STRUCTURE & UNION

030523300- Computer Programming
Asst. Prof. Dr. Choopan Rattanapoka

สตรัคเจอร์ (1)

- ถ้ามีข้อมูลอยู่จำนวนหนึ่งซึ่งเป็นข้อมูลต่างชนิดกันแต่มีความเกี่ยวข้องกัน เช่น ข้อมูลของนักศึกษาซึ่งอาจจะประกอบไปด้วย ชื่อ, นามสกุล, อายุ, เพศ
- การเก็บข้อมูลของนักศึกษาจะต้องสร้างตัวแปรขึ้นมา 4 ตัวต่อนักศึกษาหนึ่งคน
- ถ้านักศึกษามีจำนวนมากก็จำเป็นต้องสร้างตัวแปรมากขึ้นตามไปด้วยซึ่งอาจจะเกิดความสับสนในการเรียกใช้งานตัวแปรเหล่านั้น
- การนำโครงสร้างข้อมูลหรือสตรัคเจอร์มาใช้จะช่วยให้การทำงานในลักษณะนี้ง่ายขึ้น

สตรัคเจอร์ (2)

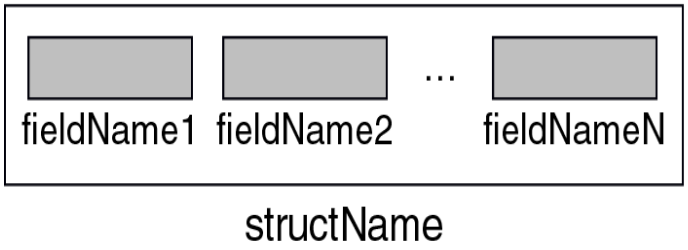
- สตรัคเจอร์ หรือโครงสร้างข้อมูล (**Structure**) เป็นการกำหนดชนิดของตัวแปรขึ้นมาใหม่
- โดยนำตัวแปรชนิดพื้นฐานในภาษาซี อย่างเช่น **int, char,** และ **float** มาประกอบกันเป็นโครงสร้างของตัวแปรชนิดใหม่


```
char  name[24];
int   code;
float salary;
```

- การสร้างตัวแปรสตรัคเจอร์ในภาษาซี สามารถทำได้ **3** วิธี

การสร้างตัวแปรสตรัคเจอร์ วิธีที่ 1

```
struct { field-list } variable_identifier ;
```

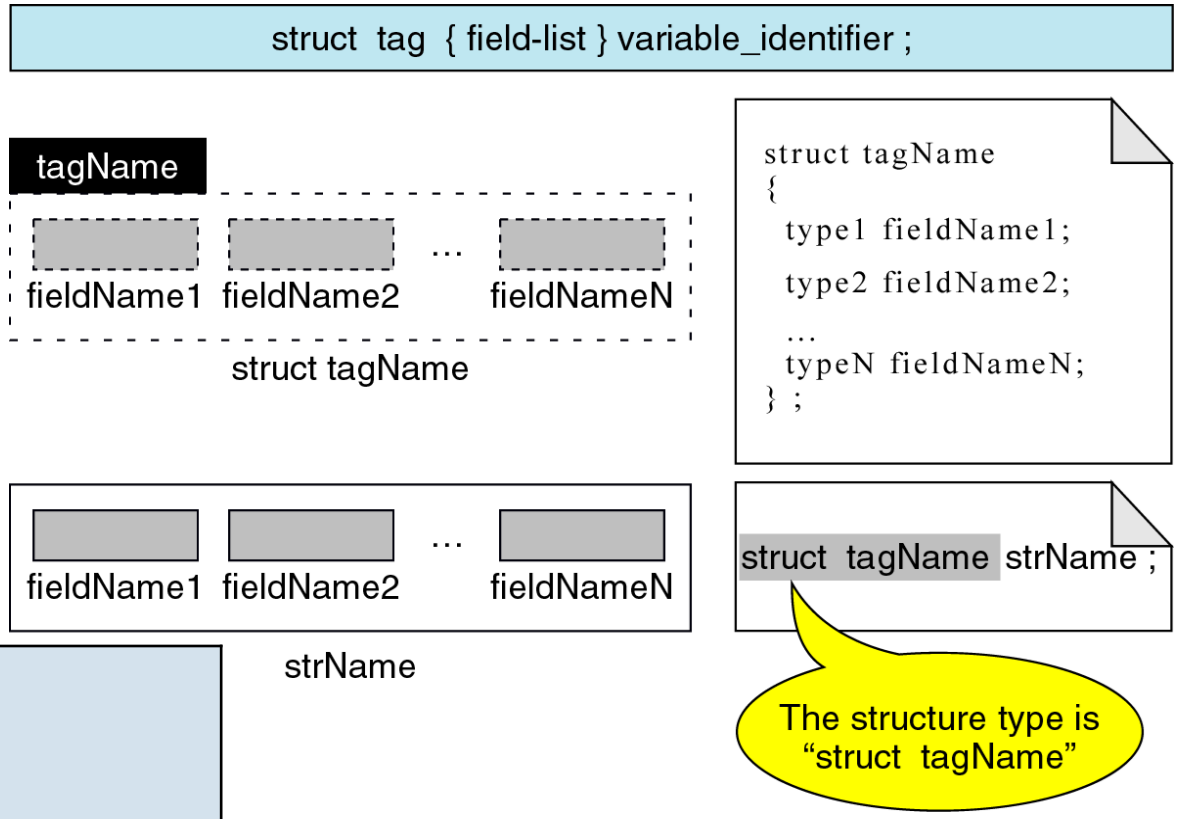



```
struct
{
  type1 fieldName1;
  type2 fieldName2;
  ...
  typeN fieldNameN;
} structName;
```

```
struct {
    char   name[24];
    int    code;
    float  salary;
} no1,  no2;
```

เป็นการประกาศตัวแปรประเภทโครงสร้างชื่อ **no1** และ **no2** โดยแต่ละตัวจะประกอบไปด้วยตัวแปรย่อยคือ name, code, salaryเหมือนกัน

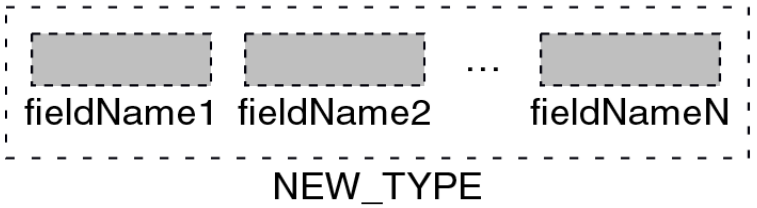
การสร้างตัวแปรสตรัคเจอร์ วิธีที่ 2



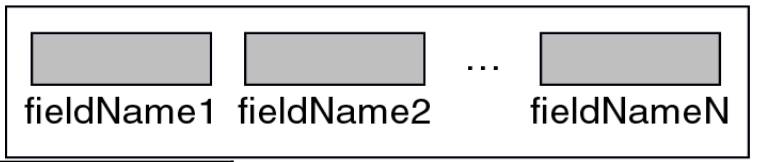

```
struct employee {
        char   name[24];
        int    code;
        float  salary;
} ;
struct employee no1, no2;
```

การสร้างตัวแปรสตรัคเจอร์ วิธีที่ 3

typedef struct { field-list } TYPE-ID ;




```
typedef struct
{
  type1 fieldName1;
  type2 fieldName2;
  ...
  typeN fieldNameN;
} NEW_TYPE;
```



NEW_TYPE strName ;

The typedef name can be used like any type

```
typedef struct {
        char   name[24];
        int    code;
        float  salary;
} employee;
employee no1, no2;
```

สรุปการประกาศตัวแปรสตรัคเจอร์

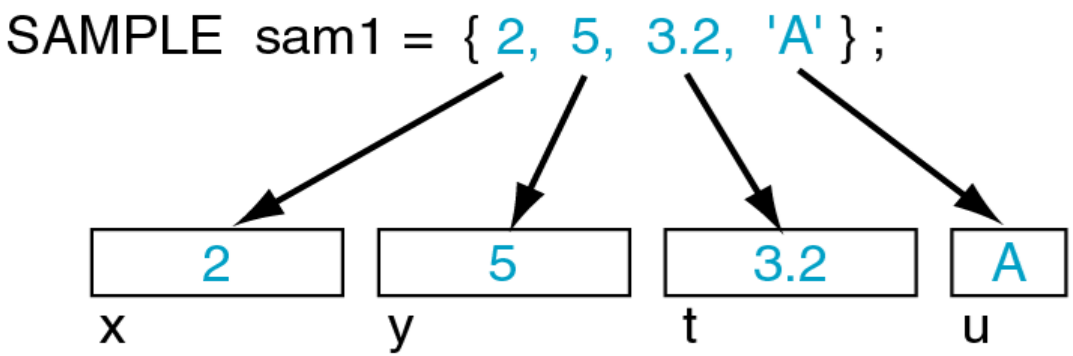
วิธีที่ 1	วิธีที่ 2	วิธีที่ 3
struct { char name[24]; int code; float salary; } no1, no2;	struct employee { char name[24]; int code; float salary; } ; struct employee no1, no2;	typedef struct { char name[24]; int code; float salary; } employee; employee no1, no2;

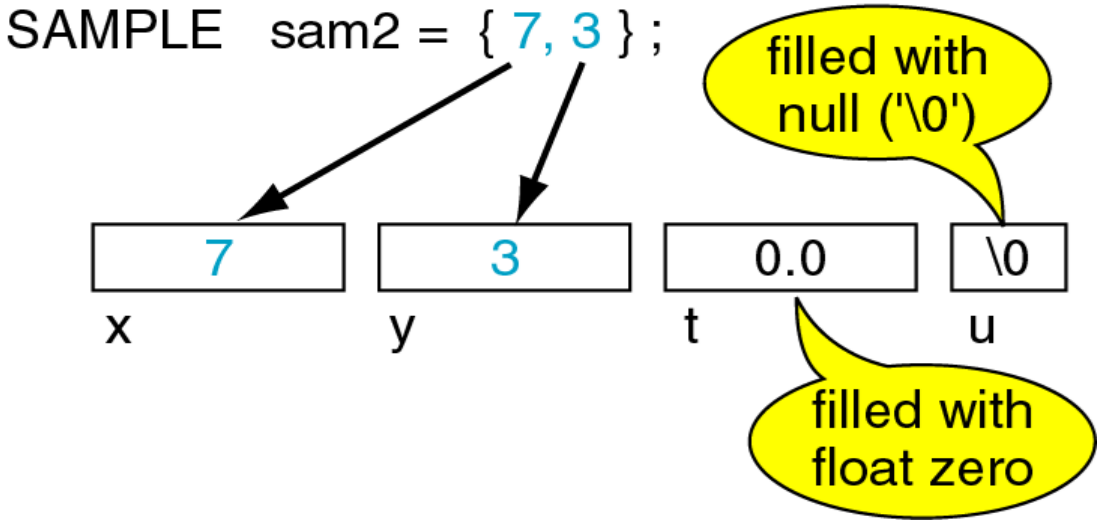
ทั้ง 3 วิธีเป็นการสร้างตัวแปรสตรัคเจอร์ที่สามารถเก็บค่าตัวแปรภายใน 3 ตัวคือ

- char name[24];
- int code;
- float salary;

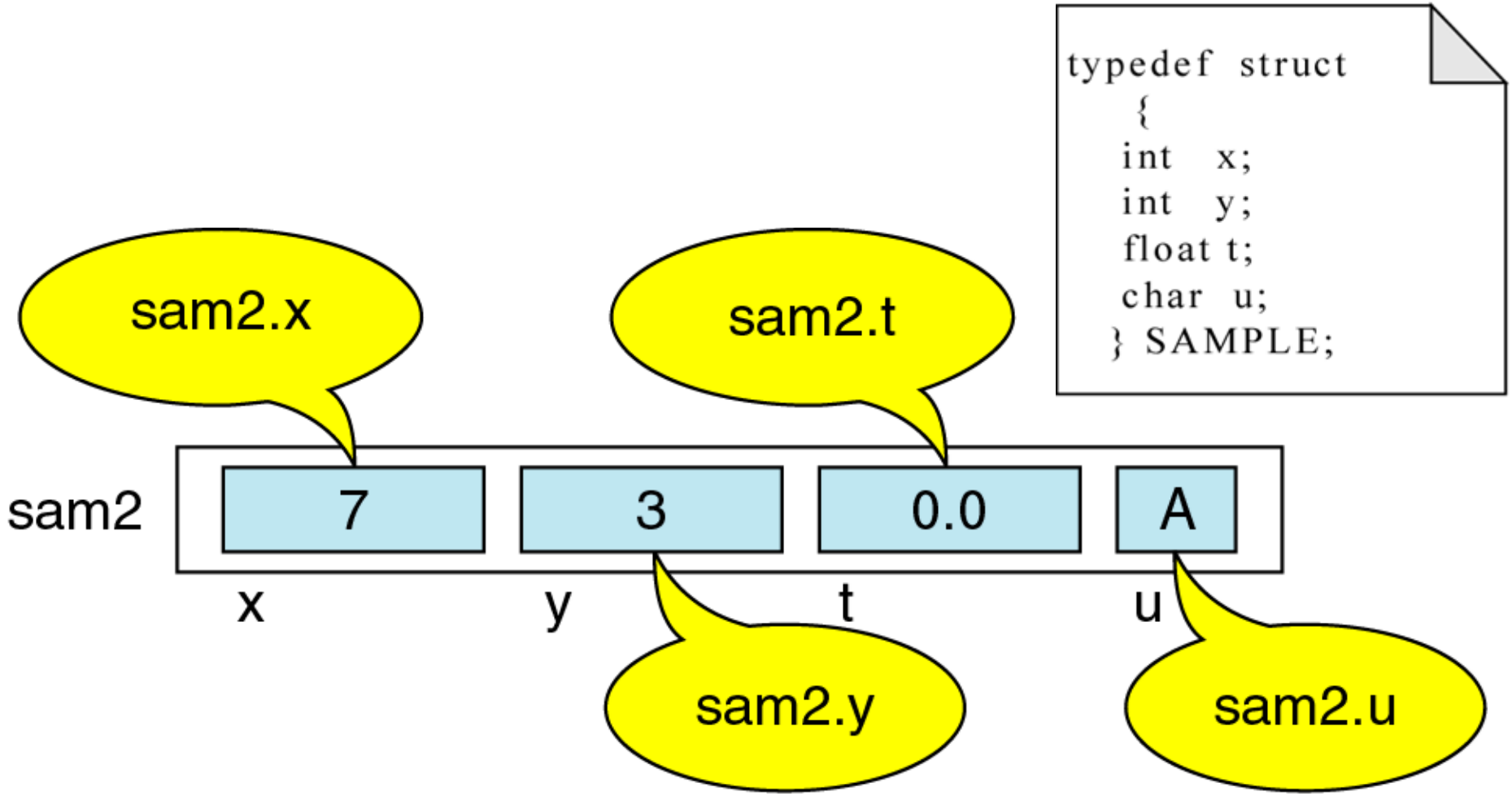
การกำหนดค่าเริ่มต้นให้กับตัวแปรสตรัคเจอร์

```
typedef struct
    {
   int   x;
   int   y;
   float t;
   char  u;
  } SAMPLE;
```





การเข้าถึงข้อมูลของตัวแปรสตรัคเจอร์



ตัวอย่างการอ้างอิงข้อมูลในตัวแปรสตรัคเจอร์

```
#include<stdio.h>

int main(int argc, char **argv) {

     struct  product {
              int     code;
              char  productName[64];
              float   price;
     } computer;

     computer.code = 1000;
     strcpy(computer.productName,  “Core 2 duo”);
     computer.price = 18000.00;
}
```

แบบฝึกหัด 1

```
#include<stdio.h>

int main(int argc, char **argv) {
    typedef   struct  {
            int     code;
            char  name[64];
            float   salary;
    } Employee;

    Employee   emp;
    printf("Enter code : " );   scanf("%d",  &(emp.code));
    printf("Enter name : ");   gets(emp.name);
    printf("Enter salary : ");   scanf("%f",  &(emp.salary));

    printf("%d\t%s\t%f\n",  emp.code,  emp.name,  emp.salary);
 }
```

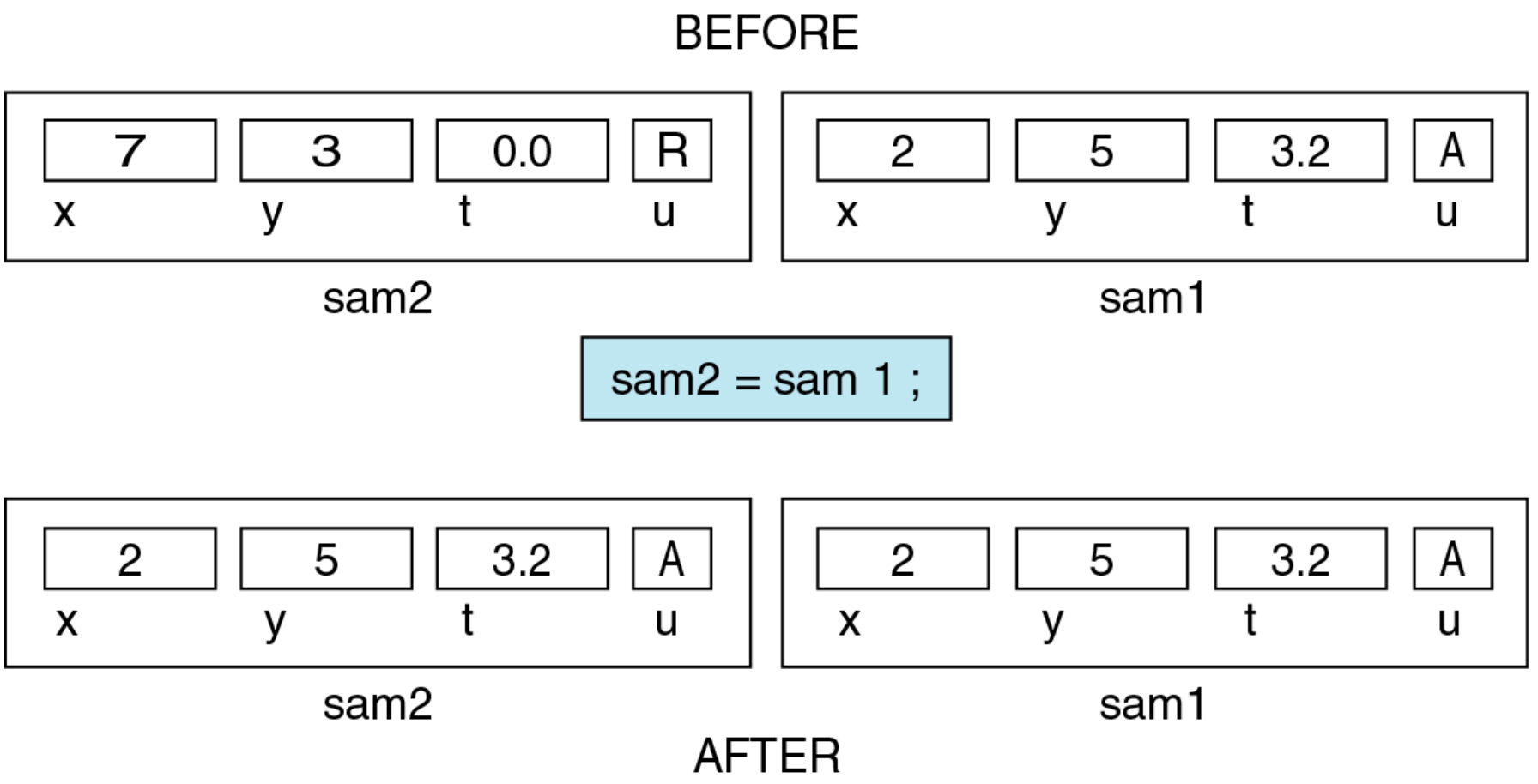
เมื่อป้อนค่า ตามลำดับ

1000

Somchai

10720

การ copy ค่าระหว่างตัวแปรสตรัคเจอร์



อาร์เรย์ของตัวแปรโครงสร้าง

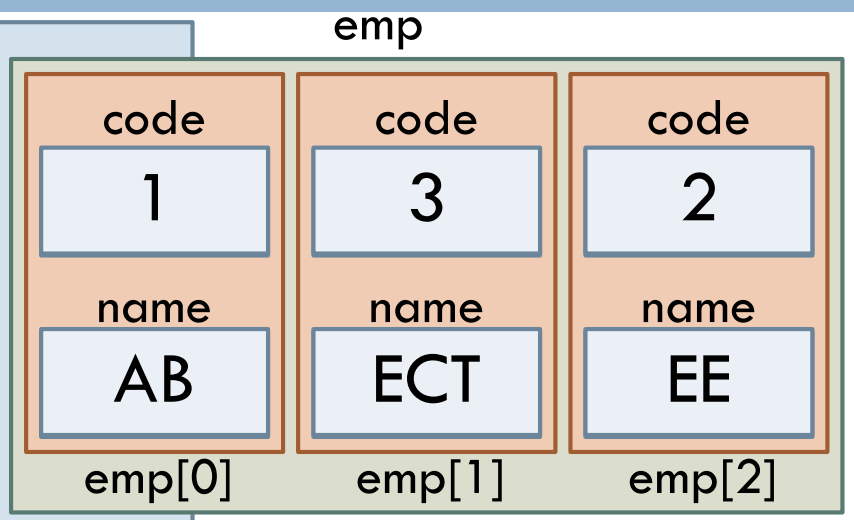
- ในกรณีที่ต้องการใช้ตัวแปรชนิดโครงสร้างแบบเดียวกันหลายๆตัว เราสามารถสร้างตัวแปรอาร์เรย์ชนิดโครงสร้างขึ้นมาได้ เช่นเดียวกันกับการสร้างตัวแปรอาร์เรย์กับชนิดข้อมูลพื้นฐานอื่นๆ

```
struct  employee {
        char   name[24];
        int      code;
        float   salary;
};
struct  employee  emp[100];
```

ตัวอย่างการใช้งาน

```
#include<stdio.h>
int main(int argc, char **argv) {
    int  i;
    typedef struct  {
        int     code;
        char  name[64];
    } Employee;

   Employee   emp[3];
    for(i = 0; i < 3; i++) {
        printf("Enter code : " );
        scanf("%d",  &(emp[i].code));
        printf("Enter name : ");
        gets(emp[i].name);
    }
    for(i = 0;  i < 3; i++)
        printf("%d:%d:\t%s\n",  i, emp[i].code,  emp[i].name);
}
```



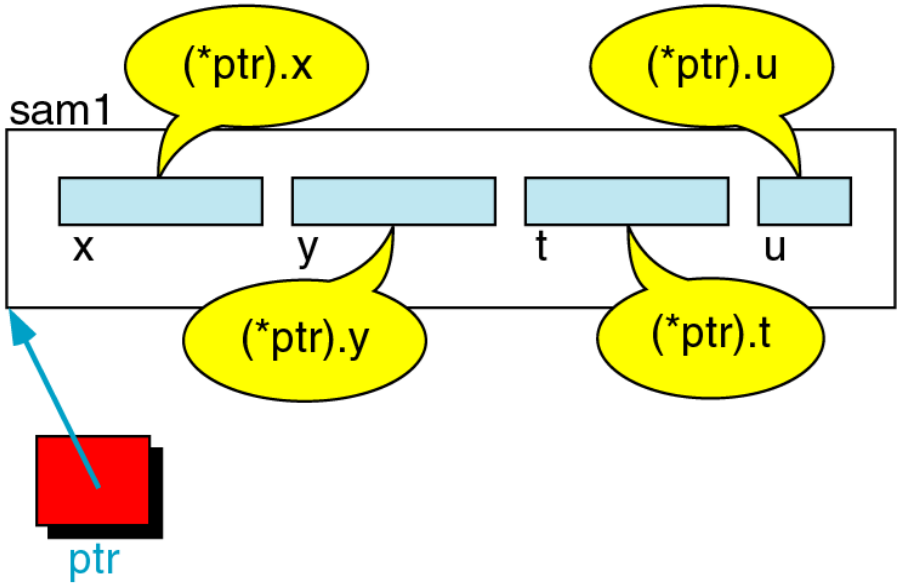
ผลการรัน :

```
Enter code : 1
Enter name :AB
Enter code : 3
Enter name :ECT
Enter code : 2
Enter name :EE
0:1:        AB
1:3:        ECT
2:2:        EE
```

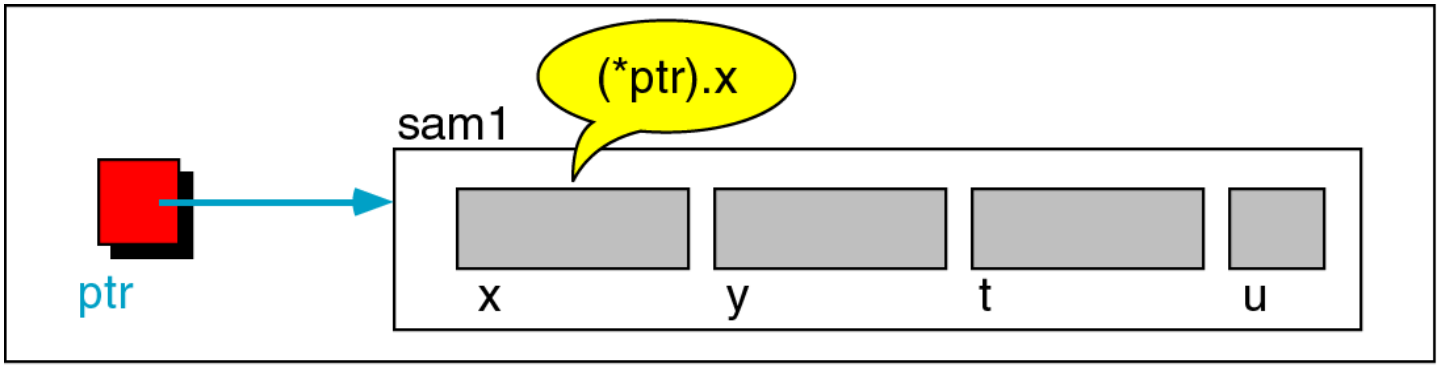
พอยน์เตอร์ของตัวแปรโครงสร้าง

- เราสามารถที่จะสร้างตัวแปรพอยน์เตอร์ของตัวแปรโครงสร้างได้ ซึ่งการใช้งานส่วนใหญ่จะเป็นการผ่านค่าตัวแปรเข้าไปในฟังก์ชัน

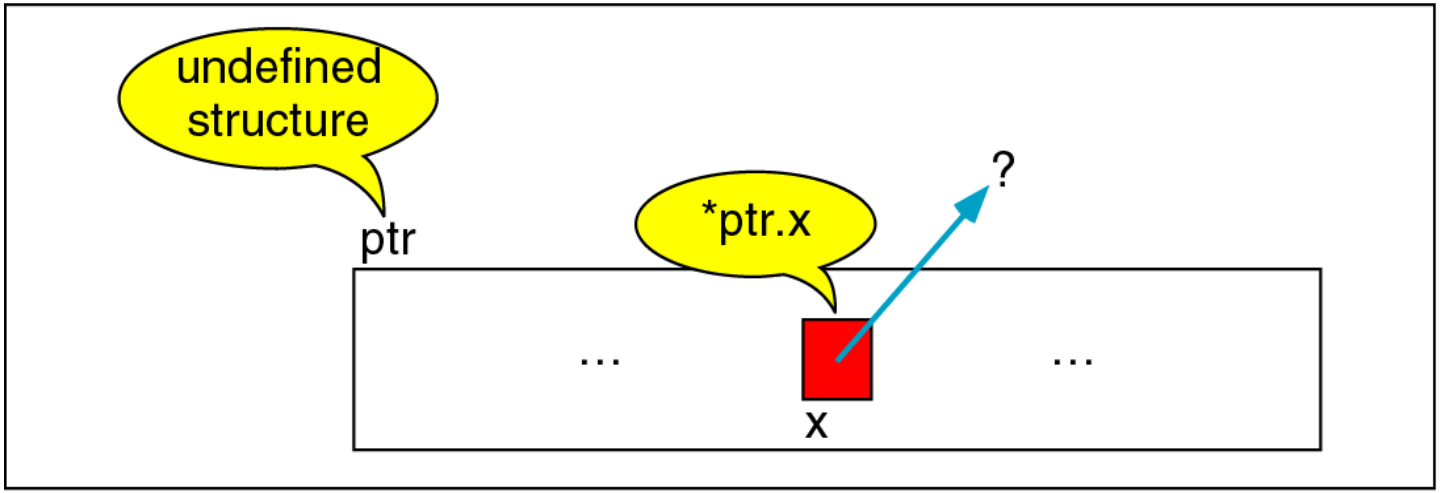
```
typedef struct
  {
   int   x;
   int   y;
   float t;
   char  u;
  } SAMPLE;
   ...
   SAMPLE   sam1;
   SAMPLE  *ptr;
   ...
   ptr = &sam1;
   ...
```



การอ้างค่าผ่านตัวแปรพอยน์เตอร์ (1)



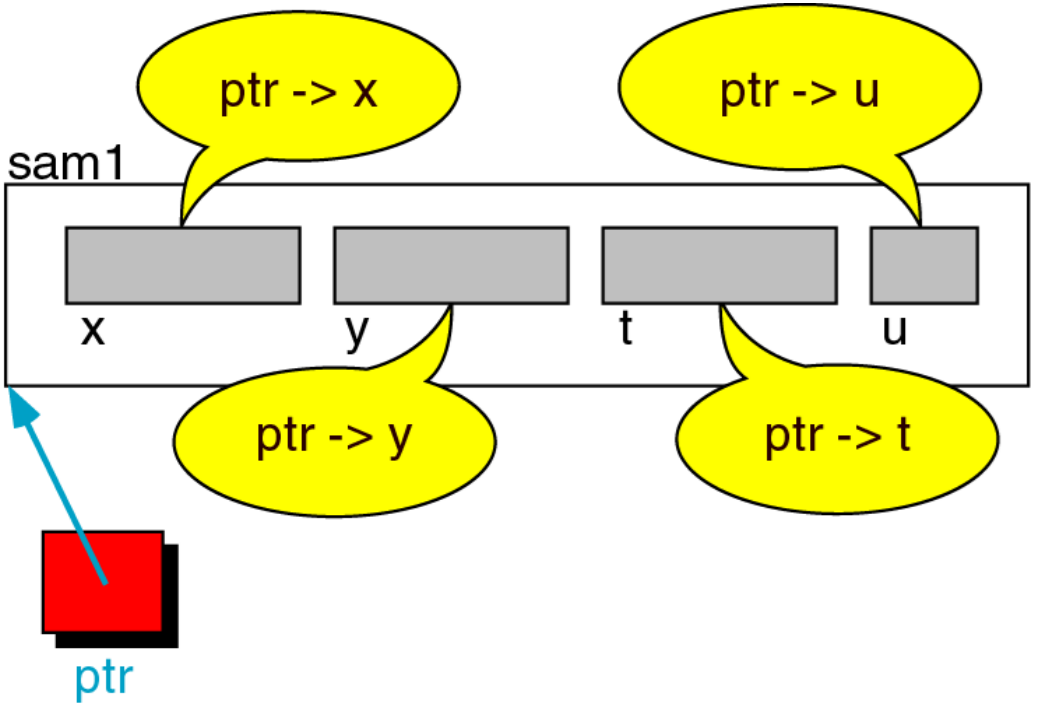
The correct reference

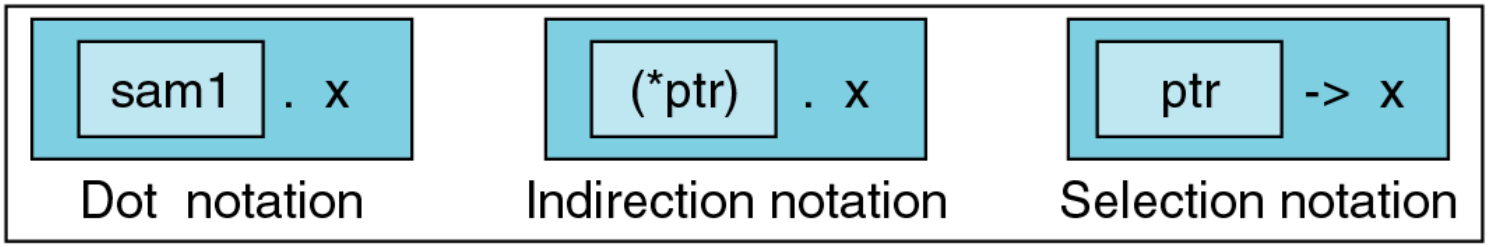


The wrong way to reference the component

การอ้างค่าผ่านตัวแปรพอยน์เตอร์ (2)

```
typedef struct
  {
   int   x;
   int   y;
   float t;
   char  u;
  } SAMPLE;
  ...
  SAMPLE   sam1;
  SAMPLE  *ptr;
  ...
  ptr = &sam1;
  ...
```

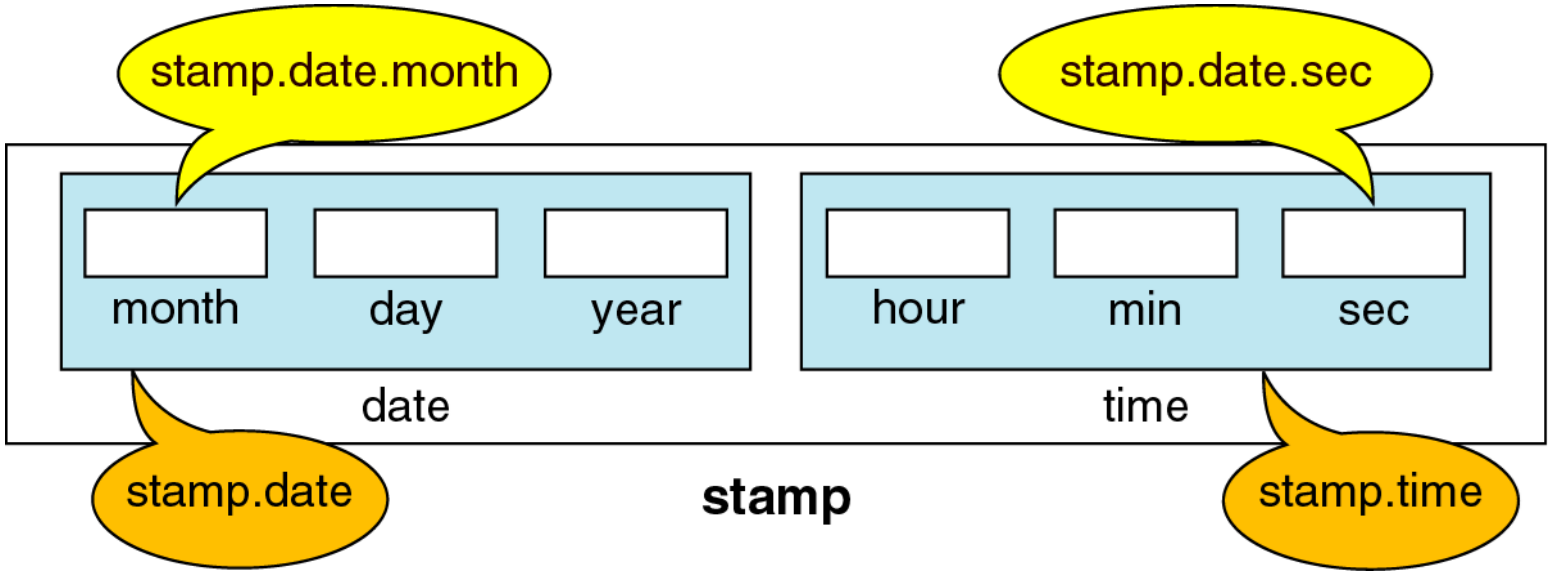




three ways to reference the field **x**

ตัวแปรโครงสร้างในตัวแปรโครงสร้าง

โครงสร้างข้อมูล DATE	โครงสร้างข้อมูล TIME	โครงสร้างข้อมูล STAMP
typedef struct { int month; int day; int year; } DATE;	typedef struct { int hour; int min; int sec; } TIME;	typedef struct { DATE date; TIME time; } STAMP; STAMP stamp;



ตัวแปรโครงสร้างและฟังก์ชัน

- ตัวแปรชนิดโครงสร้างสามารถนำไปใช้งานร่วมกับฟังก์ชันได้ ไม่ว่าจะเป็นการผ่านตัวแปรโครงสร้างทั้งหมด หรือ เพียงแค่สมาชิกภายในตัวแปรโครงสร้าง

```
struct  num_string {
      int i;
      char st[10];
};

void  output(struct num_string a)
{
  printf("x.i = %d\n",a.i);
  printf("x.st = %s\n",a.st);
}
```

แบบฝึกหัด 2

```
#include <stdio.h>

struct num_string {
        int i;
        char st[10];
};

void output(struct  num_string  a) {
        printf("x.i = %d\n",   a.i);
        printf("x.st = %s\n", a.st);
}

int  main(int  argc,  char  **argv) {
        struct    num_string   x;
        x.i  = 100;
        strcpy(x.st, "COMPUTER");
        output(x);
}
```

แบบฝึกหัด 3

```
#include <stdio.h>
struct  num_string {
        int i;
        char st[10];
};

void  output(struct  num_string  a) {
        printf("x.i = %d\n",   a.i);
        printf("x.st = %s\n", a.st);
}

void  modify(struct num_string *a,
                int  i,  char  *st)
{
        a->i = i;
        strcpy(a->st,  st);
}
```

```
int  main(int  argc,  char  **argv)
{
        struct  num_string  x;
        x.i = 100;
        strcpy(x.st, "COMPUTER");
        modify(&x, 50, “PC”);
        output(x);
}
```

ยูเนียน

- ยูเนียน (union) เป็นข้อมูลซึ่งคล้ายกับสตรัคเจอร์คือมีโครงสร้างภายในประกอบด้วยตัวแปรชนิดพื้นฐานในภาษาซีเช่นกัน
- แต่ส่วนที่แตกต่างกันก็คือ สมาชิกของข้อมูลชนิดยูเนียนจะใช้พื้นที่ในหน่วยความจำร่วมกัน โดยเปลี่ยนกันใช้พื้นที่ในตำแหน่งนั้นคนละช่วงเวลา ซึ่งเป็นการประหยัดเนื้อที่ในหน่วยความจำ
- รูปแบบการประกาศตัวแปรยูเนียนจะคล้ายกับโครงสร้างข้อมูล

```
union  employee {
      char   name[24];
      int      code;
      float   salary;
 } emp;
```

หน่วยความจำที่ใช้ของสตรัคเจอร์ และยูเนียน

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

typedef struct {
        int x;
        int y;
        int z;
} myStruct;

typedef union {
        int x;
        int y;
        int z;
} myUnion;
```

```
int main(int argc, char *argv[])
{
  printf("Size of MyStruct = %d\n", sizeof(myStruct));
  printf("Size of myUnion  = %d\n", sizeof(myUnion));
  system("PAUSE");
  return 0;
}
```