



福州大学至诚学院  
FUZHOU UNIVERSITY ZHICHENG COLLEGE

# 高级语言程序设计 (C语言与数据结构)

杨雄

83789047@qq.com





# 第九章 结构体

9.1 结构体类型的定义

9.2 结构体变量的定义

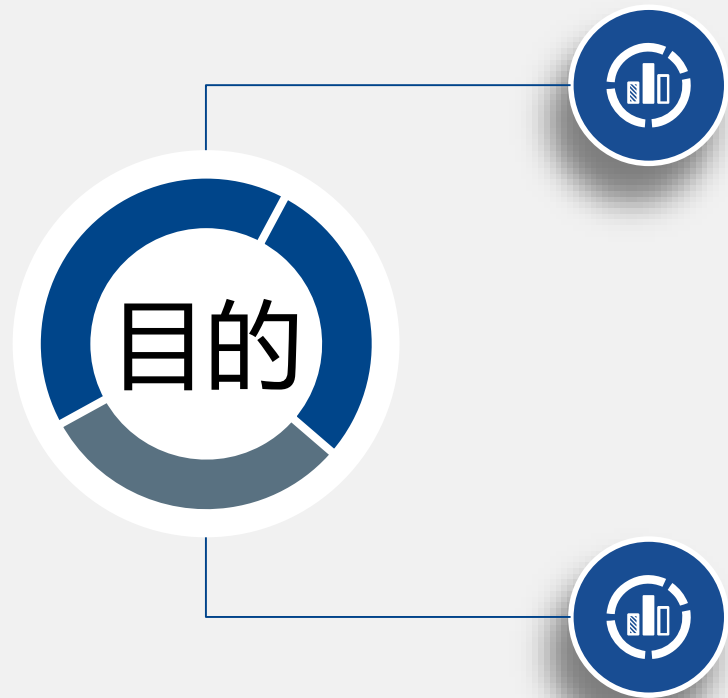
9.3 结构体变量的初始化和引用

9.4 结构体数组

9.5 指向结构体变量的指针变量



# 学习目的

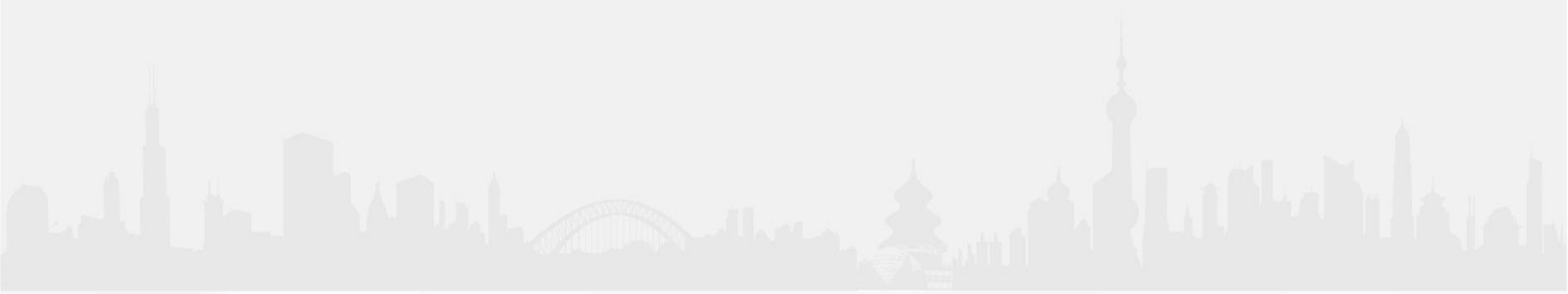


**理解结构体类型变量的定义和存储形式**

**掌握结构体类型与成员的引用**

# Part.1

## 9.1 结构体类型的定义



## 9.1 结构体类型的定义

- 结构体类型定义的格式:

**struct 结构体名**

**{ 类型说明符 成员名1 ;**

**类型说明符 成员名2 ;**

**...**

**类型说明符 成员名n ;**

**} ;**

- **struct 结构体名**: 结构体类型名, 作用等同于int、float等关键字, 用于定义结构体变量。

## 9.1 结构体类型的定义

例:

```
struct student  —————→  结构体类型名
{
    long int num ;
    char name[8] , sex ;
    float score[4];
};  //分号是必需的
```

结构student的成员

即定义了一个名为 struct student 的结构体类型, 成员有:长整型的num, 字符型name和sex, 浮点型的score。

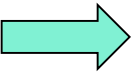


## 9.1 结构体类型的定义

例:

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

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



即定义了一个名为 **struct date** 的结构体类型, 成员有: 整型的 **year**, **month** 和 **day**。

## 9.1 结构体类型的定义

### 说明：

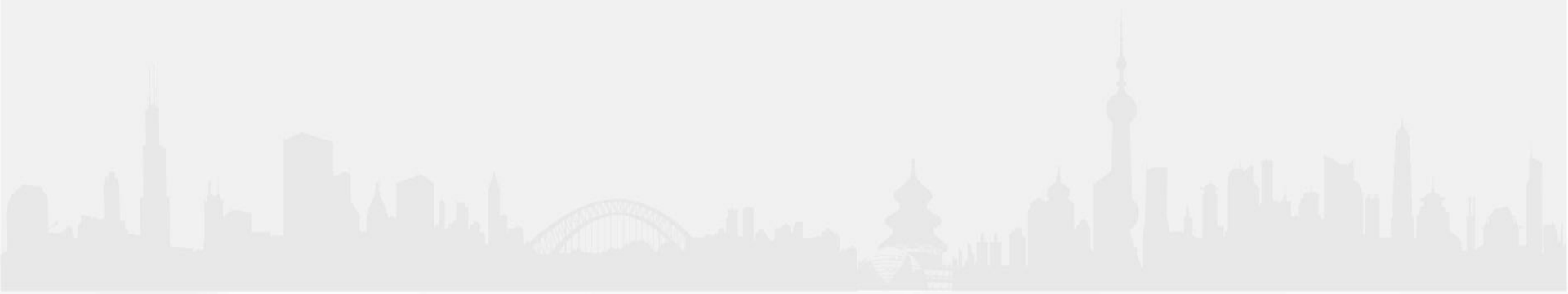
- 同一结构体的成员**不能同名**
- **不同结构的成员**可以同名, 不互相冲突, 结构中的成员名可以和程序中的变量名同名。

```
struct date { int year, month, day ; };  
  
struct Book  
{ char title[50], writer[20], publisher[50];  
  int year, month ;  
};  
  
int year, month, day;
```



# Part.2

## 9.2 结构体变量的定义



## 9.2 结构体变量的定义

- 结构体变量的定义

结构体类型的定义只是**创建了新的数据类型**，并不能保留内存空间，不能存放具体的数据。

为了在程序中使用结构类型的数据，**应在说明了结构体类型之后**，再**定义该结构类型的变量**—结构变量，以存放该结构类型的数据。

**有三种方式定义结构变量**

## 9.2 结构体变量的定义

### 1. 先声明结构类型，再定义结构变量。

- 格式： **结构体类型名 结构体变量名**；

```
struct student
{ char name[8] , sex ;
  float score[2];
};

void main()
{ struct student stu1,stu2; //定义结构变量,分配空间
  } 结构体类型名                //使用结构变量
```

- 存储形式：

name[8]	sex	score[0]	score[1]
8字节	1字节	4字节	4字节

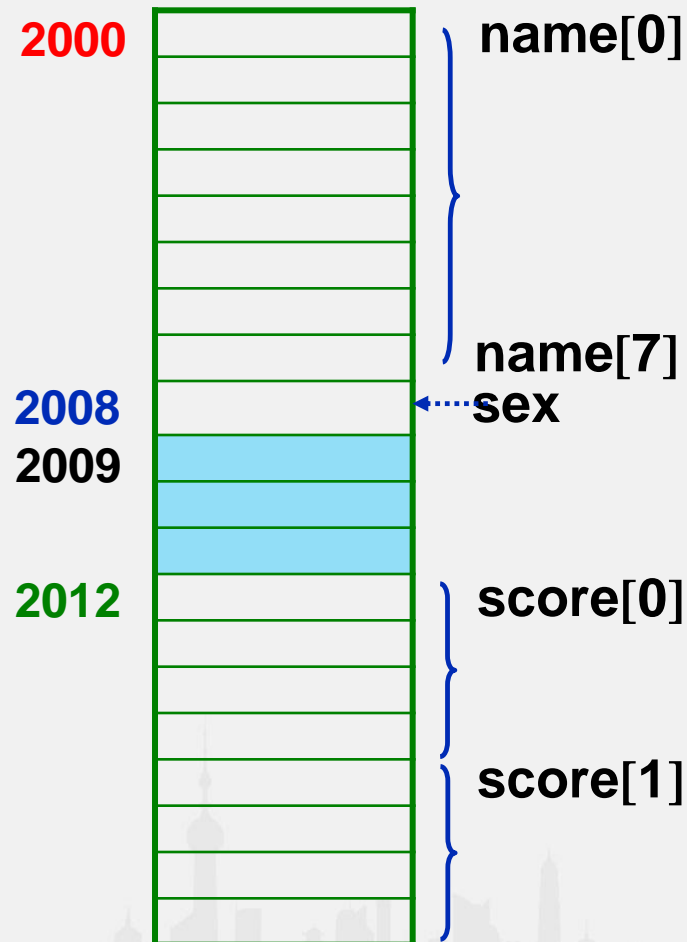
• stu1 存储形式:

name[8]	sex	score[0]	score[1]
8B	1B	4B	4B

```
struct student
{ char name[8], sex ;
  float score[2];
};
struct student stu1;
printf("%d",sizeof(student));20B
```

- 结构各成员存放的**起始地址**相对于结构变量起始地址的**偏移量**必须为该成员类型所占用**字节数的倍数**。
- 确保结构变量占用内存大小为结构体字节边界数的倍数(该结构中占用**最大空间的类型所占用的字节数**)。

用户数据区内存



例:

```
struct student  
{ char c ;  
  double d ;  
  int i ;  
};
```

**struct student m2;**

总空间大小:

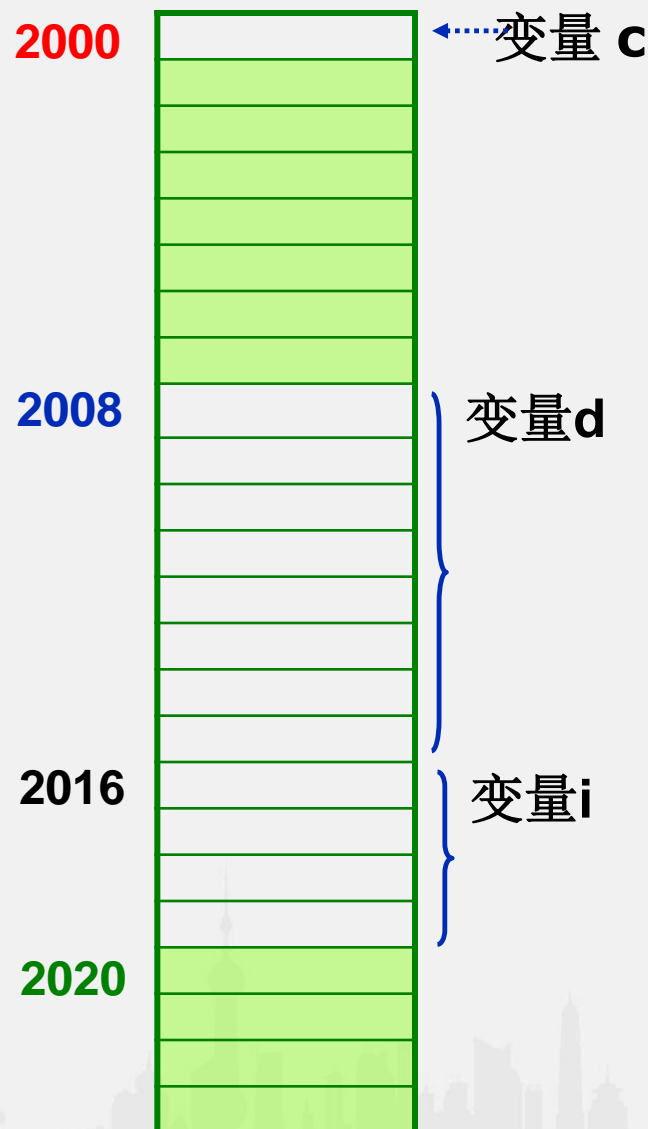
$$1+7+8+4=20\text{B}$$

不是结构体字节边界数的倍数,

$(\text{sizeof}(\text{double})) = 8$

需填充4个字节, **m2共占24B。**

用户数据区内存



## 9.2 结构体变量的定义

### 2. 在定义结构类型的同时定义结构变量

例: **struct student** —————→ 结构类型名

```
{ char name[8] , sex ;  
  int  score[2] ;  
  int  year, month, day ;  
} stu1, stu2; —————→ 结构变量名表
```

存储形式:

name[8]	sex	score[0]	score[1]	year	month	day
8字节	1字节	4字节	4字节	4字节	4字节	4字节



## 9.2 结构体变量的定义

### 3. 对于无名结构, 直接定义结构变量。

例:

```
struct  
{ int num;  
  char name[8], sex ;  
  int age ;  
} stu1, stu2 ;
```

没有结构名,  
无法再次使用

即定义了两个结构变量stu1和stu2

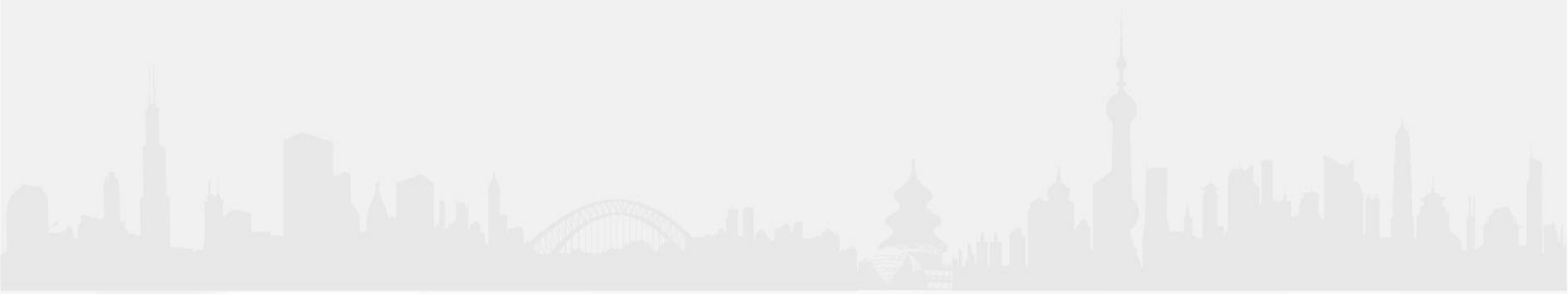
## 9.2 结构体变量的定义

### 说明：

- 结构体**类型**与结构体**变量**是不同的概念，注意区分。
  - 编译时，系统只对变量分配存贮空间，而类型则不分配。
  - 系统可以对变量赋值、存取、运算，而类型则不能。
- 结构中的成员也可以是一个结构变量，即结构的嵌套。

# Part.3

## 9.3 结构体变量的初始化



## 9.3.1 结构体变量的初始化

- 在定义结构变量的同时对其成员变量赋初值

- 格式:

**struct 结构体名 结构变量名 = {初始化表};**

- 特点:

对结构变量进行初始化时，系统是按每个成员在结构体中的**顺序一一对应**赋初值的，要求初始数据类型与相应成员变量的**类型一致**。

## 9.3.1 结构体变量的初始化

- 给全部成员赋初值

```
struct student  
{ long num ;  
    char name[8] , sex;   int score[2];  
}stu1={100, “张三” , ‘M’ , 98, 95};
```

num	name[8]	sex	score[0]	score[1]
100	张三	M	98	95

## 9.3.1 结构体变量的初始化

### • 给部分成员赋初值

- 若只对部分成员初始化, 只能给前面的若干成员赋值, 而不允许跳过前面的成员给后面的成员赋值。

**struct student stu1={100, “张三” };**

num	name[8]	sex	score[0]	score[1]
100	张三	\0	0	0

✗ **struct student stu1={100, “张三” , ,98, 95};**



## 9.3.2 结构体变量的引用

- 引用形式:

**结构体变量名.成员名**

其中：“.”为结构成员运算符，1级运算符，左结合性。

例：**stu1.num**=20033129;

表示对结构变量stu1中的num成员的引用

## 9.3.2 结构体变量的引用

### 引用规则：

#### 1) 结构变量的输入输出：

**不能对结构变量整体进行操作, 必须逐个对成员进行输入输出操作。**

`scanf( “%ld%s%c%d%d” , &stu1 );` ✗

若输入stu1中的num和name成员, 应写成:

`scanf( “%ld %s” , &stu1.num, stu1.name);`

## 9.3.2 结构体变量的引用

### 引用规则：

- 2) 若结构成员是数值型数组，则对数组成员的引用，应为对该数组元素的引用。

如： `stu1.score[0]`

`stu1.score[1]`

## 9.3.2 结构体变量的引用

### 引用规则：

3) 成员变量可以象普通变量一样进行各种运算操作  
例：

**stu1.num++;**           //结构成员**num**作自增运算

**stu1.sex=getchar( );**       //赋值运算

**stu1.sex=' F' ;**

**stu1.score[0]>=60**           //关系运算

## 9.3.2 结构体变量的引用

### 引用规则：

#### 4) 结构变量的整体赋值

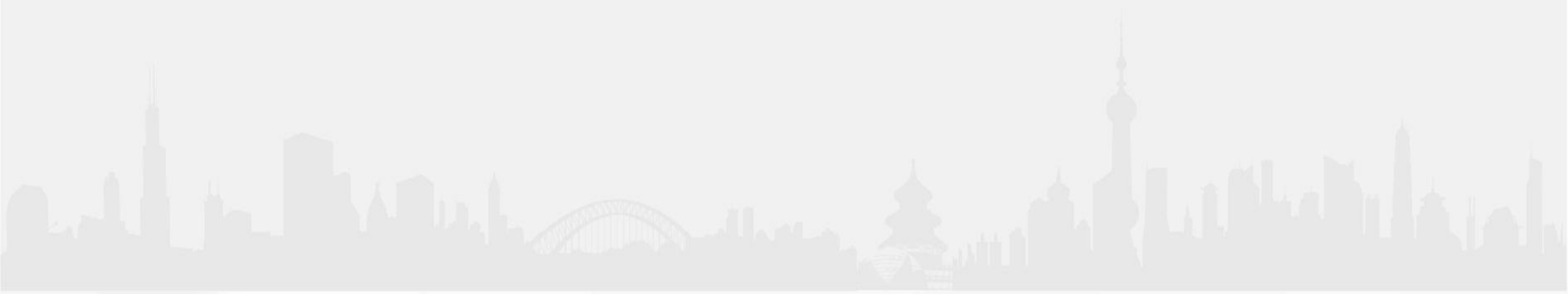
可以将一个结构变量作为一个整体赋给另一个同类型的结构变量。

```
struct student
{ long num;
  char name[10], sex ; float score[4];
} stu2, stu1={ ... };
```

- **stu2=stu1 ;**
- 两个**不同结构类型**名的变量不允许相互赋值


# Part.4

## 9.4 结构体数组





## 9.4 结构体数组

 一个结构变量只能存放一组数据(如一个学生的学号、姓名、性别等的数据), 若需存放多个学生的数据, 显然很不方便, 此时应使用**结构数组**。

在结构数组中的每一个数组元素都是一个结构变量。

## 9.4 结构体数组

### • 结构数组的定义

```
struct student    //定义学生结构体类型
{ long num;
  char name[8], sex ;
  float score[3];
};
struct student stu[3];
```

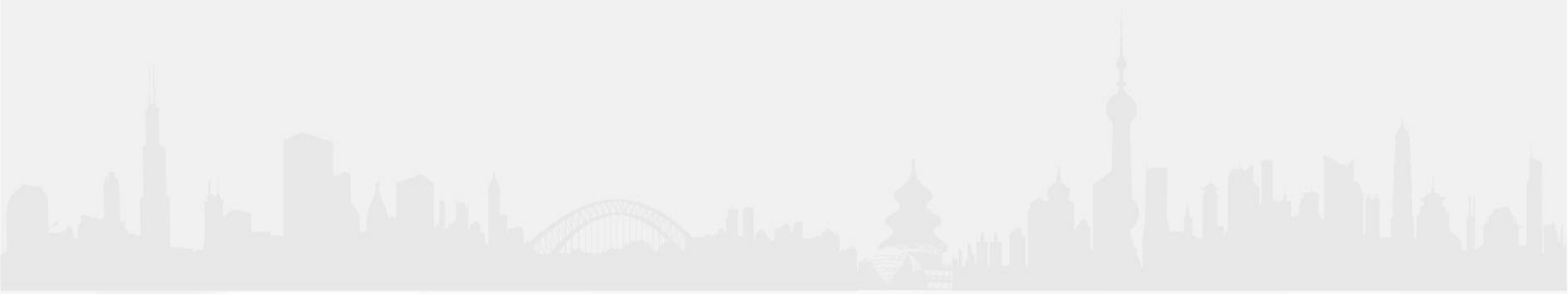
即定义了一个类型为  
**student**的结构数组**stu**,  
该数组有**3**个元素。

其数组元素各成员的引用形式为:

```
stu [0].name, stu [0].sex, stu [0].score [i] ;
```

# Part.5

## 9.5 指向结构体变量的指针变量



## 9.5 指向结构体的指针

一个结构体变量的指针就是该变量所占据的内存段的起始地址。

可以设一个指针变量，用来指向一个结构体变量，此时该指针变量的值是结构体变量的起始地址。

指针变量也可以用来指向结构体数组中的元素。

## 9.5 指向结构体的指针

### 以下3种形式等价：

- ① 结构体变量. 成员名
- ② (\*p) . 成员名
- ③ p->成员名

### 请分析以下几种运算：

- p->n 得到 p 指向的结构体变量中的成员 n 的值。
- p->n++ 得到 p 指向的结构体变量中的成员 n 的值，用完该值后使它加 1。
- ++p->n 得到 p 指向的结构体变量中的成员 n 的值加 1，然后再使用它。

## 9.5 指向结构体的指针

### 例 指向结构体

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

```
struct student
```

```
{int num;char na
```

```
struct student stu
```

```
10102, "Zhang Fun", M, 19}, {10104, "WangMing",  
'F', 20} } ;
```

```
void main()
```

```
{ struct student *p;
```

```
printf(" No.    Name    sex    age \n ") ;
```

```
for ( p =str; p <str+ 3 ; p++ )
```

```
printf("%5d %-20s %2c %4d \n", p->num, p->name, p->  
>sex, p->age);
```

```
}
```

### 运行结果:

N o .	N a m e	s e x	a g e
1 0 1 0 1	LiLin	M	18
1 0 1 0 2	Zhang Fun	M	19
1 0 1 0 4	WangMing	F	20



## 9.5 指向结构体的指针

### 注意：

(1) 如果  $p$  的初值为  $stu$ ，即指向第一个元素，则  $p$  加 1 后  $p$  就指向下一个元素。例如：

- $(++p) \rightarrow num$  先使  $p$  自加 1，然后得到它指向的元素中的  $num$  成员值（即 10102）。
- $(p++) \rightarrow num$  先得到  $p \rightarrow num$  的值（即 10101），然后使  $p$  自加 1，指向  $stu[1]$ 。

请注意以上二者的不同。

## 9.5 指向结构体的指针

### 注意：

(2) 程序已定义了 `p` 是一个指向 `struct student` 类型数据的指针变量，它用来指向一个 `struct student` 类型的数据，不应用来指向 `stu` 数组元素中的某一成员。

例如： `p = s t u [1] . n a m e ;`



## 9.5 指向结构体的指针

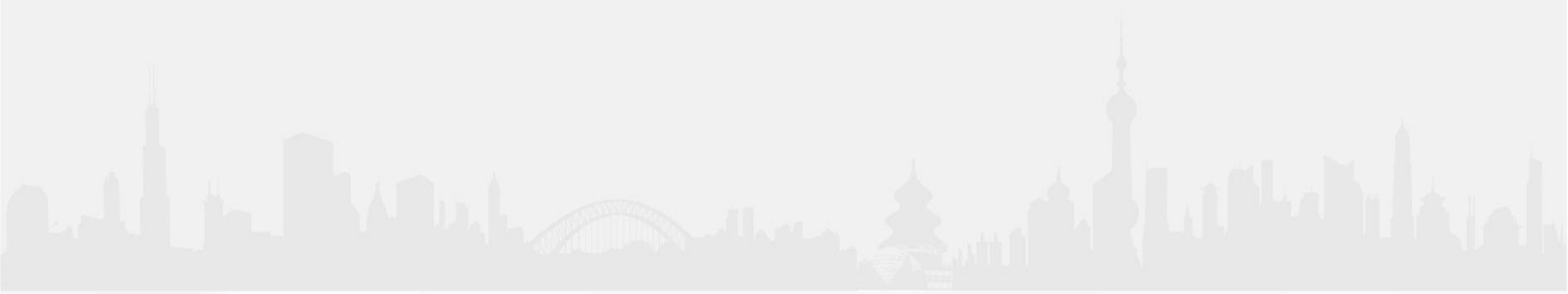
### ◆ 用结构体变量和指向结构体的指针作函数参数

将一个结构体变量的值传递给另一个函数，有3个方法：

- (1) 用结构体变量的**成员**作参数。
- (2) 用**结构体变量**作实参。
- (3) 用指向结构体变量（或数组）的**指针**作实参，将结构体变量（或数组）的地址传给形参。

# Part.6

## 总结



# 总 结

## 结构体类型和变量**定义**:

```
➤ struct 结构体类型名{  
    类型名 成员变量名;  
    类型名 成员变量名;  
    .....  
}变量名;
```

```
➤ 或 struct 结构体类型名{  
    类型名 成员变量名;  
    类型名 成员变量名;  
    .....  
};
```

```
struct 结构体类型名 变量名;
```

# 总 结

- 结构体变量**初始化**

- 定义时给出变量各个成员的值。

- struct student{

- char num[12] ;

- char name[30] ;

- int age;

- }stu1={ "20121514101" , "zhangxiaohong" , 20} ,

- stu2 , \*p=&stu1;

# 总结

## – 结构体变量使用

### • 访问成员

– `stu1.age`   `p->name`   `(*p).age`  
`stu2=stu1`

### • 输入输出（按成员逐个输入输出，不能整体输入输出）

– `scanf( "%s%s%d" , p->num, p->name, &p->age);`  
– `scanf( "( "%s%s%d" , stu1.num, stu1.name, &stu1.age);`



# 总 结

## 结构体类型的数组

- 定义与初始化：

```
struct student{  
    char num[12] ;  
    char name[30] ;  
    int    age;  
}stu[30]={ { "20121514101" , "zhangxiaohong" ,  
20},{ "20121514102" , "liming" , 19}, .....};
```

- 数组元素的使用和结构体变量使用一样

– stu[i].name      stu[i].age

- 注意:数组元素不能整体输入输出

## 指向结构体数组的指针

### 3种形式:

- ① 结构体变量. 成员名
- ② (\* p ). 成员名
- ③ p ->成员名

其中->称为指向运算符。

# 总 结

## 结构体作为函数参数

- 结构体变量成员作为函数参数
- 结构体变量作为函数参数
- 指向结构体变量的指针作为函数参数