

# บทที่ 6 สตรัคเจอร์

## Structure



คณานักวิชาชีวกรรมคอมพิวเตอร์  
[introcom@coe.psu.ac.th](mailto:introcom@coe.psu.ac.th)

# วัตถุประสงค์

- เพื่อให้นักศึกษารู้จักสตรัคเจอร์ เข้าใจรูปแบบการจัดเก็บ และนำสตรัคเจอร์ไปใช้งานได้
- เพื่อให้นักศึกษาสามารถส่งผ่านค่าของสตรัคเจอร์ไปยังฟังก์ชัน และรับค่าของสตรัคเจอร์จากฟังก์ชันได้

# หัวข้อศึกษา

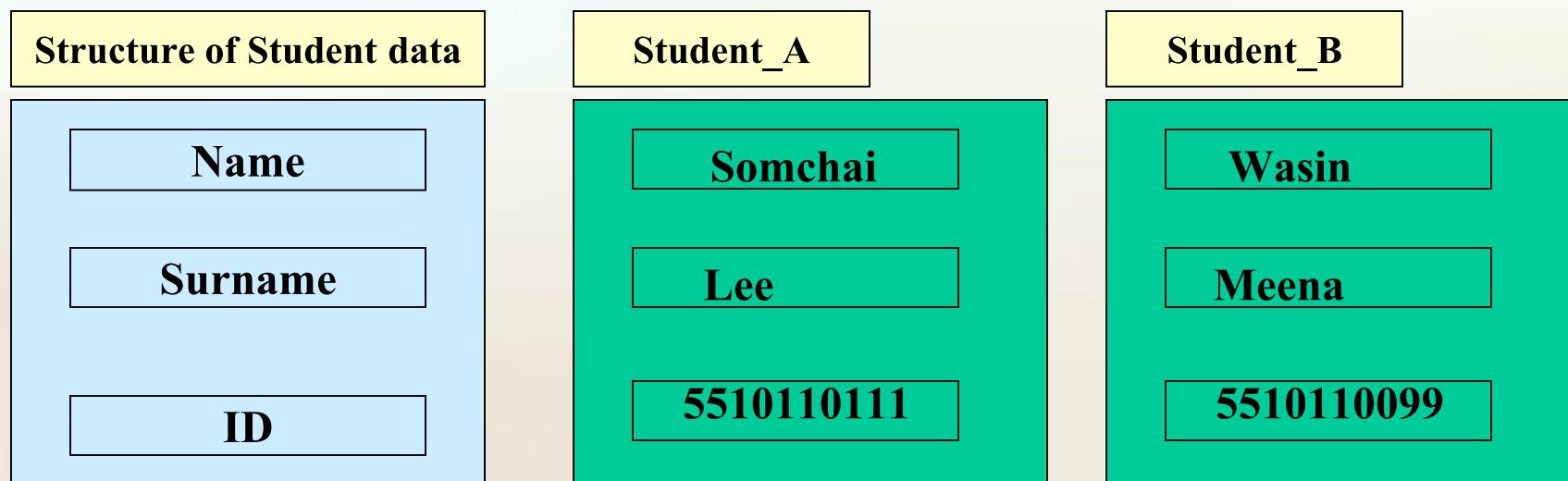
- สรุคเจอร์คืออะไร
- การนิยามสรุคเจอร์และการประกาศตัวแปรสรุคเจอร์
- การเข้าถึงสมาชิกในสรุคเจอร์
- สรุคเจอร์ที่มีสมาชิกเป็นสรุคเจอร์
- การเปรียบเทียบค่าของตัวแปรสรุคเจอร์
- การกำหนดชนิดตัวแปรใหม่
- การผ่านสรุคเจอร์ให้กับฟังก์ชัน
- การใช้อาร์เรย์กับสรุคเจอร์

# สตรัคเจอร์ (Structure) คืออะไร

- การกำหนดโครงสร้างข้อมูลใหม่ โดยการรวมกลุ่มกันของโครงสร้างข้อมูลที่มีอยู่แล้ว เช่น char, int, float, double, array หรือ โครงสร้างอื่นที่มี
- สามารถเข้าถึงข้อมูลของโครงสร้างข้อมูลใหม่ ได้โดยเดียว ก็ได้
- เพื่อกำหนดหน่วยข้อมูลใหม่ ให้เหมาะสมกับข้อมูลที่ต้องการ เช่น ชื่อ นามสกุล ที่เป็น string (char [ ]) กับรหัสนักศึกษาที่เป็นตัวเลข (int)

# ตัวอย่างสตรคเจอร์

- กำหนดโครงสร้างข้อมูลนักเรียน ประกอบด้วย ชื่อและนามสกุลที่เป็น string (char [ ]) กับรหัสนักศึกษาที่เป็นตัวเลข (int)
- ตัวแปรนักเรียน A กับ B มีโครงสร้างแบบหน่วยข้อมูลที่สร้างขึ้นดังรูป



# การนิยามและการประกาศสตอร์คเจอร์

การประกาศตัวแปรของข้อมูลชนิดโครงสร้าง struct มี 2 ขั้นตอน

1. **นิยามโครงสร้างข้อมูลใหม่ (struct)**
2. **ประกาศตัวแปรชนิดโครงสร้างข้อมูลใหม่ที่นิยามขึ้นมา**

# การนิยามกลุ่มโครงสร้างข้อมูลที่สร้างใหม่

1. กำหนดว่ามีสมาชิกข้อมูลชนิดใดบ้าง  
รูปแบบ

**struct** ชื่อแบบของสตรัคเจอร์

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

ชนิดของตัวแปร ชื่อตัวแปร [, ชื่อตัวแปร, ...];

};

# ตัวอย่างการนิยามโครงสร้างข้อมูล

รีอของกลุ่มโครงสร้างข้อมูล

```
struct client  
{ char name[30];  
    char address[50];  
    int age;  
    char telephone[10];  
};
```

มีสมาชิกภายใน 4 ตัว  
(4 fields)

รีอของกลุ่มโครงสร้างข้อมูล

```
struct student  
{ char name[30];  
    char surname[50];  
    int id;  
};
```

มีสมาชิกภายใน 3 ตัว  
(3 fields)

## ตัวอย่างการนิยามโครงสร้างข้อมูล (ต่อ)

นิยามสตรัคเจอร์ชื่อว่า date เพื่อใช้ในการเก็บข้อมูลของวันที่ โดยประกอบด้วยสมาชิก 3 ตัวชื่อว่า day, month และ year ซึ่ง สมาชิกทั้งสามเป็นจำนวนเต็ม

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

หรือ

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

## แบบฝึกหัด

จงนิยามสตรัคเจอร์ชื่อว่า person ซึ่งมีสมาชิก 2 ตัวคือ name  
ใช้สำหรับเก็บข้อมูลความที่ยาวไม่เกิน 50 ตัวอักษร และ age ซึ่ง  
เป็นจำนวนเต็ม

```
struct person {  
    char name[51];  
    int age;  
}
```

# ประกาศตัวแปรสำหรับกลุ่มข้อมูลที่สร้างขึ้นมา

มีวิธีการประกาศได้ 2 ลักษณะคือ

1. การประกาศโดยตรง โดยมีรูปแบบ

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

```
struct client input1;  
struct client input2,input3;  
client input1, input2;
```

## ประกาศตัวแปรสำหรับกลุ่มข้อมูลที่สร้างขึ้นมา (ต่อ)

- การประกาศตัวแปรร่วมกับการสร้างกลุ่มข้อมูลโดยการระบุต่อท้ายก่อนเครื่องหมาย ;

```
struct client
```

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

หลังจากนี้อาจมีการประกาศตัวแปรเพิ่มได้ เช่น

```
struct client input3, input4;
```

## การเข้าถึงสมาชิกในสตรัคเจอร์

- สำหรับ Array เราใช้เลขดัชนีหรือ index ในการอ้างอิง สมาชิกแต่ละตัวใน array เช่น `a[5], b[1][0]` เป็นต้น
- แต่รูปแบบการเข้าถึงข้อมูลสมาชิกของ struct ทำได้โดยใช้ออเปอเรเตอร์จุด (.) และตามด้วยชื่อสมาชิก

รูปแบบ:

ชื่อตัวแปรสตรัคเจอร์.ชื่อสมาชิก

# ตัวอย่างที่ 1. การเข้าถึงสมาชิกในสตรัคเจอร์

การนิยามและประกาศตัวแปร  
ของกลุ่มข้อมูลตัวเลขเชิงซ้อน

```
struct complex {  
    double real;  
    double imag;  
} num0 ;  
  
struct complex num1;
```

การกำหนดค่า เช่น

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

```
num0.real = num1.real;  
num0.imag = num1.imag;
```

## ตัวอย่างที่ 2. การเข้าถึงสมาชิกแบบ array ในสตรัคเจอร์

```
struct employee {  
    char name[25];  
    char surname[50];  
    int age;  
    int pay;  
} emp;
```

การกำหนดค่า เช่น  
emp.age = 32;  
strcpy(emp.name, "David");  
strcpy(emp.surname, "Lee");  
emp.name[4] = '\n';

# แบบฝึกหัด จงเขียนผลลัพธ์ของโปรแกรม

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

x	re	4
	im	0.5

y	re	5
	im	0.75

ผลลัพธ์

$y = 5.00 + 0.75i$

## แบบฝึกหัด จงเติมส่วนที่ขาดหายไป

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

int main() {
    struct person {
        char name[30];
        int age;
        char tel[10];
    } p1;
    -- strcpy(p1.name, "Andy Murray");
    -- p1.age = 26;
    -- strcpy(p1.tel, "074212895");
    printf("%s, %d years old.\n", p1.name, p1.age);
    printf("Telephone: %s\n", p1.tel);
    return 0;
}
```

ผลลัพธ์

Andy Murray, 26 years old.  
Telephone: 074212895

# 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;
```

# แบบฝึกหัด จงเขียนผลลัพธ์ส่วนที่เหลือ

```
#include <stdio.h>
#include <string.h>
int main() {
    struct subject {
        char name[30];
        int credit;
    } s1,s2;
    printf("Enter subject name: ");
    gets("%s",s1.name);
    printf("Enter credit: ");
    scanf("%d",&s1.credit);
    s2 = s1;
    s2.credit = s2.credit + 1;
    printf("%s, credit: %d\n",s2.name,s2.credit);
    return 0;
}
```

ผลลัพธ์

```
Enter subject name: Physics I
Enter credit: 3
Physics I, credit: 4
```

The diagram illustrates the state of memory after the program's execution. It shows two identical-looking structures, **s1** and **s2**, each consisting of two fields: **name** and **credit**. Both structures are highlighted with orange boxes.

<b>s1</b>	
<b>name</b>	Physics I
<b>credit</b>	3

<b>s2</b>	
<b>name</b>	Physics I
<b>credit</b>	4

# ສຕຣັກເຈອຣ໌ທີ່ມີສາມາຊີກເປັນສຕຣັກເຈອຣ໌

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

struct person
{
    char name[30];
    struct date birthday;
};
```

```
struct person p1;
strcpy(p1.name,"Somchai")
;
p1.birthday.day = 10;
p1.birthday.month = 5;
p1.birthday.year = 1998;
```

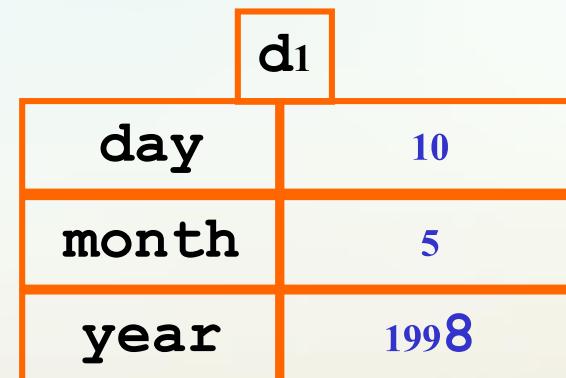
# การกำหนดค่าเริ่มต้นให้กับตัวแปรstructเจอร์

- ใช้เครื่องหมาย { } คู่กับค่าเริ่มต้นทั้งหมด และใช้เครื่องหมาย คั่นระหว่างค่าของสมาชิกแต่ละตัว ตัวอย่างเช่น

```
struct date
```

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

```
struct date d1 = {10,5,1998};
```



d1	
day	10
month	5
year	1998

# การกำหนดค่าเริ่มต้นให้กับตัวแปรสตรัคเจอร์ (ต่อ)

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

s1	
name	Physics I
credit	3
grade	A

## แบบฝึกหัด จงเติมส่วนที่ขาดหายไป

```
#include <stdio.h>
int main() {
    struct date {
        int day,month,year;
    };
    struct student {
        char name[30];
        struct date bday;
    };
    struct student std = {"Warakorn Wannarat", {15,5,1998}};
    printf("%s \n", std.name);
    printf("Birthday: %d/%d/%d\n", std.bday.day,
           std.bday.month, std.bday.year);
    return 0;
}
```

ผลลัพธ์

Warakorn Wannarat  
Birthday: 15/5/1998

# การเปรียบเทียบค่าของตัวแปรสตรัคเจอร์

- ในการเปรียบเทียบค่าของตัวแปรสตรัคเจอร์นั้น ให้เปรียบเทียบค่าของสมาชิกแต่ละตัว
- จะนำตัวแปรสตรัคเจอร์สตรัคเจอร์มาเปรียบเทียบกันโดยตรง ไม่ได้ เช่น หากมีการตัวแปรชนิด struct date ชื่อว่า d1 และ d2

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

- จะนำ d1 และ d2 มาเปรียบเทียบกันโดยตรงไม่ได้  
~~if(d1 == d2)~~  
printf("d1 is the same date as d2");

## ตัวอย่างที่ 3.

```
#include <stdio.h>
int main()
{
    struct complex{
        double re,im;
    };
    struct complex y,x = {1,0.25};
    printf("Enter complex number: ");
    scanf("%lf%lf",&y.re,&y.im);
    if((x.re == y.re) && (x.im == y.im)) // 1.
        printf("x and y is the same complex.\n");
    else // 2.
        printf("x and y is not the same complex.\n");
    return 0;
}
```

จะเติมส่วนที่ขาดหายไปเพื่อใช้ในการ  
เปรียบเทียบว่าจำนวนเชิงซ้อนที่รับจาก  
ผู้ใช้ (y) ว่ามีค่าเท่ากับ x หรือไม่

## การกำหนดชนิดตัวแปรใหม่ด้วย **typedef**

- 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 (ต่อ)

- ตัวอย่าง

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

- ในการประกาศตัวแปรก็สามารถใช้ชนิดตัวแปรใหม่ได้เลย

```
date d1,d2;
```

ข้อสังเกต การประกาศตัวแปรหลังจากการกำหนดด้วย **typedef** แล้วไม่ต้องมีคำว่า **struct** นำหน้าชนิดข้อมูลอีกต่อไป

## แบบฝึกหัด

จงกำหนดชนิดตัวแปรใหม่ชื่อ **student** ซึ่งมีสมาชิก 2 ตัวคือ **name** ใช้สำหรับเก็บชื่อซึ่งมีความยาวไม่เกิน 30 ตัวอักษร และสมาชิกตัวที่สองชื่อ **faculty** ใช้สำหรับเก็บชื่อคณะซึ่งมีความยาวไม่เกิน 15 ตัวอักษร

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

## ตัวอย่างที่ 4. เปรียบเทียบการใช้ `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;
```

## ตัวอย่างที่ 5. `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;
```

## ตัวอย่างที่ 6. การเปรียบเทียบค่าและการถ่ายโอนค่า

จากตัวอย่างที่ 5 `InfoT i1, i2;`

การให้ค่าสามารถทำได้โดยตรง

`i1 = i2;`

แต่การเปรียบเทียบไม่สามารถทำได้โดยตรง

`i1 == i2;`



ดังนั้นการเปรียบเทียบสามารถกระทำได้โดย

```
if( strcmp(i1.firstName, i2.firstName) == 0 &&
strcmp(i1.lastName, i2.lastName) == 0 &&
i1.age == i2.age)
printf("i1 is the same InfoT as i2");
```

# การผ่านสตรัคเจอร์ให้กับฟังก์ชัน

- การผ่านค่าของตัวแปรสตรัคเจอร์ให้กับฟังก์ชันทำได้เมื่อกับตัวแปรชนิดอื่น ๆ (int, float, char,double)
- การแก้ไขของพารามิเตอร์ภายในฟังก์ชัน จะ ไม่มีผล ต่อค่าของตัวแปรสตรัคเจอร์ที่ถูกส่งมาเป็นอาร์กิวเมนต์
- ถ้าในโปรแกรมมีหลายฟังก์ชัน การนิยามสตรัคเจอร์และการกำหนดชนิดตัวแปรใหม่ (typedef) ให้นำมาไว้นอกฟังก์ชัน main

# แบบฝึกหัด จงเขียนผลลัพธ์ของโปรแกรม

```
#include <stdio.h>
typedef struct {
    double real;
    double imag;
} complex;
void display(complex a);
int main() {
    complex x;
    x.real = 1;
    x.imag = 0.75;
    display(x);
    return 0;
}
void display(complex a) {
    printf("%.2f + %.2fi\n",a.real,a.imag);
}
```

x	
re	1
im	0.75

a	
re	1
im	0.75

ผลลัพธ์  
1.00 + 0.75i

# แบบฝึกหัด จงเขียนผลลัพธ์ของโปรแกรม

```
#include <stdio.h>
typedef struct {
    int day,month,year;
} date;
void edit(date a);
int main() {
    date d1 = {12,9,2013};
    edit(d1);
    printf("%d/%d/%d\n",d1.day,d1.month,d1.year);
    return 0;
}
void edit(date a)
{
    a.year = a.year + 10;
}
```

ผลลัพธ์

12/9/2013

การแก้ไขของพารามิเตอร์ภายใน  
ฟังก์ชัน จะ ไม่มีผล ต่อค่าของตัวแปร  
สตรัคเจอร์ที่ถูกส่งมาเป็นอาร์กิวเมนต์

## ตัวอย่างที่ 7. พึงก์ชันที่มีการส่งค่ากลับเป็นสตรัคเจอร์

```
#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	

ผลลัพธ์
2.00 + 0.50i

num		
re	2	
im	0.5	

## การใช้อาร์เรย์กับสตรัคเจอร์

- ในการเก็บข้อมูลที่ต้องใช้ตัวแปรสตรัคเจอร์จำนวนมาก สามารถแก้ปัญหาได้โดยการใช้ตัวแปรอาร์เรย์ของสตรัคเจอร์

```
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 (ต่อ)

- รูปแบบการเข้าถึงสมาชิกในแต่ละอิลิเมนต์ในอาร์เรย์ของสตรัคเจอร์  
ชื่อตัวแปรอาร์เรย์ของสตรัคเจอร์[ดัชนี]. ชื่อสมาชิก
- ตัวอย่างเช่น (กำหนดค่าให้กับสมาชิกในอิลิเมนต์แรก)  
`date_list[0].day = 10;`  
`date_list[0].month = 9;`  
`date_list[0].year = 2013;`

## ตัวอย่างที่ 8. 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);
        printf("Enter age: ");
        scanf("%d", &stds[i].age);
    }
    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;
}
```

## ผลลัพธ์

Enter name of student 1: **John**

Enter age: **14**

Enter name of student 2: **David**

Enter age: **15**

Enter name of student 3: **Angie**

Enter age: **12**

=====

Name	Age
------	-----

=====

John	14
------	----

David	15
-------	----

Angie	12
-------	----

## ตัวอย่างที่ 9. พัฟ์ชันที่มีการรับค่าเข้าเป็นอาร์เรย์ของสตรคเจอร์

```
#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);
        printf("Enter salary: ");
        scanf("%d", &emp_list[i].salary );
    }
    display(emp_list);
    return 0;
}
```

พัฟ์ชัน display รับค่าเข้าเป็น  
อาร์เรย์ของ employee

```
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 );  
}
```

## ผลลัพธ์

Enter name of employee 1: **Somsak**

Enter salary: **12000**

Enter name of employee 2: **Anan**

Enter salary: **15000**

Enter name of employee 3: **Somsri**

Enter salary: **9000**

---

Name	Salary
------	--------

---

Somsak	12000
--------	-------

Anan	15000
------	-------

Somsri	9000
--------	------

## ขนาดของตัวแปรสตรัคเจอร์

- ขนาดของตัวแปรสตรัคเจอร์มีค่าเท่ากับขนาดของสมาชิกแต่ละตัวรวมกัน เช่น

```
struct person
{   char name[20];
    int age;
};

struct person p1;
```

- ขนาดของตัวแปร  $p1 = 20 + 4 = 24$  ไบต์

# แบบฝึกหัด จงเขียนผลลัพธ์ของโปรแกรม

```
#include <stdio.h>
int main() {
    struct date {
        int day,month,year;
    } d1;

    struct person {
        int age;
        char name[25];
    } p1;
    printf("size of d1 = %d\n",sizeof(d1));
    printf("size of p1 = %d\n",sizeof(p1));
    return 0;
}
```

ผลลัพธ์

size of d1 = 12  
size of p1 = 32

# คำถาม ?

- ทำไมขนาดของตัวแปร p1 จึงได้เท่ากับ 32 แทนที่จะเป็น 29?
- วิธีการจองหน่วยความจำของคอมไพเลอร์
- \*\*\* สำหรับคอมไพเลอร์ GCC ตัวแปรสตรัคเจอร์มีข้อกำหนด  
ว่าการจองหน่วยความจำจะต้องมีขนาดที่ หารด้วย 4 ลงตัว  
เสมอ

## แบบฝึกหัด

จากตัวอย่างที่ 9 จะแก้ไขโปรแกรมให้ผู้ใช้สามารถกำหนด  
จำนวนพนักงานได้ (สมมุติว่าจำนวนพนักงานไม่เกิน 10 คน)

### ตัวอย่างผลลัพธ์ของโปรแกรม

```
Enter number of employees: 2
Enter name of employee 1: Sawat
Enter salary : 20000
Enter name of employee 2: Samrit
Enter salary : 16000
```

---

Name	Salary
Sawat	20000
Samrit	16000

## ตัวอย่างที่ 9. – แก้ไข เพิ่มเติม

```
#include <stdio.h>
typedef struct {
    char name[30];
    int salary;
} employee;
void display (employee emps[],int n);
int main()
{   employee emp_list[10];
    int i, n;
    printf("Enter number of emp: ");
    scanf("%d",&n);
    for (i=0; i<n; i++)
    {   printf("Enter name of employee %d:",i+1);
        scanf("%s", &emp_list[i].name);
        printf("Enter salary: ");
        scanf("%d", &emp_list[i].salary );
    }
    display(emp_list,n);    return 0;    }
```

ฟังก์ชัน display รับค่าเข้าเป็น  
อาร์เรย์ของ employee



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