

# Foundations of C Programming (Structured Programming)

- Structure (结构), Enumeration(枚举)

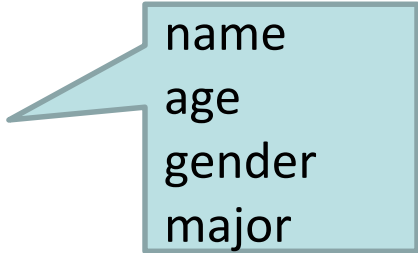
# Outline

- Structure definition
- Structure variables declaration
- Structure assignment
- Array of structures
- Enumeration

# Basic Idea

- To represent one item, we can declare single **variable**
  - E.g., `int age;`
- To represent several items of the same type, we can declare an **array**
  - E.g., `int age[100];`
- To represent several items of same or different types, together for a particular object, we can declare a **structure**
  - E.g., we have 100 students, **each student's information** includes **name, age, gender, major**, how can we do?

A student

A light blue rectangular box with a pointer on its left side, pointing towards the text 'A student'. Inside the box, the words 'name', 'age', 'gender', and 'major' are listed vertically.

name  
age  
gender  
major

# An Example - birthday

- A birthday consists of 3 parts: year, month and day.
- We can define them in this way

```
int year = 2007;
```

```
int month = 11;
```

```
int day = 13;
```

- These 3 variables are logically related, we cannot see that the above declarations are for one date with the above declarations.
- It would be better if they can be grouped **together**.
- The **structure** in C helps.

# An Example - date

A structure called **date**

```
struct date {  
    int year;        // 1st member variable  
    int month;       // 2nd member variable  
    int day;         // 3rd member variable  
};                  // don't forget the semicolon !
```

- **date** now is a new type that can be used like int, char, ...
- **date** has three members (成员).

```
int i;    //i is an integer  
struct date d; //d is a date with three members
```

# struct and array

- A structure is a collection of **related data items** of same or different types, usually **contribute to one object** (关于同一个对象的若干信息)
  - E.g.,
    - **Student**: student id, name, major, gender, start year, ...
    - **Bank account**: account number, name, currency, balance, ...
    - **Address book**: name, address, telephone number, ...
- Indicated by keyword **struct**
- An **array** contains only **data of the same type**, usually the data in an array do not have coherent relationship.

# struct Variable Declarations

- There are three ways to declare a variable of struct type

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

**Preferred**

`birthday` is a variable,  
it is not part of  
structure declaration

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

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

**Preferred**

# Accessing the Members of a Structure

- A member of a structure is accessed by specifying the **variable name**, followed by a **period(英文句号)**, and then the **member name**

```
struct date today;  
  
scanf("%d %d", &today.month, &today.day);  
if(today.month == 1 && today.day == 1)  
    printf("Happy new year!");
```

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



# Initialization of structure variable: An Example

- An Employee record consists of three elements:
  - Age of integer
  - Name of 20 characters
  - Salary of float

```
struct employee
{
    int age;
    char name[20];
    float salary;
};

.....
struct employee emp1 = {25, "Dave", 2500}, emp2;

strcpy(emp2.name, "John");
emp2.age = 30;
emp2.salary = 2000;
```

# struct-to-struct Assignment

```
struct studentRecord{
    char name[15];
    int id;
    char dept[5];
    char gender;
};
struct studentRecord student1, student2;

strcpy(student1.name, "Tom Hanks");
student1.id = 12345;
strcpy(student1.dept, "COMP");
student1.gender = 'M';

student2 = student1; →
```

```
strcpy(student2.name, student1.name);
student2.id = student1.id;
strcpy(student2.dept, student1.dept);
student2.gender = student1.gender;
```

# Class Exercises

- Can you define a struct called **point** (点) which has x coordinate (坐标) and y coordinate
- If a point is at (10, 3), how to express that?

# Nested Structures

- We can nest (嵌套) structures inside structures.
- E.g.,

```
struct point{  
    double x, y;  
};  
  
struct line{  
    struct point p1, p2;  
};  
  
struct triangle{  
    struct point p1, p2, p3;  
};  
  
struct point p;  
struct line l;  
struct triangle t;
```

Access the members:

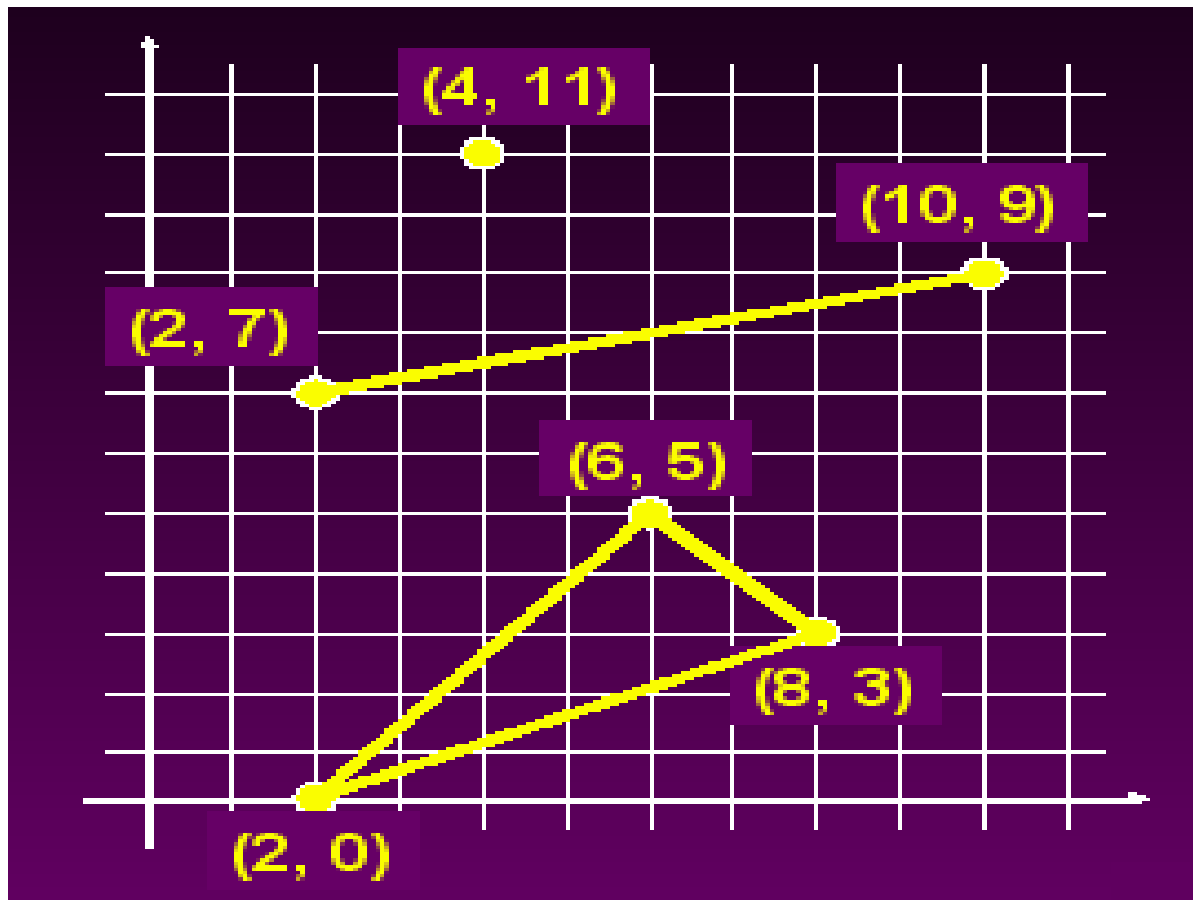
p.x } A point p  
p.y }

l.p1.x } A line l  
l.p1.y  
l.p2.x  
l.p2.y }

t.p1.x } A triangle t  
t.p1.y  
t.p2.x  
t.p2.y  
t.p3.x  
t.p3.y }

# Class Exercise

- Write code to give the following assignments.



# Structure in A Function

- Parameters in a function can be of structure type
- The return value can be of structure type
- Usually the definition of structure is put outside of function, so all the functions can use this definition

# Structure in A Function: An Example

```
struct date {  
    int year;  
    int month;  
    int day;  
};  
struct date nextDay(struct date today);  
int main() {  
    struct date today, next;  
    scanf("%d%d%d", &today.year, &today.month, &today.day);  
    next = nextDay(today); //next=tomorrow  
    printf("%d %d %d", next.year, next.month, next.day);  
    return 0;  
}  
struct date nextDay(struct date today)  
{  
    struct date tomorrow;  
    // Put code here to calculate the date of tomorrow  
    return tomorrow;  
}
```

today: 2021.11.17  
today.year=2021;  
today.month=11;  
today.day=17;

tomorrow: 2021.11.18  
tomorrow.year=2021;  
tomorrow.month=11;  
tomorrow.day=18;

**Tips: It is better to put structure definition outside any function so all functions can use it.**

# Array of Structures

- If we have 100 students, each student's information includes name, id, how can we do?

```
typedef struct {  
    char name[15];  
    int id;  
} studentRecord;  
  
int main() {  
    studentRecord student[100];  
    int i;  
    for (i = 0; i < 100; i++)  
        scanf("%s%d", student[i].name, &student[i].id);  
    // Put code here to handle the inputted info  
}
```

Tips: If the number of students is unknown, you can use scanf to read the number of students first.



# Enumeration(枚举)

- An enumeration type (also called enum) is a data type that consists of integral constants.
- Use the keyword “enum”
  - Format: enum *enumName* {const1, const2, ..... };
- Examples

Define an enumeration type *week*

```
#include <stdio.h>
enum week{Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};
int main(){
    enum week day;
    day = Sunday;
    printf("%d\n", day);
    day = Monday;
    printf("%d\n", day);
}
```

Declare a variable *day* of the type *week*

Assign a value to *day*

 Console program output

```
0
1
Press any key to continue...
```

# Summary

- Struct can be used to group data of different types together.
- Struct can be defined in different ways.
- Struct makes the processing of information easier.