บทที่ 4
โครงสร้างข้อมูลแบบ Struct

สุนทรี คุ้มไพโรจน์

โครงสร้างข้อมูล แบบ Structrue

- **เป็นการกำหนดชนิดข้อมูลใหม่ขึ้นมาใช้งานโดยชนิดข้อมูลนี้**
- **เกิดจากการรวมกันของชนิดข้อมูลพื้นฐานที่อาจจะเป็นชนิดเดียวกันหรือต่างชนิดกันก็ได้**

ตัวอย่าง

โครงสร้างข้อมูลนักศึกษาให้ชื่อว่า student ประกอบด้วย

ชื่อเป็นชนิด char

นามสกุลเป็นชนิด char

อายุเป็นชนิด int

เพศเป็น char

เกรดเฉลี่ยเป็น float

ระเบียน (Record / struct)

การประกาศตัวแปรระเบียนในภาษาซี

```
struct userrec {
        int      id;
        char  password[10];
        float  limit, used;
}
```

แผนภาพแสดงระเบียน `userrec`

id	password	limit	used
1234	aaaa	100.00	50.5

เปรียบเทียบ Array กับ Struct

Array

- เก็บข้อมูลจำนวนมาก ๆได้
- เข้าถึงข้อมูลแต่ละตัวได้โดยง่าย
- แต่ข้อมูลเหล่านั้นต้องเป็นชนิดเดียวกันเท่านั้น

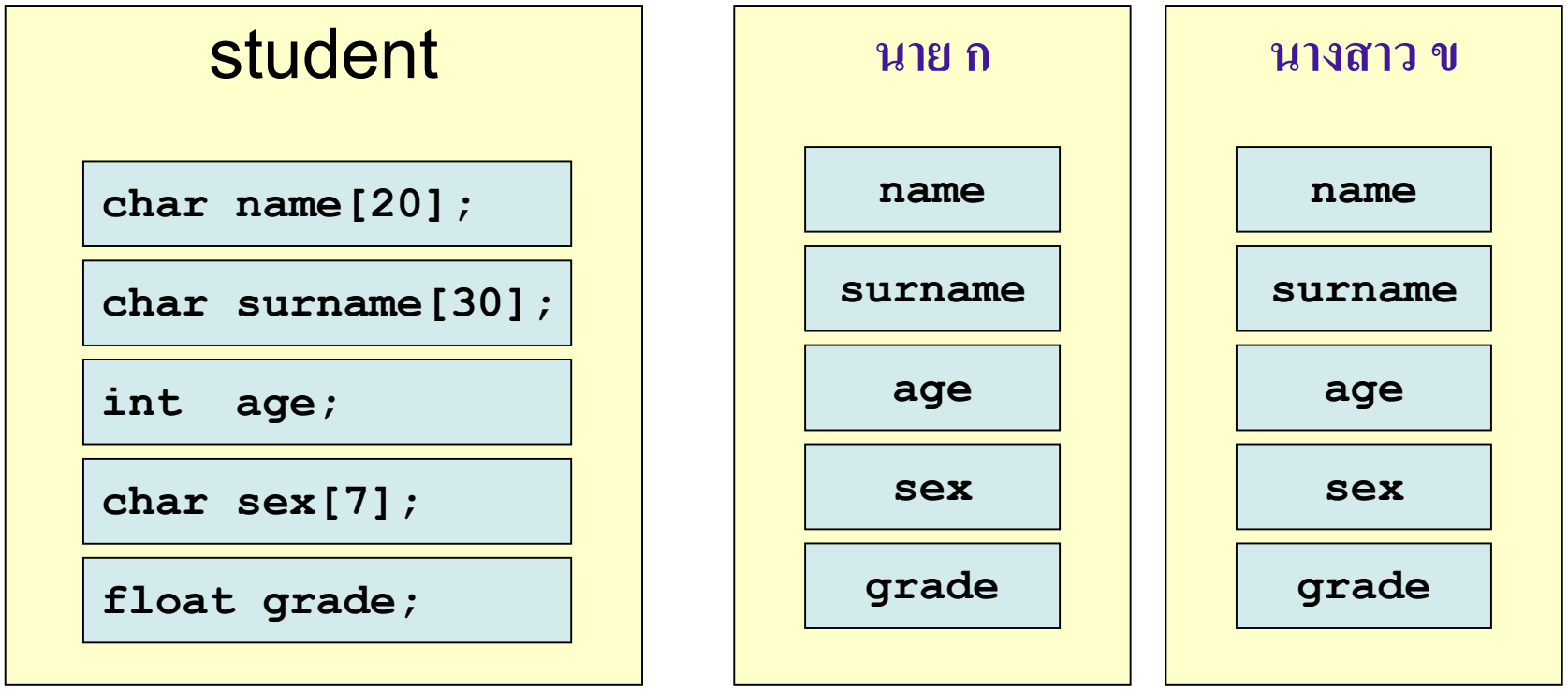
```
int  point[5];
char  car[10];
```

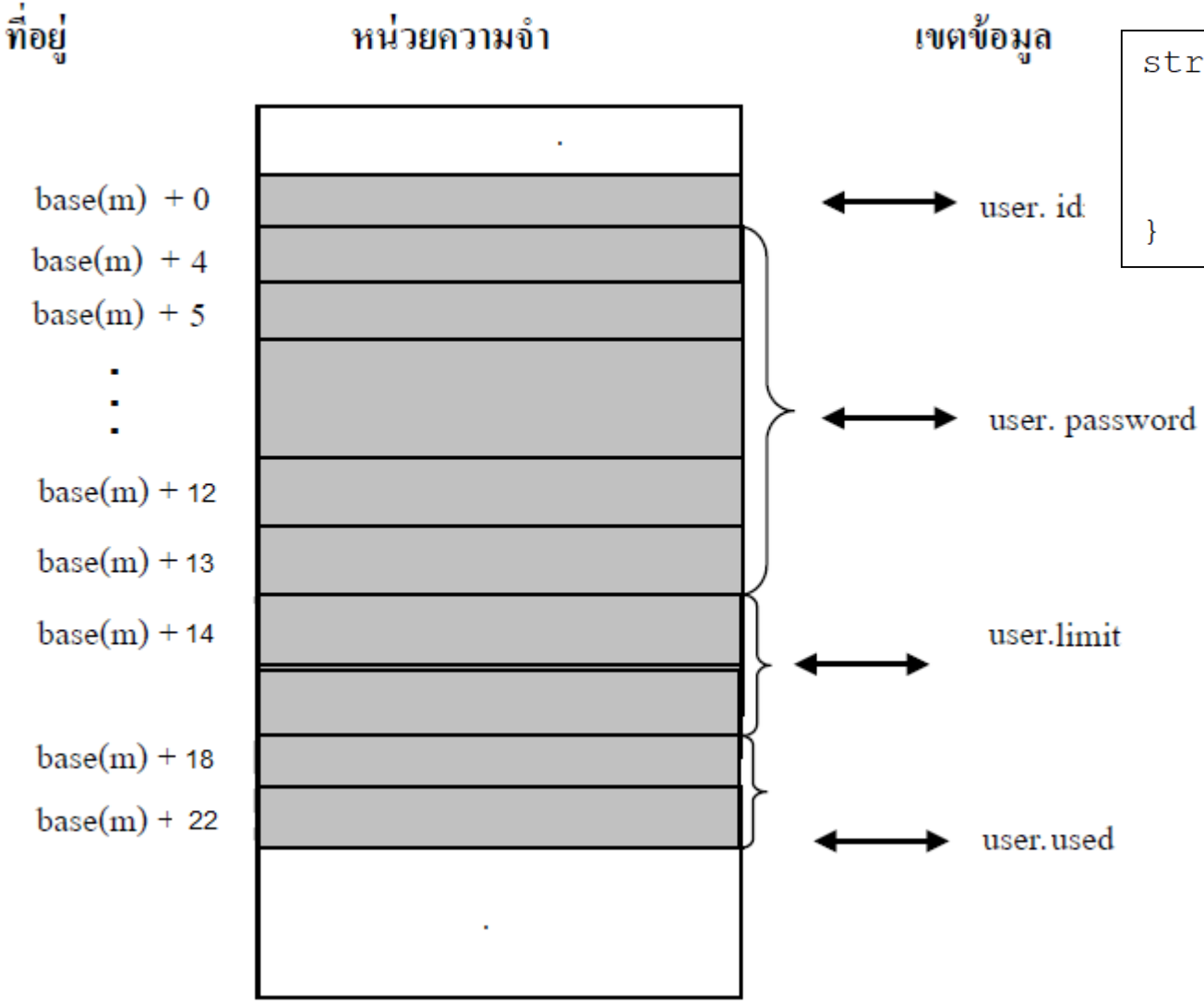
Struct

- ข้อมูลแต่ละตัวในกลุ่มสามารถมีชนิดต่างกันก็ได้ เช่น ข้อมูลผลการเรียนของนักศึกษาแต่ละคน
- เทียบเท่ากับ Record ในภาษาอื่น

```
struct student {
        char name[20];
        int age;
        float grade;
}
```

โครงสร้างข้อมูล (Structure)

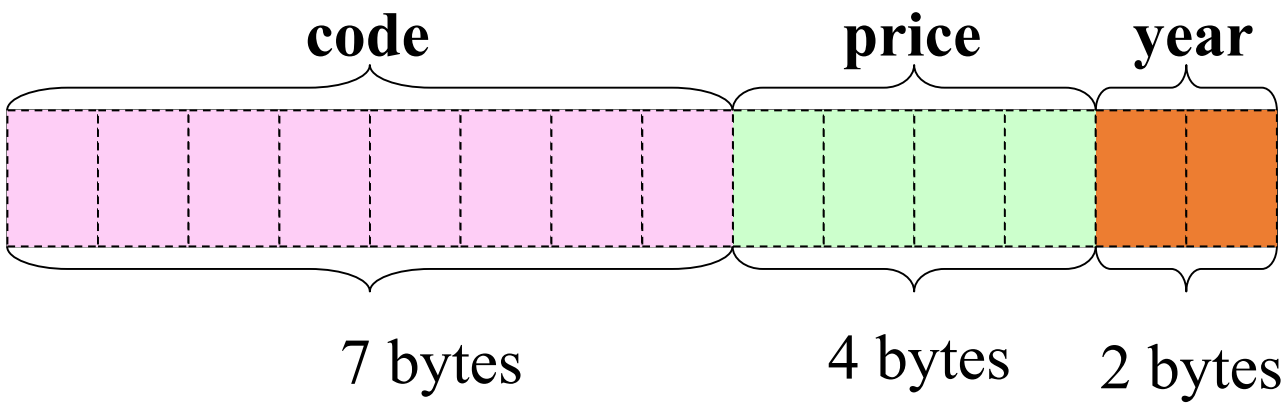





```
struct userrec {
      int     id;
      char  password[10];
      float  limit, used;
}
```

โครงสร้างข้อมูล (Structure)

```
struct book {
        char   code[7];
        float  price;
        int    year;
}
```



การกำหนดโครงสร้างข้อมูล

- ก่อนการกำหนดตัวแปรชนิดโครงสร้างเพื่อมาใช้งาน ต้องกำหนดโครงสร้างข้อมูลสำหรับตัวแปร
- โดยการกำหนดโครงสร้างข้อมูลในภาษา C มีข้อกำหนดดังนี้

```
struct  ชื่อโครงสร้างข้อมูล  {
        ชนิดข้อมูล     ชื่อข้อมูลที่หนึ่ง;
        ชนิดข้อมูล     ชื่อข้อมูลที่สอง;
          ...
        ชนิดข้อมูล     ชื่อข้อมูลที่ n;
} ;
```

ตัวอย่างการกำหนดโครงสร้างข้อมูล

```
struct student {
        char    name[20];
        char    surname[30];
        int     age;
        char    sex[7];
        float   grade;
};
```

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

```
struct sdate {
        char    day[3],month[3];
        char    year[5];
};
```

การกำหนดตัวแปรชนิดโครงสร้าง

- เมื่อกำหนดโครงสร้างข้อมูลเรียบร้อยแล้ว เราสามารถกำหนดให้ตัวแปรของเรามีโครงสร้างตามที่กำหนดไว้แล้วได้โดยใช้รูปแบบต่อไปนี้

struct ชื่อโครงสร้างข้อมูล ชื่อตัวแปร;

ตัวอย่าง

```
struct student student1;
struct student Manee, Peeti;
struct date birthday;
```

การกำหนดตัวแปรและโครงสร้าง

- การประกาศตัวแปรอาจทำพร้อมการกำหนดโครงสร้างข้อมูล โดยใช้รูปแบบดังนี้

```
struct ชื่อโครงสร้างข้อมูล {
        ชนิดข้อมูล     ชื่อสมาชิกที่หนึ่ง;
        ชนิดข้อมูล     ชื่อสมาชิกที่สอง;
            ...
        ชนิดข้อมูล     ชื่อสมาชิกที่ n;
} ชื่อของตัวแปร ;
```

ตัวอย่างการกำหนดตัวแปรและโครงสร้างข้อมูล

```
struct student {
        char    name[20];
        char    surname[30];
        int     age;
        char    sex[7];
        float   grade;
} student1;
```

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

สมาชิกในตัวแปรชนิดโครงสร้าง

- พิจารณา ส่วนประกอบต่างๆ ในโครงสร้าง เราเรียกส่วนประกอบเหล่านี้ว่า **“สมาชิก”**

```
struct student {
        char   name[20];
        char   surname[30];
        int    age;
        char   sex[7];
        float  grade;
} student1;
```

สมาชิก,field

การอ้างถึงสมาชิกในตัวแปรชนิดโครงสร้าง

- การอ้างถึงสมาชิกในตัวแปรชนิดโครงสร้าง ทำได้โดยบอกชื่อตัวแปรโครงสร้างตามด้วยจุด ต่อด้วยชื่อสมาชิกของโครงสร้างนั้น ๆ
- มีรูปแบบดังนี้

ชื่อตัวแปรโครงสร้าง.ชื่อสมาชิก;

ตัวอย่าง

`student1.name`	อ้างถึงข้อมูลชื่อของตัวแปร `student1`
`student1.surname`	อ้างถึงข้อมูลสกุลของตัวแปร `student1`
`birthday.day`	อ้างถึงข้อมูลวันของตัวแปร `birthday`
`birthday.month`	อ้างถึงข้อมูลเดือนของตัวแปร `birthday`

การกำหนดค่าให้ตัวแปรชนิดโครงสร้าง

- ตัวแปรชนิดโครงสร้างจะมีสมาชิกหลายตัว เราจะต้องกำหนดข้อมูลให้แต่ละสมาชิกในโครงสร้างนั่น ๆ ถ้าสมาชิกไม่ได้เป็นข้อความ ให้มองเหมือนการกำหนดตัวแปรปกติ เพียงแต่เราต้องอ้างถึงโดยใช้ชื่อตัวแปรนั้น ๆ ก่อน

ตัวอย่าง

```
birthday.day = 30;
birthday.month = 9;
birthday.year = 2525;
```

การกำหนดค่าเริ่มต้นให้กับโครงสร้าง

เราสามารถกำหนดค่าเริ่มต้นให้แก่โครงสร้างได้

แต่ตัวแปรโครงสร้างจะต้องกำหนดค่าเริ่มต้นให้ตามลำดับของตัวแปรที่ประกาศไว้

ตัวอย่างโปรแกรม (L45.c)

```c
#include <stdio.h>
#include <conio.h>
int main() {
    struct book {
        char name[50];
        float price;
    } b1 = {"harry",120};
    struct book b2 = {"plotter",130} ;
    struct book b3 = {150,"bangkok"} ;

    printf("%s\n",b1.name);
    printf("%f\n",b3.price);
    return 0;
}
```

```
main.c:9:26: warning: initialization makes integer
er without a cast [-Wint-conversion]
    struct book b3 = {150,"bangkok"} ;
                         ^~~~~~~~~
main.c:9:26: note: (near initialization for 'b3.na
harry
0.000000


...Program finished with exit code 0
Press ENTER to exit console.
```

ตัวอย่างโปรแกรมการกำหนดข้อมูลโครงสร้าง (L46.c)

```c
#include <stdio.h>
struct income {
    float   salary,bonus;
    int     age;
};
int main(){
   struct income somsri;
   somsri.salary = 18000.00;
   somsri.bonus  = 30000.00;
   somsri.age    = 23;
   printf("%f\n",somsri.salary);
   printf("%f\n",somsri.bonus);
   printf("%d\n",somsri.age);
   return 0;
}
```

การกำหนดสมาชิกของโครงสร้างที่เป็นข้อความ

- ถ้าสมาชิกของโครงสร้างเป็นข้อความ เราจะกำหนดตรงๆ เหมือนสมาชิกที่เป็นชนิดอื่นไม่ได้
- วิธีการกำหนดข้อมูลชนิดข้อความ ให้ทำการกำหนดผ่านคำสั่ง **strcpy** หรือรับค่าจากคีย์บอร์ดด้วยคำสั่ง **gets** หรือ **scanf** ก็ได้

ตัวอย่างการกำหนดสมาชิกของโครงสร้างที่เป็นข้อความ (L41.c)

```c
#include <stdio.h>
#include <string.h>
struct letter {
        char   name[20];
        char   address[30];
        char   message[40];
};
int main(){
    struct letter first;
    printf("Enter name :");
    scanf("%s",first.name);
    printf("Enter address :");
    scanf("%s",first.address);
    strcpy(first.message, "How r u?");

    printf("\nNAME is %s",first.name);
    printf("\nAddress is %s",first.address);
    printf("\nMessage : %s",first.message);
    return 0;
}
```

การแสดงผลสมาชิกของตัวแปรชนิดโครงสร้าง

- **เนื่องจากสมาชิกในตัวแปรชนิดโครงสร้างก็คือตัวแปรชนิดพื้นฐานทั่วไปในภาษา C ดังนั้นการแสดงผลจะใช้คำสั่ง printf และตามด้วยรหัสควบคุมการแสดงผลให้ตรงกับชนิดข้อมูล**

อะเรย์ของตัวแปรชนิดโครงสร้าง

- ปกติเราจะใช้ข้อมูลชนิดโครงสร้างมากกว่าหนึ่งข้อมูล เช่น โครงสร้างนักเรียนประกอบด้วย รหัส, ชื่อ, เพศ, และอายุ
- แต่โดยส่วนมากเราจะมีข้อมูลของนักเรียนหลายคน
- ดังนั้นเราสามารถสร้างตัวแปรของนักเรียนให้เป็น อะเรย์ของตัวแปรโครงสร้าง เพื่อทำการเก็บข้อมูลแบบโครงสร้างไว้เป็นกลุ่มข้อมูลที่เกี่ยวข้องกัน เช่น โครงสร้างนักเรียน ซึ่งโรงเรียนหนึ่งจะต้องมีนักเรียนหลายคนจะต้องเก็บข้อมูลของนักเรียนหลาย ๆ คนเพื่อนำมาประมวลผลต่อไป

อะเรย์ของโครงสร้าง (Arrays of structures)

- ข้อมูลโครงสร้าง 1 ตัวก็คือข้อมูล 1 รายการ (1 record)
- เช่น กำหนดข้อมูลโครงสร้างชนิด student

```
struct student {  char  code[9];
                  char  name[41];
                  float   mid, lab, fin;
                  char   grade;
  } stu1;
```

- stu1 เป็นตัวแปรโครงสร้าง สามารถเก็บข้อมูลนักศึกษาได้ 1 คน
- ถ้าต้องการเก็บข้อมูลนักศึกษาหลายๆคน ไม่ต้องประกาศตัวแปรโครงสร้างหลายๆตัว สามารถสร้างอะเรย์ของโครงสร้างได้

การประกาศอะเรย์ของโครงสร้าง

- มีรูปแบบดังนี้

struct ชื่อโครงสร้าง ตัวแปรโครงสร้าง[จำนวนสมาชิก];

- เช่น

```
struct student {
                char code[9];
                char name[41];
                float mid, lab, fin;
                char grade;
        };
struct student stu[3];
```

- ได้ตัวแปรอะเรย์ที่มีสมาชิก 3 ตัว (เพื่อเก็บข้อมูล student 3 คน)
- สมาชิกแต่ละตัวคือ ตัวแปรโครงสร้าง(structure) ชนิด student

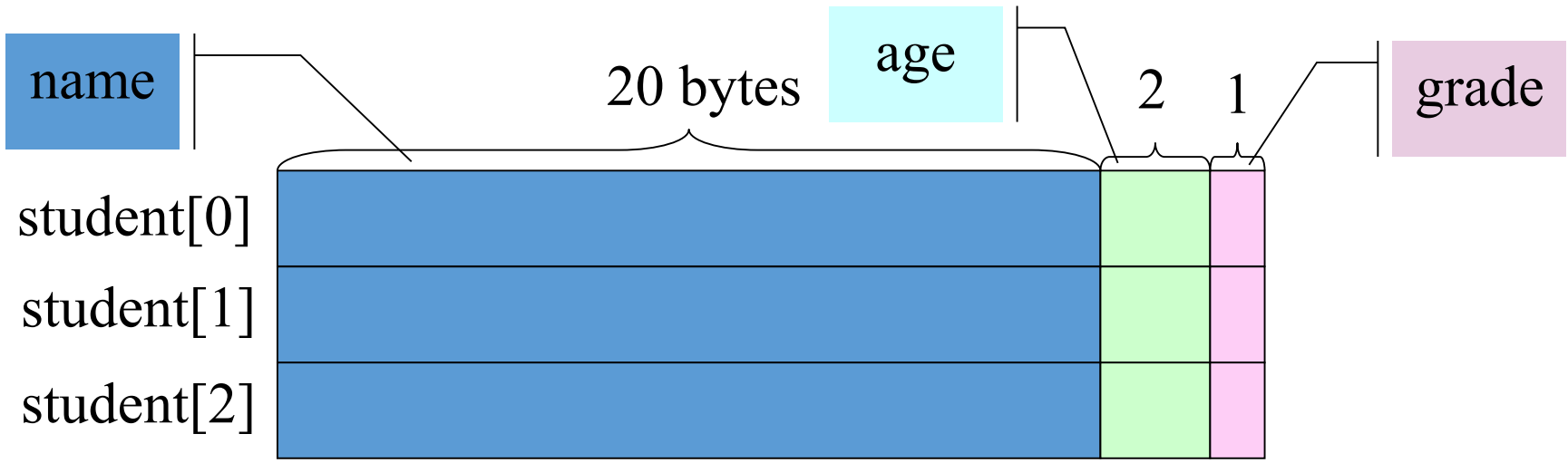
ตัวอย่าง

- **ตัวอย่างต่อไปนี้จะแสดงการใช้อะเรย์ของโครงสร้าง**

```
struct profile {
    char name[20];
    int age;
    char grade;
}student[10];
```

Array of Structure

```
struct profile {
    char name[20];
    int  age;
    char grade;
}student[3];
```



การกำหนดค่าเริ่มต้นให้ข้อมูลอะเรย์ของโครงสร้าง

```
struct ชื่อโครงสร้าง  ตัวแปร[จำนวนสมาชิก] = {
                    { รายการสมาชิกตัวที่ 1},
                    { รายการสมาชิกตัวที่ 2},
                        ...
                    { รายการสมาชิกตัวที่ n}
 };
```

- เช่น

```
struct student  stu[100] =  {  {“61007332”, “Somchai”, 20, 15, 30, ‘C’} ,
                               {“61017332”, “Decha”, 25, 18, 40, ‘A’} ,
                               {“61027332”, “Tanee”, 12, 15, 25,  ‘D’}   };
```

ตัวอย่างโปรแกรม (L42.c)

```c
#include <stdio.h>
#include <string.h>
int main() {
    struct profile {
        char name[20];
        int age;
        char grade;
    }student[10];

    strcpy(student[0].name, "Manee");
    student[2].age = 12;
    student[4].grade = 'A';
    printf("%s\n",student[0].name);
    printf("%d\n",student[2].age);
    printf("%c\n",student[4].grade);
    return 0;
}
```

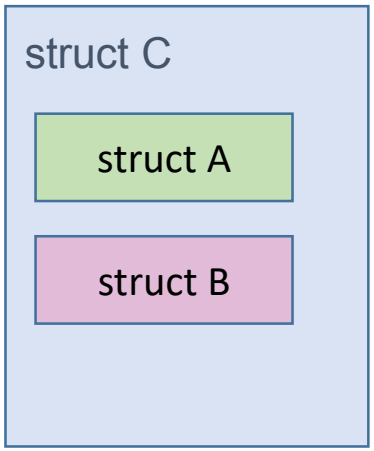
ตัวอย่างโปรแกรมอีกอัน

```c
#include <stdio.h>
#include <string.h>
int main() {
    struct food {
        char code[10];
        char description[30];
        int  price;
    } fd[4];
    printf("Please <enter> for input 4 record: \n");
    for(int i=0;i<4;i++){
        printf("**********************");
        printf("Record# %d\n",i+1);

        printf("ID:");
        scanf("%s",fd[i].code);

        printf("description:");
        scanf("%s",fd[i].description);

        printf("PRICE:");
        scanf("%d",&fd[i].price);
    }
}
```

Nest Structure

- โครงสร้างซ้อนโครงสร้าง
- ตัวอย่างเช่น โครงสร้าง C ประกอบด้วยโครงสร้าง A และ B

```
struct A {
        …
};
struct B{
        …
};
struct C {
        struct  A  var_a;
        struct  B  var_b;
};
```



Name	Age	st_addr	
		District	City

ตัวอย่างโปรแกรม

Nest Structure

```c
#include <stdio.h>
#include <string.h>
int main() {
    struct address{
       char  district[20];
       char  city[20];
    };
    struct student{
        char name[20];
        int  age;
        struct address st_addr;
    };
    struct student yr1;
    strcpy(yr1.name,"Alice");
    yr1.age = 20;
    strcpy(yr1.st_addr.district,"Muang");
    strcpy(yr1.st_addr.city,"Loei");

    printf("Name: %s\n",yr1.name);
    printf("Age : %d\n",yr1.age);
    printf("Address: %s, %s.\n",yr1.st_addr.district,
    yr1.st_addr.city);
    return 0;
}
```

Exercise

- เขียนโครงสร้างข้อมูล ดอกไม้ในประเทศไทย ประกอบด้วย รหัส , ชื่อดอกไม้, ราคา, แหล่งที่ปลูก โดยที่แหล่งที่ปลูกประกอบด้วย ตำบล และ จังหวัด
- เขียนโปรแกรมรับข้อมูลดอกไม้ 3 ชนิด และพิมพ์ข้อมูลออกมา
- ส่งโปรแกรม และ ตัวอย่างผลลัพธ์

Pointer vs. Structure

เราสามารถใช้งานสมาชิกของตัวแปรแบบโครงสร้างได้อยู่ 2 วิธี

- ใช้เครื่องหมายจุด

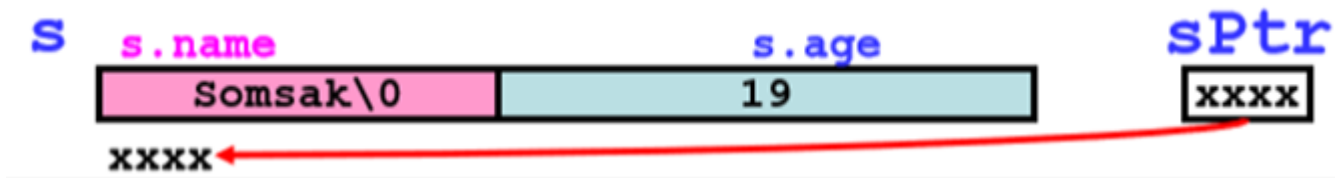
  ชื่อตัวแปรโครงสร้าง.ชื่อสมาชิก

- ใช้เครื่องหมาย **->** เมื่อเป็นตัวแปร pointer ไปยังตัวแปรแบบโครงสร้าง

  ชื่อตัวแปรแบบพอยเตอร์->ชื่อสมาชิก

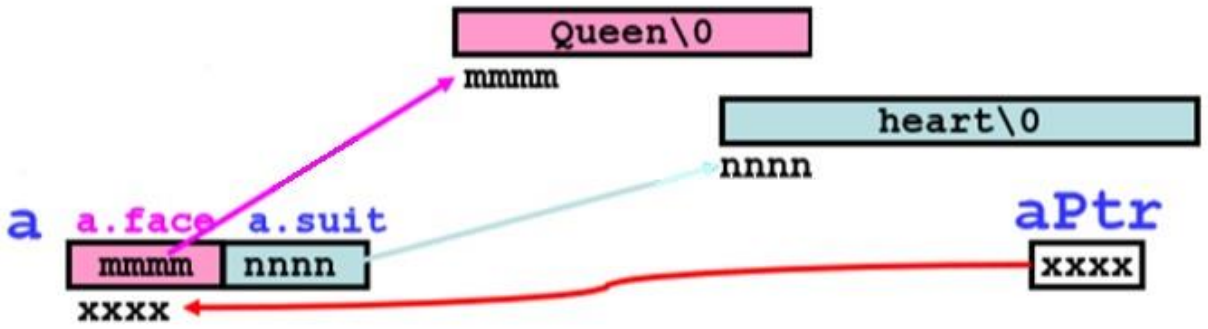
L43.c

```c
#include <stdio.h>
#include <string.h>
int main() {
    struct student {
        char name[20];
        int age;
    };
    struct student s;
    struct student *sPtr;

    strcpy(s.name, "Somsak");
    s.age = 19;
    sPtr = &s;
    printf("%s\n",s.name);
    printf("%s\n",sPtr->name);
    return 0;
}
```



L44.c

```c
#include <stdio.h>
#include <string.h>
int main() {
    struct card {
        char *face;
        char *suit;
    };
    struct card a;
    struct card *aPtr;

    a.face = "Queen";
    a.suit = "heart";
    aPtr   = &a;

    printf("%s\n",a.suit);
    printf("%s\n",aPtr->suit);
    return 0;
}
```



ตัวอย่างโปรแกรมการใช้ structure

```c
#include <stdio.h>
int main() {
    struct card {
        char *face;
        char *suit;
    };
    struct card a;
    struct card *aPtr;
    a.face = "Ace";
    a.suit = "spade";
    aPtr = &a;
    printf("%s of %s\n",a.face,a.suit );
    printf("%s of %s\n",aPtr->face, aPtr->suit);
    printf("%s of %s\n",(*aPtr).face,(*aPtr).suit);
    return 0;
}
```

Exercise

จงเขียนโปรแกรมภาษา C เก็บข้อมูลนิสิตจำนวน 5 คน โดยมีรายละเอียดดังนี้

- ข้อมูลประกอบด้วย ชื่อ อายุ
- รับข้อมูลนิสิตจาก keyboard
- ให้ใช้ตัวชี้(pointer) ที่ชี้ไปยัง ข้อมูลประเภทโครงสร้าง
- เมื่อรับข้อมูลแล้ว ให้เขียนโปรแกรมพิมพ์เฉพาะนิสิตที่มีอายุมากกว่า 20 ปี โดยแสดงชื่อ และ อายุออกจอภาพ

ตัวอย่างการเขียน struct ในภาษา C++

```
// structs.cpp
// Defines and uses a struct.
// ------------------------------------------------
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
struct Representative  // Defining struct Representative
{
  string name;         // Name of a representative.
  double sales;        // Sales per month.
};
inline void print( const Representative& v)
{
   cout << fixed << setprecision(2)
        << left  << setw(20) << v.name
        << right << setw(10) << v.sales << endl;
}
int main()
{
   Representative rita, john;
   rita.name    = "Strom, Rita";
   rita.sales  = 37000.37;
   john.name   = "Quick, John";
   john.sales = 23001.23;

   rita.sales += 1700.11;            // More Sales
   cout << "  Representative           Sales\n"
        << "-------------------------------" << endl;
   print( rita);
   print( john);
   cout << "\nTotal of sales: "
        << rita.sales + john.sales << endl;
   Representative *ptr = &john;         // Pointer ptr.
                                        // Who gets the
   if( john.sales < rita.sales)         // most sales?
     ptr = &rita;
   cout << "\nSalesman of the month: "
        << ptr->name << endl;   // Representative's name
                                // pointed to by ptr.
   return 0;
}
```

ประเภทข้อมูลที่ผู้ใช้กำหนดเอง

typedef ยอมให้ผู้ใช้นิยามประเภทของข้อมูลซึ่งเทียบเท่ากับประเภทข้อมูลที่มีอยู่แล้ว เมื่อนิยามเรียบร้อยแล้ว สามารถประกาศตัวแปรในรูปแบบของประเภทข้อมูลใหม่นี้ได้

<u>รูปแบบ</u>

typedef type new-type;

type คือ ประเภทข้อมูลที่มีอยู่แล้ว

new-type คือ ประเภทข้อมูลที่ผู้ใช้กำหนดขึ้นใหม่

ประเภทข้อมูลที่ผู้ใช้กำหนดเอง

ตัวอย่าง

```
typedef   int   gender;
```

จากตัวอย่าง ประกาศให้ gender เป็นประเภทข้อมูลที่ผู้ใช้ประกาศขึ้น โดยเทียบเท่ากับข้อมูลประเภท int ดังนั้นการประกาศตัวแปรต่อไปสามารถประกาศ ดังนี้

```
gender  male, female;     ซึ่งเท่ากับ     int male, female;
```

ตัวอย่างการใช้ typedef กับตัวแปรชนิดโครงสร้าง

```
typedef  struct  {
        member 1;
        member 2;
        .......
        member n;
}new-type;
```

new-type คือ ประเภทข้อมูลแบบ structure ที่ผู้ใช้กำหนดขึ้น
หากจะประกาศตัวแปรแบบ structure โดยใช้ประเภทข้อมูลใหม่นี้ ได้โดย

new-type struct-variable

Union

- เป็นประเภทข้อมูลแบบโครงสร้างข้อมูลเหมือน struct ที่สามารถเก็บข้อมูลต่างๆ ประเภทกันได้ที่หน่วยความจำเดียวกัน
- Union ประกอบไปด้วยสมาชิกประเภทต่างๆ
- แตกต่างจาก struct คือ union สามารถเก็บข้อมูลได้เพียงหนึ่งค่าสำหรับสมาชิกใดๆ
  ในขณะที่ struct นั้นจะสามารถเก็บข้อมูลได้พร้อมกันหมด
- ขนาดของ union นั้นมีขนาดเท่ากับสมาชิกที่ใช้หน่วยความจำมากที่สุด

Union

ตัวอย่างโปรแกรม(L47.c)

```c
#include <stdio.h>
#include <string.h>
union profile {
    char  name[20];
    int   age;
    char  grade;
} var1;

int main() {
    printf("Size of var1= %d\n", sizeof(var1));
    strcpy(var1.name,"Somsri");
    var1.age  = 10;
    var1.grade  = 'A';

    printf("Name = %s\n", var1.name);
    printf("Age  = %d\n", var1.age);
    printf("grade= %c\n", var1.grade);
    return 0;
}
```


```
Size of var1= 20
Name = A
Age  = 65
grade= A
```

ตัวอย่างการใช้ unions ในภาษา C++

```
// unions.cpp
// Defines and uses a union.
// ------------------------------------------------
#include <iostream>
using namespace std;
union WordByte
{
  private:
    unsigned short w;           // 16 bits
    unsigned char b[2];         // Two bytes: b[0], b[1]
  public:                       // Word- and byte-access:
    unsigned short& word()     { return w; }
    unsigned char&  lowByte()  { return b[0]; }
    unsigned char&  highByte(){ return b[1]; }
};

int main()
{
    WordByte wb;
    wb.word() = 256;
    cout << "\nWord:      " << (int)wb.word();
    cout << "\nLow-byte:  " << (int)wb.lowByte()
         << "\nHigh-byte: " << (int)wb.highByte()
         << endl;
    return 0;
}
```

แหล่งอ้างอิง

- สไลด์ ของ อ.วิวัฒน์ ชินนาทศิริกุล
- http://www.ce.kmitl.ac.th
- http://www.cs.science.cmu.ac.th
- https://www.cs.kku.ac.th
- http://marcuscode.com/lang/c/other-data-types