

# ข้อมูลโครงสร้าง Structure

- สตริงเจอร์ คืออะไร
- การประกาศตัวแปร สตริงเจอร์
- การเข้าถึงข้อมูลสมาชิกใน สตริงเจอร์
- การประยุกต์ใช้อาเรย์ (Array) กับ สตริงเจอร์
- การใช้งานข้อมูลสมาชิกประเภท สตริงเจอร์ ใน สตริงเจอร์
- การใช้งาน สตริงเจอร์ กับ ฟังก์ชัน

## สตรัคเจอร์ (Structure) คืออะไร



- วัตถุต่าง ๆ ในความเป็นจริงไม่สามารถใช้ตัวแปรตัวเดียวในการแสดงผลได้
- วัตถุเหล่านี้ประกอบด้วยตัวแปรหลาย ๆ ตัว ทั้งที่เป็นชนิดเดียวกันหรือต่างชนิดกัน เพื่ออธิบายคุณสมบัติของวัตถุเหล่านั้น

# รู้จัก โครงสร้าง (Structure)

โครงสร้าง เป็นโครงสร้างข้อมูลที่รวบรวมข้อมูลหลายอย่างไว้ด้วยกัน

- ข้อมูลแต่ละอย่างนั้นจะเป็นชนิดเดียวกันหรือต่างกันได้

ความรู้เดิม : อาเรย์มีโครงสร้างข้อมูลเป็นอย่างไร ???

คำตอบ : อาเรย์มีข้อมูลหลายตัวได้แต่ชนิดข้อมูลเป็นแบบเดียวกันหมด

Student Record



ID

Name

Address

Phone no.

E-mail

- **สตรัคเจอร์** เป็นการสร้างชนิดข้อมูลตัวใหม่ขึ้นมา
- **ประโยชน์** คือ การรวบรวมข้อมูลหลายตัวที่มีความสัมพันธ์กันเป็นองค์ประกอบย่อยของข้อมูลมาอยู่ด้วยกัน
  - เช่น ข้อมูลพนักงาน ควรประกอบด้วย ชื่อ นามสกุล ที่อยู่ อายุ หมายเลขโทรศัพท์ เงินเดือน
  - ตัวละครในเกมส์ ควรประกอบด้วย Name Type Location Strength Factor Intelligence Factor Type of Armor

# การประกาศตัวแปร สตรักเจอร์ (Structure)

- โครงสร้างของ สตรักเจอร์ ประกอบด้วย
  1. ชื่อ สตรักเจอร์ (Structure Name)
  2. ฟیلด์ข้อมูล หรือ สมาชิกข้อมูล
    - ชนิดข้อมูล (Data type)
    - ชื่อสมาชิกข้อมูล (Variable Name)
  3. การประกาศตัวแปรชนิด สตรักเจอร์ ด้วย คีย์เวิร์ด (Keyword) ว่า **struct**

## ตัวอย่างการประกาศโครงสร้าง สตริงเจอร์

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

ชื่อ สตริงเจอร์

คำถาม : struct Date จองพื้นที่  
เก็บข้อมูลทั้งหมดเท่าไร .....

สมาชิกข้อมูลใน สตริงเจอร์

- ข้อมูลทั้งสามตัว month day year คือสมาชิกข้อมูลใน สตริงเจอร์
- สตริงเจอร์ มีการจองพื้นที่เก็บข้อมูลสำหรับ สมาชิกข้อมูล แต่ละตัวใน สตริงเจอร์

# การประกาศโครงสร้างของตัวแปร สตักเจอร์

```
struct <name> {  
    member 1;  
    member 2;  
    :  
    member n;  
} <variableName> ;
```

ตัวอย่าง :

```
struct employee {  
    char name[16];  
    char surname[31];  
    char address[151];  
    float salary;  
};
```

- เราได้ชนิดข้อมูลใหม่ขึ้นมา เป็นข้อมูลชนิด **struct** employee (เหมือน int ,float ,char ....)
- เราสามารถใช้ **struct** employee ไปประกาศเป็นชนิดข้อมูลของตัวแปรได้ราวกับว่ามันเป็นชนิดข้อมูลทั่วไป เช่น

```
struct employee emp;
```


# การประกาศสตรัคด้วย typedef

## แบบที่ 1

- เป็นการประกาศชื่อเรียกอีกอย่างให้กับชนิดข้อมูล

- ชนิดข้อมูลอันหนึ่งสามารถมีชื่อเรียกได้หลายชื่อ

เขียนแบบนี้จะทำให้ struct employee มีชื่อเรียกสั้น ๆ ว่า EMPLOYEE



```
typedef struct {  
    char name[16];  
    char surname[31];  
    char address[151];  
    float salary;  
} EMPLOYEE;
```

- เราสามารถใช้คำว่า EMPLOYEE ในการประกาศตัวแปรได้เลย

```
EMPLOYEE emp;
```



# การประกาศสตรัคด้วย typedef

## แบบที่ 2

- วิธีประกาศแบบเดิมทำให้ **struct** มีชื่อว่า EMPLOYEE แบบเดียว เราไม่สามารถใช้ชื่อชนิดข้อมูลว่า **struct** employee ได้
- ถ้าอยากให้มีหลายชื่อ เราสามารถประกาศ typedef ไว้ด้านท้าย เช่น

```
struct employee {  
    char name[16];  
    char surname[31];  
    char address[151];  
    float salary;  
} typedef EMPLOYEE;
```

- วิธีนี้จะทำให้ได้ชื่อชนิดข้อมูลเดียวกันมาสองชื่อคือ **struct** employee และ **EMPLOYEE** มาเฉย ๆ

## การประกาศสตรัคด้วย typedef

แบบที่ 3

```
struct employee {  
    char name[16];  
    char surname[31];  
    char address[151];  
    float salary;  
};  
Typedef struct employee EMPLOYEE;  
EMPLOYEE a,b,c  
//Instead of using struct employee a,b,c
```

# การเข้าถึงข้อมูลใน สตริงเจอร์

- เราใช้เครื่องหมาย . (dot) ในการเข้าถึงข้อมูล
- การอ้างถึงสตริงเจอร์ ต้องอ้างถึง ตัวแปร ไม่ใช่ ชนิดข้อมูล

แบบนี้ได้

```
EMPLOYEE emp;  
emp.salary = 18000;
```

แบบนี้ไม่ได้

```
EMPLOYEE emp;  
EMPLOYEE.salary = 18000;
```

สรุป : การเข้าถึงข้อมูลใน สตริงเจอร์

ชื่อของตัวแปรสตริงเจอร์.ชื่อของตัวแปรที่เป็นสมาชิกในสตริงเจอร์

variable.member

# การรับข้อมูลเข้าไปเก็บไว้ในสตรัคเจอร์

- เวลาที่เราใช้ scanf เราจะต้องใช้มันกับตัวแปรแต่ละตัว
  - เราไม่สามารถ scanf กับสตรัคทั้งก้อนรวดเดียวได้
  - ต้องทำทีละตัว และต้องระบุชนิดข้อมูลใน scanf ให้ถูกต้องด้วย

```
scanf ("%s", emp.name) ;  
scanf ("%s", emp.surname) ;  
scanf ("%f", &emp.salary) ;
```

- กฎเกณฑ์การรับข้อมูลเข้าด้วย scanf เหมือนเดิมทุกประการ คือเรา仍需ส่งที่อยู่ของตัวแปรไปให้
  - ยังต้องใช้ & นำหน้าชื่อตัวแปรทั่วไป เช่น &emp.salary
  - แต่ไม่ต้องใช้ & นำหน้าตัวแปรสตริง เช่น emp.name
  - แยกให้ออกว่าสตริงกับช่องข้อมูลช่องหนึ่งในอาเรย์เป็นคนละอย่างกัน

## การพิมพ์ข้อมูลภายในสตรัคเจอร์

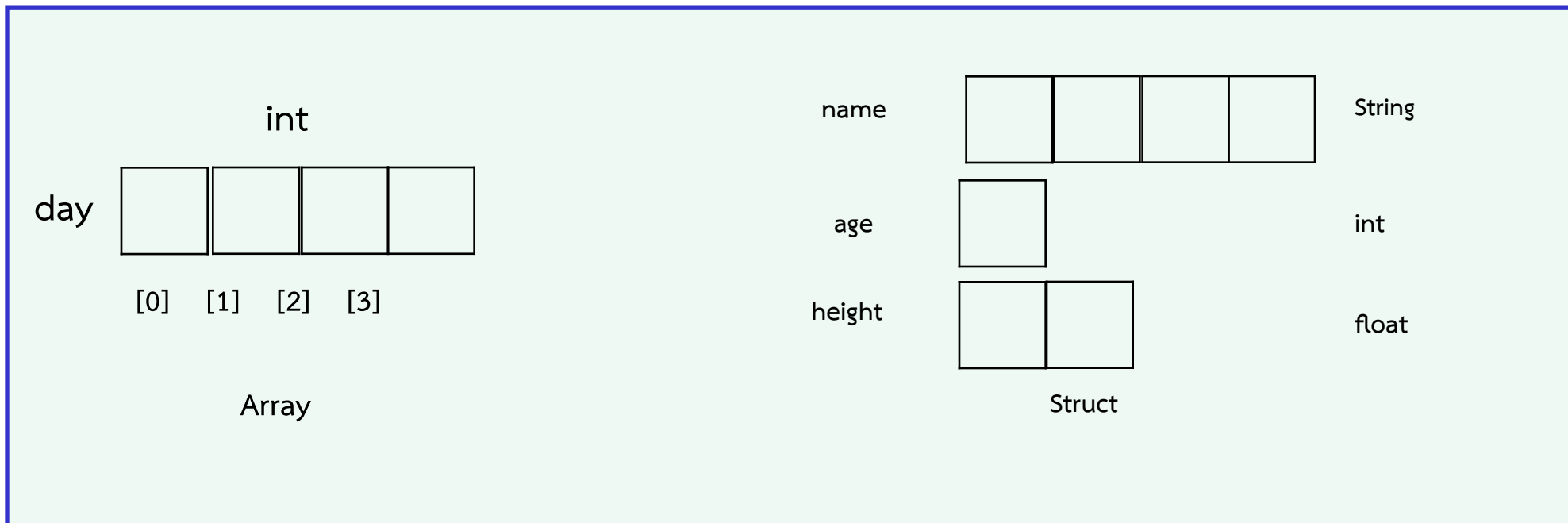
- โดยทั่วไปเราไม่สามารถพิมพ์ข้อมูลทั้งหมดในสตรัคออกมารวดเดียวได้
  - เราต้องพิมพ์ออกมาทีละตัวคล้ายกับตอนทำ scanf()
  - ต้องระบุชนิดข้อมูลให้ตรงกันตามระเบียบ

```
printf("%s %s\n", emp.name, emp.surname);  
printf("%s\n", emp.address);  
printf("%.2f", emp.salary);
```

# การประยุกต์ใช้อาร์เรย์ (Array) กับ สตริงเจอร์

## ทบทวน Array

- เป็นโครงสร้างการเก็บข้อมูลชนิดหนึ่ง เหมือนกันกับ สตริงเจอร์
- ชุดข้อมูลที่เก็บโดย อาร์เรย์ ชื่อเดียวกัน ต้องเป็นข้อมูลชนิดเดียวกัน
- แตกต่างจากข้อมูลใน สตริงเจอร์ ที่เป็นข้อมูลต่างชนิด หรือ ต่างประเภทกันได้



## การใช้งานอาร์เรย์ (Array) กับ สตริงเจอร์

- เมื่อต้องการสร้างตัวแปรหลาย ๆ ตัวที่มีคุณสมบัติเหมือนกัน เราใช้ Array
- เราสามารถสร้างตัวแปร Array ชนิด สตริงเจอร์ ขึ้นมาได้เช่นเดียวกัน

```
typedef struct{  
    char CodeID[5] ;  
    char name[25] ;  
    float GPA;  
} STUDENT;
```

CodeID

name

GPA

1st Structure



11023

Hanson,K.

3.50

2nd Structure



14045

Tony,S.

3.87

3rd Structure



14098

Williams,B.

3.78

สร้างตัวแปร Array จาก สตริงเจอร์ ทำได้โดย

1.ประกาศตัวแปร STUDENT std[3];

หรือ

```
2.struct Student{  
    char CodeID[5];  
    char name[25];  
    float GPA;  
}std[3];
```

.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....



## การกำหนดค่าเริ่มต้นให้กับอาร์เรย์และสตริงในโค้ด

- เราสามารถกำหนดค่าเริ่มต้นให้กับอาร์เรย์ได้ ผ่านการใช้เครื่องหมาย { } เช่น
  - `int A[5] = {9, 7, 10, 0, 2};`
  - `float F[4] = {2.35, 1.78, -1.2, 0.5};`
- วิธีข้างบนนี้จะทำให้ตัวเลขไปปรากฏในอาร์เรย์เรียงตามลำดับจากช่องที่ 0 ไปช่องที่ 1, 2, ... n
- เราสามารถกำหนดค่าเริ่มต้นให้กับสตริงได้เหมือนกันผ่านเครื่องหมาย { }  
เราใส่ข้อมูลเข้าไปทีละตัวตามลำดับการปรากฏตอนประกาศ สตริง  
เช่น `std[0] = {40123, "Ernst,T.",3,21};`

## กำหนดค่าเริ่มต้น อาเรย์ของสตรัค

1. เราใช้ { } กับอาเรย์ด้านนอกตามปกติ
2. ส่วนข้อมูลของสตรัคแต่ละตัวจะมี { } ของมันเอง

### ตัวอย่าง

```
STUDENT    std[3]  =  
            { {40123, "Pinyo,T", 3.21},  
              {40456, "Opas,W", 3.45},  
              {40789, "Ann,C", 3.99}  } ;
```

## การเข้าถึงข้อมูลของสมาชิกใน สตริงเจอร์ จากตัวแปรอาร์เรย์

1. เข้าถึง สตริงเจอร์ ตัวที่ต้องการโดยใช้ชื่อตัวแปรอาร์เรย์ และ index of array []
2. ระบุชื่อสมาชิกที่ต้องการเข้าถึง

ชื่อของตัวแปรอาร์เรย์[index].ชื่อของตัวแปรที่เป็นสมาชิกในสตริงเจอร์  
arrayVariable[index].member

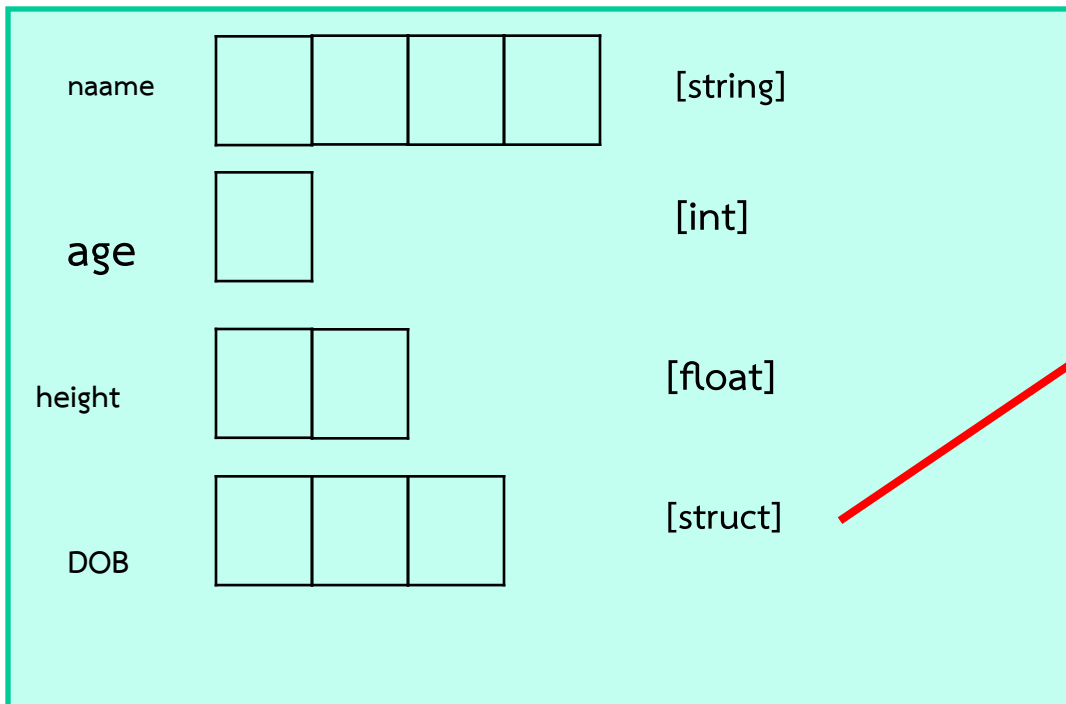
ตัวอย่าง :

```
std[0].CodeID = 45123 ;  
std[1].name = "Usanee,N.";  
std[1].GPA = 3.67;
```

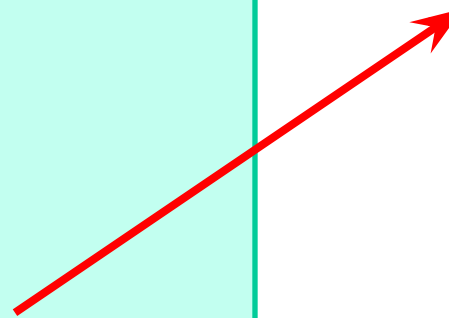
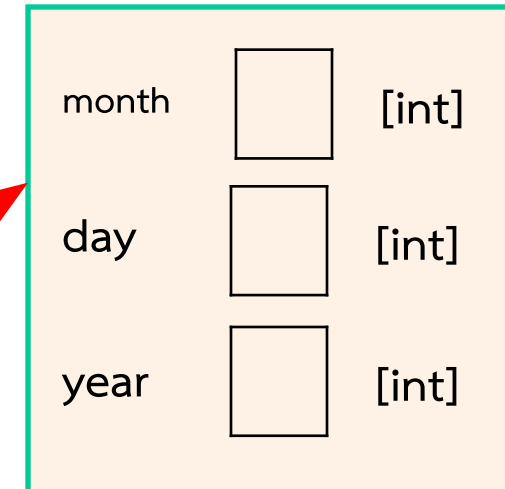
# การใช้งานข้อมูลสมาชิก สตริงเจอร์ ใน สตริงเจอร์

- บางครั้งโครงสร้าง สตริงเจอร์ ที่ถูกสร้างขึ้นมาก่อนหน้าเป็นส่วนประกอบของโครงสร้างใหม่ที่ต้องการจะสร้าง เราสามารถนำ โครงสร้างเดิมกลับมาใช้งานได้ เช่น โครงสร้าง Date สามารถนำมาใช้แสดง วัน เดือน ปี ของนักเรียนได้

Student Record



Date



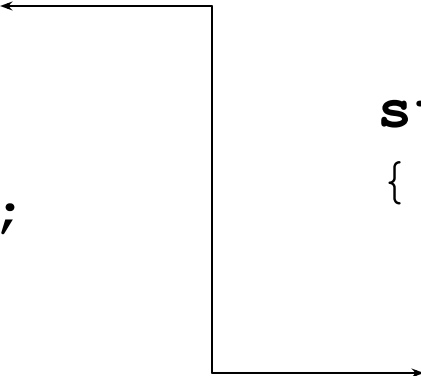
การนำโครงสร้าง Date มาใช้เก็บข้อมูล วัน เดือน ปี ที่มีการจ่ายเงิน

1<sup>st</sup> Structure

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

2<sup>nd</sup> Structure

```
struct account
{
    int acct_no;
    char acct_type;
    char name[80];
    float balance;
    DATE d_payment;
} customer;
```



## การเข้าถึง สมาชิกย่อย (submember)

- สมาชิกย่อย คือ ข้อมูลสมาชิกใน สตรัคเจอร์ ที่อยู่ใน สตรัคเจอร์ ตัวอื่น

variable.member.submember

ตัวอย่าง :     customer.d\_payment.month = 8;

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

```
struct account
{
    int acct_no;
    char acct_type;
    char name[80];
    float balance;
    DATE d_payment;
}customer;
```

# การใช้งาน สตรัคเจอร์ กับ ฟังก์ชัน

## ทบทวน ฟังก์ชัน (Function)

- ส่วนของโปรแกรมที่ถูกสร้างขึ้นเพื่อทำหน้าที่/ทำงานเฉพาะ
- ส่วนประกอบของฟังก์ชัน มีดังนี้
  - Type ประเภทของฟังก์ชัน ขึ้นอยู่กับค่าของผลลัพธ์ที่ส่งกลับ (มีค่าส่งกลับ(return) กับ ไม่มีค่าส่งกลับ (void))
  - Name ชื่อของฟังก์ชัน ซึ่งจะนำมาใช้ในการเรียกใช้ฟังก์ชัน
  - Parameter ตัวแปรที่ส่งเข้ามาในฟังก์ชัน (Input) มีหรือไม่มีก็ได้
  - Statement คำสั่งของโปรแกรมเพื่อให้ฟังก์ชันทำงานตามที่ต้องการ

## ทบทวนฟังก์ชัน (ต่อ)

```
void display() {  
    printf("Comp CampII") ;  
}
```

ฟังก์ชันที่

- ไม่มี parameter
- ไม่มีการคืนค่า (void)

```
int sum(int a,int b) {  
    return(a+b) ;  
}
```

ฟังก์ชันที่

- มี parameter คือ a,b ประเภท integer
- มีการคืนค่า (a+b) เป็นชนิด integer



- ตัวแปร สตริงค์เจอร์ ก็สามารถส่งค่าเป็น parameter เข้าฟังก์ชันได้

```
struct {  
    int IDnum;  
    double payRate;  
    double hours;  
} emp;
```

```
display(emp.IDnum) ;
```

```
calcNet(emp) ;
```

```
void display(int a) {  
    printf("%d",a) ;  
}
```

ฟังก์ชัน display จะรับสตริงค์เจอร์ไปเป็นพารามิเตอร์ในฟังก์ชัน โดยใช้เป็นสมาชิกในสตริงค์เจอร์

ฟังก์ชัน calcNet จะรับสตริงค์เจอร์ไปเป็นพารามิเตอร์ในฟังก์ชัน โดยใช้ส่งไปทั้งสตริงค์เจอร์ (ทุกสมาชิกของสตริงค์เจอร์)

## สรุป สตรัคเจอร์

- สตรัคเจอร์ อนุญาตให้ ตัวแปรที่มีความสัมพันธ์กัน จับกลุ่มกันเพื่อสร้างชนิดข้อมูลใหม่
- สตรัคเจอร์ มีประโยชน์ คล้ายกับ อาเรย์ ในการเก็บชุดข้อมูล
- ชนิดข้อมูลใหม่ที่สร้างโดย สตรัคเจอร์ สามารถใช้งานติดต่อกับฟังก์ชัน ได้เหมือนตัวแปรทั่วไป