนดค่าเริ่มต้นให้กับตัวแปรสตรัคเจอร์

ตัวอย่าง printf กับ ตัวแปร struct ทั้งหมด

```
#include <stdio.h>
int main()
{
    struct {
        char name[30];
        char detail[50];
        int age;
        char telephone[10];
    } address, newAddress = {"newUser", "1 Samsennai Phayathai
Bangkok", 20, "212895"};
    /* Display address of data groups */
    printf("Address of address variable is %p \n",&address);
    printf("Address of newAddress variable is %p \n",&newAddress);
    printf("sizeof of newAddress = %d\n",sizeof newAddress);
    return 0;
}
```



Address of address variable is 0022FF10  
Address of newAddress variable is 0023FE00  
sizeof of newAddress = 96  
Process returned 0 (0x0) execution time : 0.047 s  
Press any key to continue.

## การเปรียบเทียบค่าของตัวแปรสตรัคเจอร์

- ในการเปรียบเทียบค่าของตัวแปรสตรัคเจอร์นั้น ให้เปรียบเทียบค่าของสมาชิกแต่ละตัว
  - จะนำตัวแปรสตรัคเจอร์มาเปรียบเทียบกับกันโดยตรงไม่ได้  
เช่น หากมีการตัวแปรชนิด struct date ชื่อว่า d1 และ d2
- ```
struct date {
    int day,month,year
} d1,d2;
```
- จะนำ d1 และ d2 มาเปรียบเทียบกับกันโดยตรงไม่ได้

## การกำหนดชนิดตัวแปรใหม่ (Type Definition)

- ในภาษาซี สามารถกำหนดชนิดตัวแปรขึ้นมาใหม่ได้ โดยใช้คำสั่ง  
**typedef (type definition)**
- รูปแบบ: **typedef** ชนิดตัวแปรที่มีอยู่แล้ว ชนิดตัวแปรใหม่;  
เช่น  

```
typedef int my_int;
```
- รูปแบบการใช้ typedef ร่วมกับการนิยามสตรัคเจอร์:  

```
typedef struct
{
    ชนิดตัวแปร ชื่อตัวแปรที่ 1;
    ชนิดตัวแปร ชื่อตัวแปรที่ 2;
    ...
    ชนิดตัวแปร ชื่อตัวแปรที่ n;
} ชนิดตัวแปรใหม่;
```

## การกำหนดชนิดตัวแปรใหม่ (Type Definition)

- ตัวอย่างเช่น

```
typedef struct {  
    int day, month, year;  
} date;
```

- ในการประกาศตัวแปรก็สามารถใช้ชนิดตัวแปรใหม่ได้เลย เช่น  
`date d1, d2;`

ข้อสังเกต เมื่อกำหนดตัวแปรหลังจากการกำหนดด้วย typedef แล้ว  
ไม่ต้องมีคำว่า struct นำหน้าชนิดข้อมูลอีก

## ตัวอย่าง

จงกำหนดชนิดตัวแปรใหม่ชื่อ Student ซึ่งมีสมาชิก 2 ตัวคือ name  
ใช้สำหรับเก็บชื่อซึ่งมีความยาวไม่เกิน 30 ตัวอักษร และสมาชิกตัวที่  
สองชื่อ faculty ใช้สำหรับเก็บชื่อคณะซึ่งมีความยาวไม่เกิน 15  
ตัวอักษร

```
typedef struct  
{  
    char name[31];  
    char faculty[16];  
} Student;
```

## ตัวอย่าง การใช้ typedef กับ struct

```
struct info  
{  
    char firstName[20];  
    char lastName[20];  
    int age;  
};
```

`struct info i1, i2;` ✓

`info j;` ✗

```
typedef struct  
{  
    char firstName[20];  
    char lastName[20];  
    int age;  
} Info;
```

`Info j;` ✓

`struct info k;` ✗

```
typedef struct info infoType;  
  
infoType i3, i4;
```

## ตัวอย่าง typedef ที่มีสมาชิกเป็น typedef

```
typedef struct {  
    char firstName[20];  
    char lastName[20];  
    int age;  
} InfoT;  
  
typedef struct {  
    InfoT info;  
    double salary;  
} EmployeeT;  
  
EmployeeT e1;  
e1.info.age = 21;
```

## การผ่านสตรัคเจอร์ให้กับฟังก์ชัน

- การผ่านค่าของตัวแปรสตรัคเจอร์ให้กับฟังก์ชันทำได้เหมือนกับตัวแปรชนิดอื่นๆ (int, float, char)
- การแก้ไขของพารามิเตอร์ภายในฟังก์ชัน จะไม่มีผลต่อค่าของตัวแปรสตรัคเจอร์ที่ถูกส่งมาเป็นอาร์กิวเมนต์
- ถ้าในโปรแกรมมีหลายฟังก์ชัน การนิยามสตรัคเจอร์และการกำหนดชนิดตัวแปรใหม่ให้นำมาไว้นอกฟังก์ชัน main

## ตัวอย่างการผ่านสตรัคเจอร์ให้กับฟังก์ชัน

```
#include <stdio.h>
typedef struct {
    double re;
    double im;
} complex;

void display(complex a);

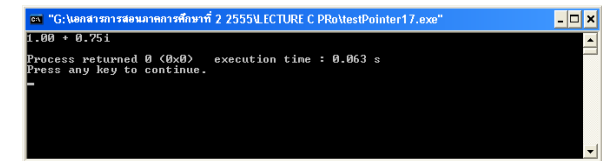
int main() {
    complex x;
    x.re = 1;
    x.im = 0.75;
    display(x);
    return 0;
}

void display(complex a) {
    printf("%.2f + %.2fi\n", a.re, a.im);
}
```

|    |      |
|----|------|
| x  |      |
| re | 1    |
| im | 0.75 |

|    |      |
|----|------|
| a  |      |
| re | 1    |
| im | 0.75 |



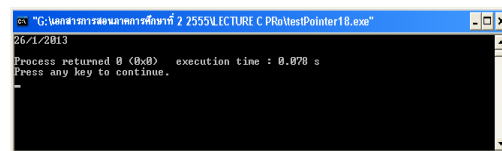
## ตัวอย่าง

```
#include <stdio.h>
typedef struct {
    int day, month, year;
} date;

void edit(date a);

int main() {
    date d1 = {26, 01, 2013};
    edit(d1);
    printf("%d/%d/%d\n", d1.day, d1.month, d1.year);
    return 0;
}

void edit(date a)
{
    a.year = a.year + 10;
}
```



การแก้ไขของพารามิเตอร์ภายในฟังก์ชัน จะไม่มีผลต่อค่าของตัวแปรสตรัคเจอร์ที่ถูกส่งมาเป็นอาร์กิวเมนต์

## ตัวอย่าง (ฟังก์ชันที่มีการส่งค่ากลับเป็นสตรัคเจอร์)

```
#include <stdio.h>
typedef struct {
    double re, im;
} complex;

complex cconst(double a, double b);

int main() {
    double x = 2, y = 0.5;
    complex cnum;
    cnum = cconst(x, y);
    printf("%.2f + %.2fi\n", cnum.re, cnum.im);
    return 0;
}

complex cconst(double a, double b) {
    complex num;
    num.re = a; num.im = b;
    return num;
}
```

|      |     |
|------|-----|
| cnum |     |
| re   | 2   |
| im   | 0.5 |

|     |     |
|-----|-----|
| num |     |
| re  | 2   |
| im  | 0.5 |





## อาร์เรย์ของโครงสร้าง

- การใช้งานโครงสร้างนอกจากใช้ในลักษณะของตัวแปรแล้ว ยังสามารถใช้งานในลักษณะของอาร์เรย์ได้อีกด้วย เช่น การเก็บข้อมูลประวัติของพนักงาน จะมีโครงสร้างที่ใช้เก็บข้อมูลของพนักงานแต่ละคน
- หากใช้ในลักษณะของตัวแปรปกติจะสามารถเก็บข้อมูลของพนักงานได้เพียง 1 คน ซึ่งพนักงานทั้งบริษัทอาจจะมีหลายสิบหรือหลายร้อยคน การเก็บข้อมูลในลักษณะนี้จะใช้อาร์เรย์เข้ามาช่วย เช่น

```
Person staff [STAFFSIZE];
```

---

## การอ้างโดยใช้คำสั่งต่าง ๆ

- staff อ้างถึงอาร์เรย์ของโครงสร้าง
  - staff[i] อ้างถึงสมาชิกที่ i ในอาร์เรย์
  - staff[i].forename อ้างถึงชื่อหน้าของสมาชิกที่ i ของอาร์เรย์
  - staff[i].surname[j] อ้างถึงตัวอักษรตัวที่ j ในนามสกุลของสมาชิกที่ i ของอาร์เรย์
- 

## การเรียกใช้งานสมาชิกบางตัวในอาร์เรย์ของโครงสร้างผ่านฟังก์ชัน

- การใช้ข้อมูลสมาชิกแต่ละตัวจะอ้างถึงโดยการอ้างผ่านระบบดัชนีเหมือนอาร์เรย์ทั่วไป เช่น `print_person ( staff[k] );`
  - รูปแบบฟังก์ชันสามารถกำหนดด้วย  
`void print_person ( Person employee )`
  - หากต้องการเรียกใช้งานฟังก์ชันที่ทำงานกับทั้งอาร์เรย์ เช่น การเรียกใช้งานฟังก์ชันที่ทำการเรียงลำดับอาร์เรย์ตามชื่อหน้า เช่น  
`sort_forename ( staff, STAFFSIZE );`
  - รูปแบบฟังก์ชันสามารถกำหนดด้วย  
`void sort_forename ( Person staff[ ], int size )`
- 

## การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ของโครงสร้าง

- การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ของโครงสร้างสามารถทำได้โดย  

```
Person staff[ ] = { { "Bloggs", "Joe", MALE, 21 },  
                    { "Smith", "John", MALE, 30 },  
                    { "Black", "Mary", FEMALE, 25 } };
```
-

## การใช้อาร์เรย์กับสตรัคเจอร์

- ใช้เก็บข้อมูลที่ต้องใช้ตัวแปรสตรัคเจอร์จำนวนมาก โดยการใช้ตัวแปรอาร์เรย์ของสตรัคเจอร์

```
struct date {  
    int day,month,year;  
};  
struct date date_list[3];
```

|              |  |              |  |              |  |
|--------------|--|--------------|--|--------------|--|
| day          |  | day          |  | day          |  |
| month        |  | month        |  | month        |  |
| year         |  | year         |  | year         |  |
| date_list[0] |  | date_list[1] |  | date_list[2] |  |

## การใช้อาร์เรย์กับสตรัคเจอร์

- รูปแบบการเข้าถึงสมาชิกในแต่ละอีลีเมนต์ในอาร์เรย์ของสตรัคเจอร์  
ชื่อตัวแปรอาร์เรย์ของสตรัคเจอร์[ดัชนี].ชื่อสมาชิก
- ตัวอย่างเช่น (กำหนดค่าให้กับสมาชิกในอีลีเมนต์แรก)

```
date_list[0].day = 12;  
date_list[0].month = 10;  
date_list[0].year = 2002;
```

## ตัวอย่าง (การรับค่าและแสดงค่าอาร์เรย์ของสตรัคเจอร์)

```
#include <stdio.h>  
int main() {  
    typedef struct {  
        char name[30];  
        int age;  
    } student;  
    student stds[3];  
    int i;  
    for(i=0;i<3;i++) {  
        printf("Enter name of student %d: ",i+1);  
        //scanf("%s",stds[i].name);  
        gets(stds[i].name);  
        fflush(stdin);  
        printf("Enter age: ");  
        scanf("%d",&stds[i].age);  
        fflush(stdin);  
    }  
    printf("=====\n");  
    printf("Name      Age\n");  
    printf("=====\n");  
    for(i=0;i<3;i++){  
        printf("%-15s%d\n",stds[i].name, stds[i].age);  
    }  
    return 0;  
}
```

## ตัวอย่าง (ฟังก์ชันที่มีการรับค่าเข้าเป็นอาร์เรย์ของสตรัคเจอร์)

```
#include <stdio.h>  
typedef struct {  
    char name[30];  
    int salary;  
} employee;  
void display (employee emps[]);  
int main()  
{  
    employee emp_list[3];  
    int i;  
    for(i=0;i<3;i++)  
    {  
        printf("Enter name of employee %d: ",i+1);  
        //scanf("%s", emp_list[i].name );  
        gets(emp_list[i].name);  
        fflush(stdin);  
        printf("Enter salary: ");  
        scanf("%d", &emp_list[i].salary );  
        fflush(stdin);  
    }  
    display(emp_list);  
    return 0;  
}
```

```
void display(employee emps[])  
{  
    int i;  
    printf("-----\n");  
    printf("Name      Salary\n");  
    printf("-----\n");  
    for(i=0;i<3;i++)  
        printf("%-15s%d \n",  
            emps[i].name,emps[i].salary);  
}
```

## การประกาศตัวแปรชี้ (pointer) ชี้ไปยัง struct

- กรณีการส่งอาร์กิวเมนต์เป็นตัวแปร struct จะไม่เหมาะกับ struct ที่มีขนาดใหญ่ เนื่องจากทุกครั้งที่ส่งตัวแปร struct จะเป็นการสำเนาตัวแปรตัวใหม่ขึ้นมาในฟังก์ชัน ซึ่งจะทำให้ช้าและเปลืองพื้นที่หน่วยความจำ เราจะใช้พอยน์เตอร์เข้ามาช่วยแก้ปัญหานี้
- โดยส่งแอดเดรสของตัวแปร struct มายังฟังก์ชันซึ่งรับอาร์กิวเมนต์ เป็นพอยน์เตอร์ อาร์กิวเมนต์จะชี้ไปยังแอดเดรสเริ่มต้นของตัวแปร struct จะช่วยให้การทำงานเร็วขึ้นและเปลืองหน่วยความจำน้อยลง

## การประกาศตัวแปรชี้ (pointer) ชี้ไปยัง struct

- `struct point origin, *pp;`  
`pp = &original;`  
`printf ( "origin is (%d, %d)\n", (*pp).x, (*pp).y );`
- จะได้ตัวแปร `pp` ชี้ไปยังข้อมูลแบบโครงสร้างชื่อ `struct point` การเขียน `*pp` จะเป็นการอ้างถึงโครงสร้าง
- การอ้างถึงสมาชิกสามารถทำได้โดยอ้าง `(*pp).x` หรือ `(*pp).y`
- สิ่งที่ต้องระวังคือ `(*pp).x` จะไม่เหมือนกับ `*pp.x` เนื่องจากเครื่องหมาย `.` จะมีลำดับความสำคัญสูงกว่า `*`
- การแปลความหมาย `*pp.x` จะเหมือนกับการอ้าง `*(pp.x)` ซึ่งจะทำให้เกิดความผิดพลาดขึ้น

## การประกาศตัวแปรชี้ (pointer) ชี้ไปยัง struct

- การอ้างถึงสมาชิกอาจเขียนอีกลักษณะหนึ่งโดยใช้เครื่องหมาย `->` สมมติ `p` เป็นพอยน์เตอร์ รูปแบบการใช้เป็นดังนี้
- `p->member-of-structure`
- จะสามารถแปลงประโยคการใช้พอยน์เตอร์อ้างสมาชิกของ struct จากตัวอย่างข้างบนได้ว่า
- `printf ( "origin is (%d, %d)\n", pp->x, pp->y);`

## การประกาศตัวแปรชี้ (pointer) ชี้ไปยัง struct

```
typedef struct {  
    int day;  
    int month;  
    int year;  
} Date;
```

```
Date today;
```

```
Date *ptrdate;
```

แบบที่ 1

การประกาศแบบ  
ข้อมูลโครงสร้าง

การประกาศตัวแปร  
ข้อมูลแบบโครงสร้าง

การประกาศตัวแปร  
pointer ชี้ไปยังโครงสร้าง

## การประกาศตัวแปรชี้ (pointer) ชี้ไปยัง struct

```
struct date {  
    int day;  
    int month;  
    int year;  
} *ptrdate;
```

แบบที่ 2

## การประกาศตัวแปรชี้ (pointer) ชี้ไปยัง struct

```
typedef struct {  
    int day;  
    int month;  
    int year;  
} Date;  
  
typedef Date *PtrDate;  
  
PtrDate ptrdate;
```

แบบที่ 3

การประกาศแบบ  
ข้อมูลโครงสร้าง

การประกาศประเภท  
ตัวแปร pointer ชี้ไปยัง  
โครงสร้าง

การประกาศตัวแปร  
pointer ชี้ไปยัง โครงสร้าง

## การอ้างถึงสมาชิกของโครงสร้างผ่านตัวแปรพอยน์เตอร์

- การอ้างถึงสมาชิกโครงสร้างโดยใช้เครื่องหมาย ->

```
ptrdate->day = 7;
```

```
if ( ptrdate->day == 31 && ptrdate->month == 12 ) .....
```

```
    scanf ( "%d", &ptrdate->year );
```

- การอ้างถึงสมาชิกของโครงสร้างผ่านตัวแปรพอยน์เตอร์

```
(*ptrdate).day = 7;
```

```
if ( (*ptrdate).day == 31 && (*ptrdate).month == 12 ) .....
```

```
    scanf ( "%d", &(*ptrdate).year );
```

## โปรแกรมตัวอย่างการใช้ตัวชี้ (pointer) ชี้ไปยังโครงสร้าง

```
#include <stdio.h>  
struct date {  
    int day;  
    int month;  
    int year;  
};  
  
typedef struct date Date;  
typedef Date *PtrDate;  
  
main ( ) {  
    Date    today;  
    PtrDate ptrdate;  
    ptrdate = &today;  
    ptrdate->day    = 27;  
    ptrdate->month   = 9;  
    ptrdate->year    = 1985;  
    printf ( "Today's date is %2d/%2d/%4d\n", ptrdate->day,  
            ptrdate->month, ptrdate->year );  
}
```