

# Structures and Unions

Assembled for 204112  
by Kittipitch Kuptavanich

## Structure

- ในภาษา C structure (หรือ record) เป็นชนิดข้อมูลเชิงซ้อน (complex data type) ที่ประกอบด้วย data type แบบพื้นฐาน นำมาประกอบกัน เพื่อใช้เก็บข้อมูลที่มีความสัมพันธ์กัน

### ตัวอย่าง 1

- ข้อมูลนักศึกษา
  - หนึ่งคนประกอบด้วย
    - รหัสนักศึกษา ชื่อ นามสกุล GPA

## Primitive Data Type - recap

Integer (จำนวนเต็ม)	Floating point (ทศนิยม)	อื่น ๆ
_bool	float	enum
char	double	.
signed char	long double	.
unsigned char		.
short		
unsigned short		
int		
unsigned int		
long		
unsigned long		
long long		
unsigned long long		

## Structure [2]

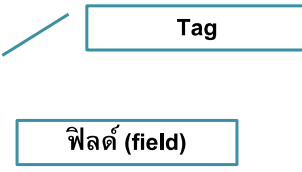
### ตัวอย่าง 2

- ข้อมูลพิกัด (coordinate) x, y บนระนาบ สองมิติ
  - หนึ่งจุดประกอบด้วย
    - ตำแหน่งในแนวแกน x
    - ตำแหน่งในแนวแกน y

## Syntax – Structure Declaration

- หนึ่งจุด (point) ประกอบด้วย
  - x (int) (สมมติมีแต่จำนวนเต็ม)
  - y (int)

```
struct point
{
    int x;
    int y;
};
```



*/\*สังเกตเครื่องหมาย ; หลังปิดปีกกา \*/*

5

## Syntax – Structure Declaration [2]

- นักศึกษาหนึ่งคนประกอบด้วย
  - รหัสนักศึกษา ชื่อ นามสกุล GPA

```
struct student
{
    char studentID[9]; // เก็บได้ 8 หลักถ้าใช้ '\0'
    char firstName[20];
    char lastName[20];
    float gpa
};
```

*/\* ข้อมูลแต่ละ field ไม่จำเป็นต้องเป็นข้อมูลชนิดเดียวกัน \*/*

6

## Syntax – Variable Declaration

- การสร้าง variable ชนิด point

แบบที่ 1:

```
struct point
{
    int x;
    int y;
};
```

```
struct point p1;
```

แบบที่ 2:

```
struct point
{
    int x;
    int y;
} p1;
```

7

## Assigning Value

Method 1:

```
struct point p1;
p1.x = 4;
p1.y = 2;
```

Method 2:

```
struct point p2 = {1,2};
```

Method 3:

```
struct point p3 = {.y = 3, .x = 8};
```

8

## Assigning Value [2]

- เราสามารถตั้ง alias (shortcut) แทนคำว่า **struct point** โดยการใช้คำสั่ง **typedef**
- Syntax  
**typedef type shortcut\_name;**
- เช่น  
**typedef struct point pointT;**

Before  
**struct point p1 = 3;**      ➡      After  
**pointT p1 = 3;**

9

## typedef

Method 1:

```
struct point {
    int x;
    int y;
};
typedef struct point pointT;
```

Method 2:

```
typedef struct point {
    int x;
    int y;
} pointT;
```

Method 3:

```
typedef struct {
    int x;
    int y;
} pointT;
```

สับตำแหน่งได้

10

## Assignment

```
pointT p1 = {1,3};
pointT p2 = p1;

p1.x = 5
printf("p1.x = %d\n",p1.x)
printf("p2.x = %d\n",p2.x)
```

11

## Summary on Declarations

- เราใช้ keyword **struct** เพื่อสร้างชนิดข้อมูลชนิดใหม่
- โดยปกติจะประกาศ **struct** ไว้นอก function หรือใน header file (.h)
  - หากประกาศใน function ใด ๆ ก็จะสามารถเรียกใช้ได้แค่ภายใน function นั้น ๆ
- เราไม่จำเป็นต้องประกาศชื่อของ **struct**
  - **struct {...} x, y, z;**
- ตัวแปรที่มีการประกาศขึ้นภายใน **struct** เรียกว่าสมาชิก (member)
- การประกาศตัวแปรให้มีชนิดข้อมูลเป็น **struct** ที่กำหนด สามารถทำได้ในลักษณะเดียวกับการประกาศตัวแปรอื่น ๆ
  - **struct point ptA;**

12

## Summary on Declarations [2]

- การให้ค่าเริ่มต้น (initialization) ทำได้โดยการระบุค่าของสมาชิกแต่ละตัว
  - `struct point ptA = {10, 20};`
- Assignment operator (เครื่องหมาย =) จะทำการคัดลอกค่าจากทุกสมาชิกของ structure
  - หากมีการดำเนินการที่เกี่ยวข้องกับ pointer ควรใช้ความระมัดระวังเป็นพิเศษ

13

## More Examples

```

struct triangle {
    struct point ptA ;
    struct point ptB ;
    struct point ptC ;
} ;

struct chain_element {
    int data ;
    struct chain_element *next;
} ;

/* สามารถมีสมาชิกเป็น pointer ไปยัง structure ที่กำลังประกาศได้ */
// สามารถมีสมาชิกเป็น structure อื่น

```

14

## Structures with Functions

```

/*structs as function arguments */
double distance(pointT p1, pointT p2)
{
    double diffX = p1.x - p2.x;
    double diffY = p1.y - p2.y;
    return sqrt((diffX * diffX) + (diffY * diffY))
}

/*structs as return value */
pointT maxDistFromOrigin(pointT p1, pointT p2)
{
    pointT origin = {0, 0};
    double dist1 = distance(p1, origin);
    double dist2 = distance(p2, origin);
    if (dist1 >= dist2)
        return p1;
    else
        return p2;
}

```

15

## Array of Structures

- Declaring arrays of `int`: `int x[10];`
  - Declaring arrays of `structure`: `struct point p[10];`
  - Initializing arrays of `int`: `int x [4]={0,20,10,2};`
  - Initializing arrays of `structure`:
    - `struct point p[] = {{1, 2}, {3, 2}, {3, 1}};`
    - `struct point p[] = {1, 2, 3, 2, 3, 1};`
- ↕
- `pointT pArray1[] = {{1, 2}, {3, 2}, {3, 1}};`
  - `pointT pArray2[] = {1, 2, 3, 2, 3, 1};`

16

# Reading Inputs into Arrays

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

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

```
typedef struct {
    char name[80];
    char address[150];
    date birthdate;
} record;
```

```
record readinput();
void writeoutput(record id);
```

```
int main()
{
    int i, n;
    record id[100];

    printf("How many records?");
    scanf("%d", &n);

    for (i = 0; i < n; ++i) {
        id[i] = readinput(i);
    }

    for (i = 0; i < n; ++i) {
        printf("\n\nRecord %d\n", i);
        writeoutput(id[i]);
    }

    return 0;
}
```

17

# Reading Inputs into Arrays [2]

```
record readinput()
{
    record id;
    printf("\n Name:");
    scanf(" %[^\\n]", id.name);
    printf(" Address:");
    scanf(" %[^\\n]", id.address);
    printf(" Birth date(dd/mm/yyyy):");
    scanf("%d/%d/%d\\n", &id.birthdate.day,
        &id.birthdate.month,
        &id.birthdate.year );
    return (id);
}
```

อ่าน input จนถึง '\\n' แต่ไม่  
เขียน '\\n' ลงใน char array

18

# Reading Inputs into Arrays [3]

```
void writeoutput(record id)
{
    printf("Name: %s\\n", id.name);
    printf("Address: %s\\n", id.address);
    printf("Birth date: %d/%d/%d\\n", id.birthdate.day,
        id.birthdate.month,
        id.birthdate.year );
    return;
}
```

19

# Structure Pointers

- เนื่องจากการคัดลอกค่าใน structure ต้องทำที่ละสมาชิก สำหรับ structure ขนาดใหญ่การ pass by reference (ใช้ pointer) จะทำให้โปรแกรมทำงานได้อย่างมีประสิทธิภาพมากกว่า

```
void function(struct point * pp);
struct point pt;
function(&pt);
```

20

## Structure Pointers [2]

- การเข้าถึงสมาชิกของ structure จาก pointer ทำได้โดยการ ใช้เครื่องหมาย '->'

```
struct point p = {10, 20};
struct point *pp = &p;
```

```
pp->x = 10;           // changes p.x
int y = pp->y;        // same as y = p.y
```

- Other ways to access structure members ?

```
struct point p = {10, 20};
struct point *pp = &p;
```

```
(*pp).x = 10;        // changes p.x
int y = (*pp).y;      // same as y = p.y
```

Why is the () is required?

21

## Size of Structures [2]

```
struct {
    char c;           // 1 byte
    /* padding */     // padding of 3 bytes
                        // so that int will be 4 byte aligned
    int i;             // 4 bytes
    short s;           // 2 byte
    /* padding */     // padding of 2 byte
};                   // 12 bytes total
```

- และจะมีการ padding หลังสมาชิกตัวสุดท้ายเพื่อให้ขนาดของ struct หารขนาดของสมาชิกที่ใหญ่ที่สุดลงตัว (ทำให้เกิด alignment ใน กรณี array of struct)
- การเรียงประเภทของสมาชิกจากเล็กไปใหญ่ จะทำให้ต้องการ padding byte น้อยลง และประหยัดเนื้อที่มากขึ้น

23

## Size of Structures

- structure มีขนาดมากกว่า หรือเท่ากับผลรวมของขนาดของสมาชิกทุกตัวใน structure
- เมื่อ compile จะมีการ padding ระหว่างสมาชิกที่มี byte alignment ที่ต่างกัน (เพื่อการอ่านข้อมูลที่รวดเร็ว)  
(char - 1, short - 2, int - 4, long - 4, float - 4, และ double - 8 byte aligned)

```
struct {
    char c;           // 1 byte
    /* padding */     // padding of 3 bytes
                        // so that i will be 4 byte aligned
    int i;             // 4 bytes
    short s;           // 2 byte
    /* padding */     // padding of 2 byte
};                   // 12 bytes total
```

22

## Unions

- Union เป็นตัวแปรที่สามารถเก็บข้อมูลหลายๆ ชนิด / ที่มีขนาดที่แตกต่างกันลงในพื้นที่หน่วยความจำเดียวกันเช่น

```
union data {
    int idata;
    float fdata;
    char *sdata;
} d1, d2, d3;
```

```
d1.idata = 10;
d2.fdata = 3.14F;
d3.sdata = "hello world";
```

- วัตถุประสงค์หลักของ union คือเพื่อการประหยัดพื้นที่หน่วยความจำ หากมีการจัดเก็บชนิดข้อมูลที่ต่างกัน ในเวลาที่ต่างกัน (non-overlapping time)

24

## Unions [2]

- ขนาดของ union จะเท่ากับขนาดของสมาชิกที่ใหญ่ที่สุด
- **Warning: compiler** จะไม่ทำการ **check** ว่าข้อมูลที่ถูกอ่านอยู่ในชนิดที่ถูกต้องหรือไม่ เช่น

```
union data d;
d.idata = 10;
float f = d.fdata; /* ได้ค่าขยะ */
```

- แนวทางแก้ปัญหา: ใช้ **struct** แทนเพื่อแยกตัวแปรออกจากกัน

## Reference

- <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-087-practical-programming-in-c-january-iap-2010/>
- <http://www.catb.org/esr/structure-packing/>