Structured Programming

- Structure

Dr Donglong Chen

With Thanks to Dr. Xin Feng and Dr. Haipeng Guo

Outline

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

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 different types, we can declare a structure
  - E.g., we have 100 students, each student's information includes name, age, gender, major, how can we do?

Example #1 - date

- 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
- It would be better if they can be grouped together.
- The structure in C helps.

Example #1 - 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.

struct and array

- A structure is a collection of related data items of same or different types, usually contribute to one object
  - 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;
```

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

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

date birthday;
```

Accessing the Members of a Structure

- A member of a structure is accessed by specifying the variable name, followed by a period, 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!");
```

Example #2 – Person Record

- A Person record consists of THREE elements:
  - Name of 50 characters
  - Address of 100 characters
  - Salary of float
- There are two ways to create the structure variables.

```
struct Person
{
    char name[50];
    char address[100];
    float salary;
};

int main()
{
    struct Person person1, person2, p[20];
    return 0;
}
```

Example #2 – Person Record

- The second way is to create the variables directly within the struct.

```
struct Person
{
    char name[50];
    int citNo;
    float salary;
} person1, person2, p[20];
```

Suppose, you want to access the salary of person2 or salary of p[1]. Here's how you can do it.

```
person2.salary
```

```
P[1].salary
```

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;
```

Structure in A Function

- A structure parameter value can be passed to a function
- A structure can be returned

```
struct date nextDay(struct date today)
{
        struct date tomorrow;
        // code to calculate the date of tomorrow
        ......
        return tomorrow;
}
```

Example #3 – Employee

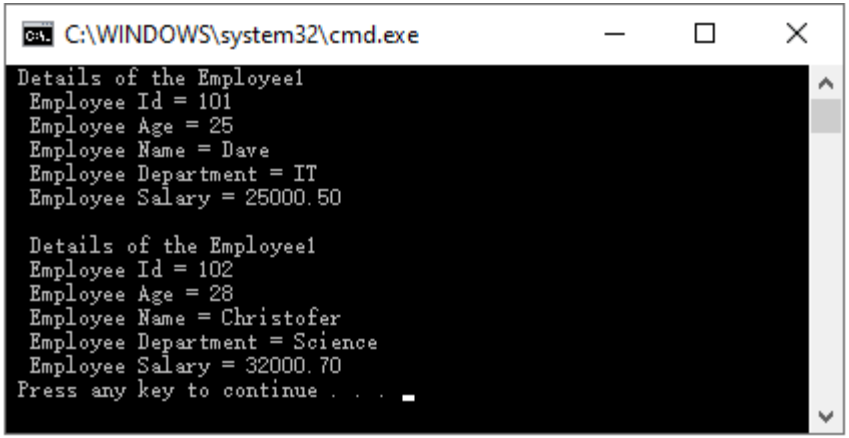
- An Employee record consists of THREE elements:
  - Age of integer
  - Name of 20 characters
  - Salary of float

```
struct Employee
{
  int age;

  char name[50];

  float salary;
};

//Creating the Structure variable in main() function
struct Employee emp1, emp2;

//Two ways of assigning values to the structure elements
emp1 = {25, “Dave”, 25000};
emp2.age = 30;
…
```

Class Exercise #1 Add two distances

Complete the program statements between the comments in the right-hand side to get the results below.

Output




```
/* Structure Example - Employee */
#include <stdio.h>
#include <string.h>

struct Employee
{
  int Empolyee_ID;
  int age;
  char Name[50];
  char Department[20];
  float Salary;
};

int main()
{
  struct Employee emp1 = { 101, 25, "Dave", "IT", 25000.50 };
  struct Employee emp2;

     emp2.Empolyee_ID = 102;
     emp2.age = 28;
     //--please provides your codes here, start--//

     //--please provides your codes here, end--//

  printf(" Details of the Employee1 \n " );
  printf(" Employee Id = %d \n ", emp1.Empolyee_ID );
  printf(" Employee Age = %d \n ", emp1.age );
  printf(" Employee Name = %s \n ", emp1.Name );
  printf(" Employee Department = %s \n ", emp1.Department );
  printf(" Employee Salary = %.2f \n\n ", emp1.Salary );

  printf(" Details of the Employee1 \n " );
  printf(" Employee Id = %d \n ", emp2.Empolyee_ID );
  printf(" Employee Age = %d \n ", emp2.age );
  printf(" Employee Name = %s \n ", emp2.Name );
  printf(" Employee Department = %s \n ", emp2.Department );
  printf(" Employee Salary = %.2f \n ", emp2.Salary );

  return 0;
}
```

Class Exercise #2 Add two distances

Complete the program statements between the comments in the right-hand side to get the results below.

Output

```
1st distance
Enter feet: 12
Enter inch: 7.9
2nd distance
Enter feet: 2
Enter inch: 9.8
Sum of distances = 15'-5.7"
```

```c
// Program to add two distances (feet-inch)
#include <stdio.h>
struct Distance
{
    int feet;
    float inch;
} dist1, dist2, sum;

int main()
{
    printf("1st distance\n");
    printf("Enter feet: ");
    scanf("%d", &dist1.feet);

    printf("Enter inch: ");
    scanf("%f", &dist1.inch);
    printf("2nd distance\n");

    printf("Enter feet: ");
    scanf("%d", &dist2.feet);

    printf("Enter inch: ");
    scanf("%f", &dist2.inch);

    // adding feet
    sum.feet = dist1.feet + dist2.feet;
    // adding inches
    sum.inch = dist1.inch + dist2.inch;

    // changing to feet if inch is greater than 12
    while (sum.inch >= 12)
    {
       //--please provides your codes here, start--//
       //--hint: 1 feet = 12 inch--//
       //--please provides your codes here, end--//
    }

    printf("Sum of distances = %d\'-%.1f\"",
            sum.feet, sum.inch);
    return 0;
}
```

Class Exercise #3

- Can you define a struct called point which has *x* coordinate and *y* coordinate
- Give three ways of declaration
- If the point is at (10, 3), how to assign that?

Nested Structures

- We can nest structures inside structures.
- E.g.,

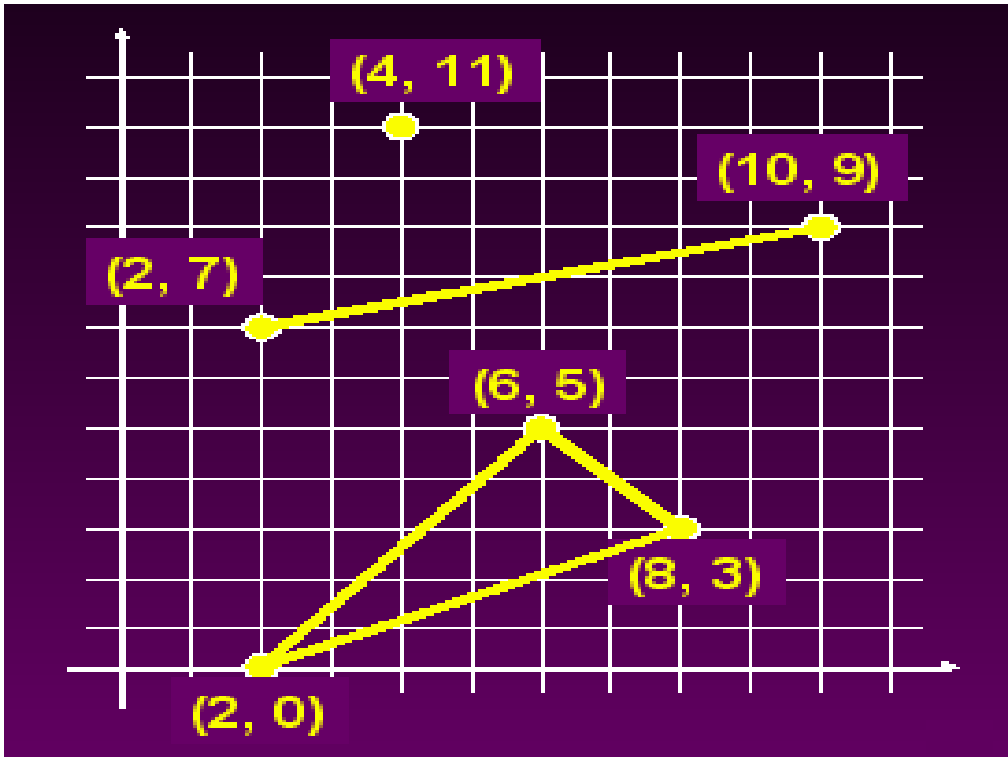
```
struct point{
  double x, y;
};
struct point p;

struct line{
  struct point p1, p2;
};
struct line l;

struct triangle{
  struct point p1, p2, p3;
};
struct triangle t;
```

Assessing the members:

```
p.x
p.y

l.p1.x
l.p1.y
l.p2.x
l.p2.y
```

Class Exercise #4

1. Write code to declare the structures for:
   - Point
   - Line
   - Triangle
2. Write code to calculate
   - Length of the Line



Array of Structures

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

```
struct studentRecord student[100];

strcpy(student[98].name, "Tom Hanks");
student[98].id = 12345;
strcpy(student[98].dept, "COMP");
student[98].gender = 'M';

student[0] = student[98];
```

Class Exercise #5

- Can we initialize a variable of structure when we declare that variable? If yes. Give an example.
- Can we do the same thing if this time is an array of structures? Says student[3]. Illustrate how can we do that.

Summary

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

Please submit your class exercise into iSpace on the day the lecture is given.