Jiangxi University of Science and Technology

Chapter 12 Structures

Lecture1201 Structures

Objectives

- 12.1 Single Structures 单一结构体
- 12.2 Arrays of Structures 结构体数组
- 12.3 Passing and Returning Structures 传递和返回结构体
- 12.4 Unions (Optional) 共用体
- 12.5 Common Programming and Compiler Errors

Introduction

- **structure**
  - Each data item listed in Figure 12.1 is an entity by itself, called a **data field**
  - Together, all the data fields form a **single unit** called a **record**
  - In C, **a record** is referred to as a **structure**

Name:
Type:
Location in Dungeon:
Strength Factor:
Intelligence Factor:
Type of Armor:

Figure 12.1 Typical components of a video game character

Introduction

- Structure
  - A structure's **form** consists of the symbolic names, data types, and arrangement of individual data fields in the record
  - The structure's **contents** consist of the actual data stored in the symbolic names

Introduction

- The form and contents of a structure

Name: Golgar
Type: Monster
Location in Dungeon: G7
Strength Factor: 78
Intelligence Factor: 15
Type of Armor: Chain Mail

Figure 12.2 The form and contents of a structure

12.1 Single Structures

➢ Structure definition in C:

- **struct**
- {
- **int** month;
- **int** day;
- **int** year;
- }birth;
- The three data items are the **members** of the structure
- Assigning actual data values to the data items of a structure is called populating the structure

12.1 Single Structures

- Program 12.1

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      struct
5.      {    int month;    int day;    int year;  } birth;
6.      birth.month = 12;
7.      birth.day = 28;
8.      birth.year = 1987;
9.      printf("My birth date is %d/%d/%d\n",birth. month, birth.day, birth.year % 100);
10.     return 0;
11. }
```

Name	Value
sizeof(birth)	12
birth	{month=12 day=28 year=1987 }
month	12
day	28
year	1987

SPACING OF A STRUCTURE DEFINITION IS NOT RIGID

12.1 Single Structures

- **Definition of** Structure **variable**
  - **Multiple variables** can be defined in one statement
  - **struct** {**int** month; **int** day; **int** year;} **birth**, **current**;
  - Common to list the form of the structure with **no following variable names**
  - The list of structure members must be preceded by a user-selected **structure type name**
  - **struct Date**{**int** month; **int** day; **int** year;};

12.1 Single Structures

By convention the first letter of user-selected structure type names is uppercase

- Program 12.2

```
#include <stdio.h>
struct Date {  int month;  int day;  int year;};
int main(){
    struct Date birth;
    birth.month = 12;
    birth.day = 28;
    birth.year = 1987;
    printf("My birth date is %d/%d/%d\n", birth.month,birth.day,birth.year % 100);
    return 0;
}
```

12.1 Single Structures

➢Initialization of structures

- **struct Date** birth = {12, 28, 1987};
- **Structure members can be of any data type**
- **struct PayRecord**
- {
- **char** name[20]; **int** idNum;
- **double** regRate; **double** otRate;
- };
- **struct PayRecord** employee = {"H. Price", 12387, 15.89, 25.50};

12.1 Single Structures

- **Individual members** can be **arrays** and **structures**
  - **struct**
  - {
  - **char** name[20];
  - **struct Date** birth;
  - } person;
  - Example: **person.name[4]**

12.2 Arrays of Structures

Employee number	Employee name	Employee pay rate
32479	Abrams, B.	6.72
33623	Bohm, P.	7.54
34145	Donaldson, S.	5.56
35987	Ernst, T.	5.43
36203	Gwodz, K.	8.72
36417	Hanson, H.	7.64
37634	Monroe, G.	5.29
38321	Price, S.	9.67
39435	Robbins, L.	8.50
39567	Williams, B.	7.20

Figure 12.3 A list of employee data

12.2 Arrays of Structures

	Employee Number	Employee Name	Employee Pay Rate
1st Structure →	32479	Abrams, B.	6.72
2nd Structure →	33623	Bohm, P.	7.54
3rd Structure →	34145	Donaldson, S.	5.56
4th Structure →	35987	Ernst, T.	5.43
5th Structure →	36203	Gwodz, K.	8.72
6th Structure →	36417	Hanson, H.	7.64
7th Structure →	37634	Monroe, G.	5.29
8th Structure →	38321	Price, S.	9.67
9th Structure →	39435	Robbins, L.	8.50
10th Structure →	39567	Williams, B.	7.20

Figure 12.4 A list of records

12.2 Arrays of Structures

- Program 12.2

```
1.  #include <stdio.h>
2.  #define NUMRECS 5
3.  struct PayRecord {int id;char name[20];double rate;};
4.  int main(){
5.     struct PayRecord employee[NUMRECS]={{32479, "Abrams, B.",6.72},{33623, "Bohm, P.",
       7.54}, {34145, "Donaldson, S.",5.56}, {35987,"Ernst,T.", 5.43}, {36203, "Gwodz, K.", 8.72}};
6.     for (int i = 0; i < NUMRECS; i++)
7.         printf("%d %-20s %4.2f\n", employee[i].id, employee[i]. name, employee[i].rate);
8.     return 0;
9.  }
```

Name	Value	Type
employee	0x0058fd9c {id=32479 name=0x0058fda0 "Abrams, B." rate=6.71999999999	PayRecord
[0]	{id=32479 name=0x0058fda0 "Abrams, B." rate=6.7199999999999998 }	PayRecord
[1]	{id=33623 name=0x0058fdc0 "Bohm, P." rate=7.5400000000000000 }	PayRecord
[2]	{id=34145 name=0x0058fde0 "Donaldson, S." rate=5.5599999999999996 }	PayRecord
[3]	{id=35987 name=0x0058fe00 "Ernst,T." rate=5.4299999999999997 }	PayRecord
[4]	{id=36203 name=0x0058fe20 "Gwodz, K." rate=8.7200000000000006 }	PayRecord

12.2 Arrays of Structures

- Without explicit initializers, the numeric elements of both static and external arrays or structures are **initialized to 0** (or **nulls**)

- An inferior alternative to an array of structures is parallel arrays
  - **Parallel arrays** are two or more arrays, where each array has the same number of elements and the elements in each array are directly related by their position in the arrays
  - They are rarely used any more

12.3 Passing and Returning Structures

- **Individual structure members may be passed to a function** in the same manner as any scalar variable
  - display(emp.idNum)
  - calcPay(emp.payRate, emp.hours);
- On most compilers, complete copies of all members of a structure can also be passed to a function by including the name of the structure as an argument to the called function
  - calcNet(emp);

12.3 Passing and Returning Structures

- Program 12.4

```
#include <stdio.h>
struct Employee {
        int idNum;
        double payRate;
        double hours;
};
double calcNet (struct Employee temp)
{return( temp.payRate * temp.hours); }
int main()
{
  struct Employee emp={6787, 8.93, 40.5};
   double netPay;
   netPay = calcNet(emp);
    printf("The net pay of employee %d is $%6.2f\n", emp.idNum,netPay);
   return 0;
}
```

12.3 Passing and Returning Structures

- A structure can be passed by reference

```
1. double calcNet (struct Employee *pt)
2. {return((*pt).payRate*(*pt).hours); }
3. calcNet(&emp);
```

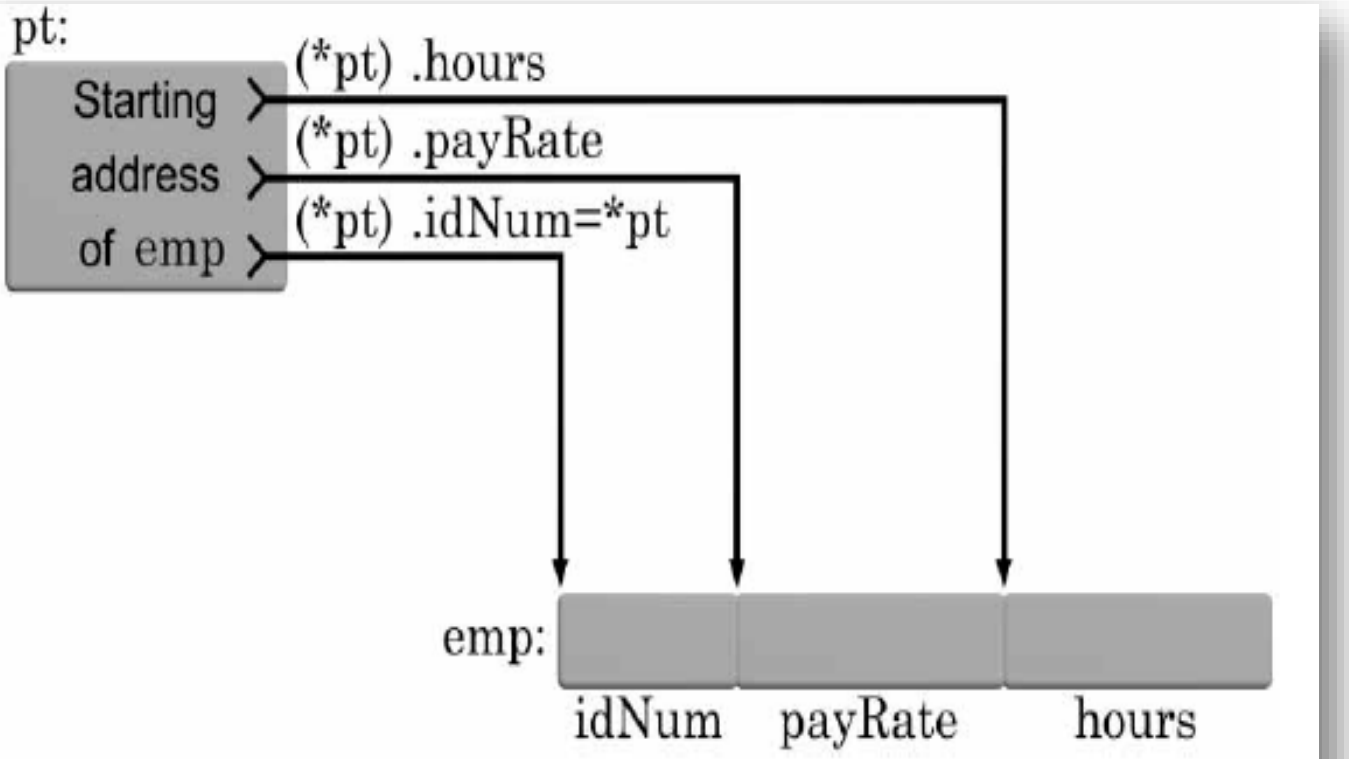

Figure 12.5 A pointer can be used to access structure members

12.3 Passing and Returning Structures

- Program 12.5

```
1.  #include <stdio.h>
2.  struct Employee {
3.     int idNum;
4.     double payRate;
5.     double hours;
6.  };
7.  double calcNet (struct Employee *pt)
8.  {
9.  return( pt->payRate * pt->hours);
10. }
```

12.3 Passing and Returning Structures

- ++ and -- can be applied to structures
  - ++**pt->hours**
  - (**pt**++)->hours
  - (++**pt**)->hours

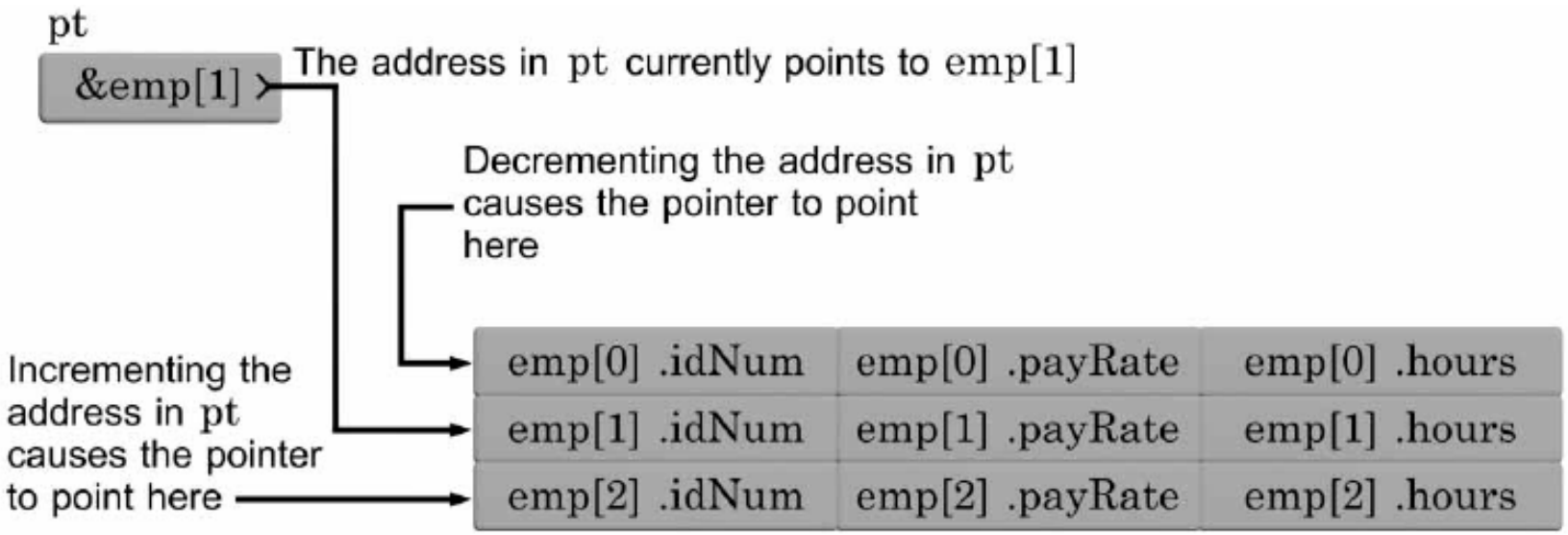

Figure 12.6 Changing pointer addresses

12.4 Returning Structures

Program 12.6

```
#include <stdio.h>
struct Employee /* declare a global structure type */
{
  int idNum;
  double payRate;
  double hours;
};

struct Employee getValues(); /* function prototype */

```

12.4 Returning Structures

```
11  int main()
12  {
13    struct Employee emp;
14
15    emp = getValues();
16    printf("\nThe employee id number is %d\n", emp.idNum);
17    printf("The employee pay rate is $%5.2f\n", emp.payRate);
18    printf("The employee hours are %5.2f\n", emp.hours);
19
20    return 0;
21  }
22
23  struct Employee getValues()
24  {
25    struct Employee newemp;
26
27    newemp.idNum = 6789;
28    newemp.payRate = 16.25;
29    newemp.hours = 38.0;
30
31    return (newemp);
32  }
```

Reference

- https://www.codesdope.com/blog/article/int-main-vs-void-main-vs-int-mainvoid-in-c-c/