

第九章 结构体

蒋玉茹

结构体初步

问题A?

请输入1名学生的姓名、性别、年龄、身高、体重、住址信息，然后格式化输出。

程序A

问题B

请输入10名学生的姓名、性别、年龄、身高、体重、住址信息，然后格式化输出。

程序B

使用结构体的三个步骤

声明结构体
类型

定义结构体
类型变量

结构体变量
的引用

结构体使用VS函数的使用

声明结构体类型

定义结构体类型
变量

结构体变量的引
用

函数定义

函数声明

函数调用

使用结构体的三个步骤

```
graph LR; A[声明结构体类型] --> B[定义结构体类型变量]; B --> C[结构体变量的引用];
```

声明结构体
类型

定义结构体
类型变量

结构体变量
的引用

结构体名：结构体类型的名称。遵循标识符规定。

```
struct 结构体名
{
    类型1  成员1;
    类型2  成员2;
    ... ..
    类型n  成员n;
};
```

结构体有若干数据成员，分别属于各自的数据类型，结构体成员名同样遵循标识符规定，它属于特定的结构体变量（对象），名字可以与程序中其它变量或标识符同名。

分号不可少！！！！

结构体类型声明举例

(1) 声明一个包含如下学生信息的结构体类型

num	name	sex	age	score	addr
10010	Li Fun	M	18	87.5	Beijing

struct student

```
{  
    int num;  
    char name[20];  
    char sex;  
    int age;  
    float score;  
    char addr[30];  
};
```

说明:

- 1、struct student是结构体类型名，struct是关键字。
- 2、该结构体类型由6个成员组成，分别属于不同的数据类型，分号“;”不能省略。
- 3、在声明了结构体类型后，可以定义结构体变量。

结构体声明练习

- ▶ 个人数据: 包含姓名、性别、年龄、身高、体重、住址:

```
struct person
{
    char name [20] ;           /*姓名*/
    char sex;                  /*性别*/
    int age;                   /*年龄*/
    float height;              /*身高*/
    float weight;              /*体重*/
    char addr [50] ;           /*住址*/
};
```

结构体声明练习

▶ 平面上的点

```
struct point2
{
    float x; /*横坐标*/
    float y; /*纵坐标*/
};
```

结构体声明练习

► 空间中的点

```
struct point3
{
    float  x; /*X坐标*/
    float  y; /*Y坐标*/
    float  z; /*Z坐标*/
};
```

结构体声明练习

- ▶ 日期，包括年、月、日

```
struct date
{
    int year; /*年*/
    int month; /*月*/
    int day; /*日*/
};
```

结构体声明练习：可以嵌套

▶ 职工信息结构体类型

```
struct person
{
    char name[20];        /*姓名*/
    char address[40];     /*地址*/
    float salary;         /*工资*/
    float cost;           /*扣款*/
    struct date hiredate; /*聘任日期*/
};
```


使用结构体的三个步骤

声明结构体
类型

定义结构体
类型变量

结构体变量
的引用

先声明结
构体类型,
再定义变
量名



定义结构
体类型变
量的方法



直接定义
结构体类
型变量



在声明结
构体类型
的同时定
义变量

(一) 先声明结构体类型，再定义变量名

格式

struct 结构体类型名 结构体变量名;

举例

```
struct student
{
    int num;
    char name[20];
    char sex;
    int age;
    float score;
    char addr[30];
};
struct student student1;
```

前提：结构体类型
struct 结构体类型名
已经被声明

使用结构体的三个步骤

声明结构体
类型

定义结构体
类型变量

结构体变量
的引用

三、结构体变量的引用

“.”运算符是成员运算符。

格式 结构体变量名. 成员名

举例

```
student1.num=10010;  
scanf("%s",student1.name);  
student1.age++;
```

说明

1、成员本身又是结构体类型时，子成员的访问使用成员运算符逐级访问：

例：student1.

2、同一种类型（整体赋值，成员赋值）

例：student1 = student2;

```
scanf("%d,%s,%c,%d,%f,%s",&student1);  
printf("%d,%s,%c,%d,%f,%s",student1);
```

3、不允许将一个结构体变量整体输入/输出。

练习

- ▶ 有结构体类型如下，请编写程序，完成一个人的信息的输入和输出

```
struct person
{
    char name [20] ;           /*姓名*/
    char sex;                  /*性别*/
    int age;                   /*年龄*/
    float height;              /*身高*/
    float weight;              /*体重*/
    char addr [50] ;           /*住址*/
};
```

练习

- ▶ 职工信息包括姓名、地址、工资、扣款、聘任日期，请定义该结构体类型，并编写程序完成一个职工信息的录入和输出。

```
struct person
{
    char name[20];        /*姓名*/
    char address[40];     /*地址*/
    float salary;         /*工资*/
    float cost;           /*扣款*/
    struct date hiredate; /*聘任日期*/
};
```

练习

求某同学上学期6门课程的总成绩与平均成绩。

思路分析：

可以先构建一个结构体，包含学生的姓名、6门课的成绩、以及总成绩及平均成绩。

```
struct student
{
    char name[10];
    float score[6];
    float total, average;
};
```

然后在程序中输入姓名及各科成绩后即可进行运算，运算结果存放到total和average两个成员变量中。

程序如下：

```
#include <stdio.h>
void main ( )
{
    int i;
    struct student
    {
        char name [10] ;
        float score [6] ;
        float total, average;
    };
    struct student stu;
    scanf ( " %s" , stu.name) ;
    for (i=0; i<6;i++)
        scan ( " %f" , &stu.score [i] ) ;
    stu.total=0;
    for (i=0; i<6;i++)
        stu.total+=stu.score [i] ;
    stu.average=stu.total/6;
    printf ( " %s的总成绩=%.2f\n平均成绩=%.2f",
        stu.name, stu.total, stu.average) ;
}
```

输入数据：

CHEN 80 86 79 98 88 72

运行结果：

CHEN的总成绩=503.00

平均成绩= 83.83

结构体数组

结构体数组

概念 结构体数组指数组元素的类型为结构体类型的数组。

定义 与定义结构体变量一样，有如下三种方法：

- 1、先声明结构体类型，再定义结构体数组
- 2、在声明结构体类型的同时定义结构体数组
- 3、直接定义结构体数组

```
struct student{  
    long int num;  
    char name[20];  
    char sex;  
    char addr[20];  
};  
struct student stu[30];
```

练习

- ▶ 利用前面定义的某个结构体，比如职工信息，完成结构体数组内容的输入与输出

练习

- ▶ 候选人得票的统计。设有三个候选人，每次输入一个得票的候选人的名字，要求最后输出各人得票结果。

练习

- 定义一个结构体类型 `struct student`，包含学生的学号、姓名和成绩三项基本信息。分别读入3个学生的信息并按照表格形式显示输出。

输入三个学生的信息（学号、姓名和成绩）：

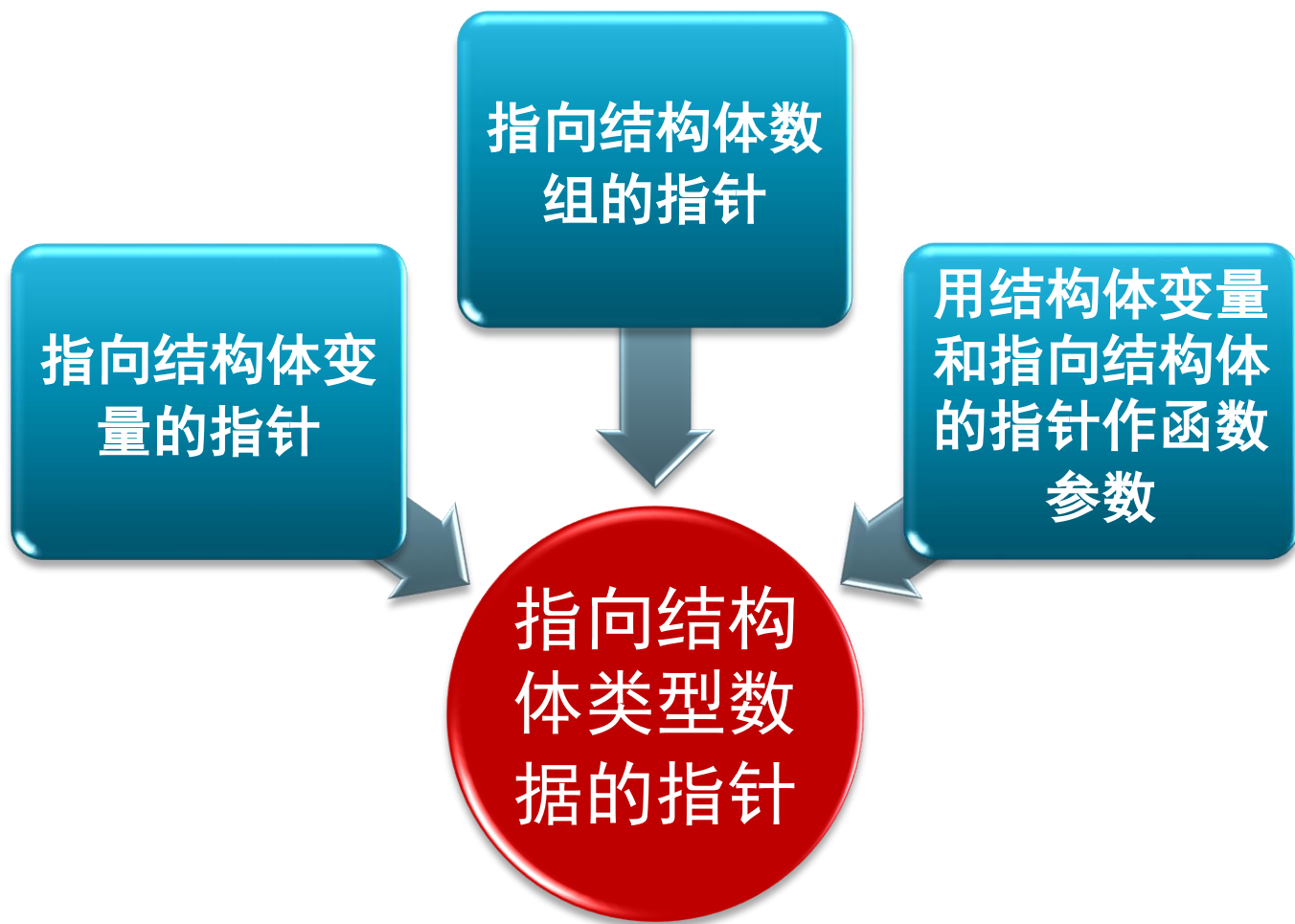
102 zhang 85

103 li 80

104 wang 76

No.	Name	Score
102	zhang	85
103	li	80
104	wang	76

第二讲



指向结构体变量的指针

指向结构体变量的指针

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。可以定义一个指针变量，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。

指向结构体变量的指针

定义格式 `struct` **类型名** ***指针变量名;**

例如: `struct student *pstu, stu;`
`pstu=&stu;`

定义了一个指针变量`pstu`,并使指针指向结构体变量`stu`。

`*pstu` 表示指针`pstu`所指示的结构体变量(对象) `stu`,因而,其成员表示为 `(*pstu).name` , `(*pstu).age`。

为方便起见,C语言定义了结构体的指针变量引用结构体成员的特殊形式:

指针变量名->成员名

“->”为指向运算符

上面示例 `(*pstu).name`, `(*pstu).age`。可以等价表示为:

`pstu->name`, `pstu->age`

比较结构体成员的几种引用方式

```
void main()
{ struct student {    long int num;
                     char sex;
                     float score; } stu_1 , *p;

  p=&stu_1;
  stu_1.num=200301;
  stu_1.sex='M';
  stu_1.score=89.5;
  printf("No.:%ld\nsex:%c\nscore:%f\n",
         stu_1.num,stu_1.sex,stu_1.score);
  printf("\nNo.:%ld\nsex:%c\nscore:%f\n",
         (*p).num,(*p).sex,(*p).score);
  printf("\nNo.:%ld\nsex:%c\nscore:%f\n",
         p->num,p->sex,p->score);
}
```

总结：结构体成员变量引用方式

【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

【1】 【2】 为常用形式

指向结构体数组的指针

练习：添加代码

```
#include <stdio.h>
struct student
{
    int num;
    char name[20];
    char sex;
    int age;
};
```

```
int main()
{
    struct student *p;
    struct student stu[3];
    p=stu;
    //输入、输出3个学生的数据

    return 0;
}
```

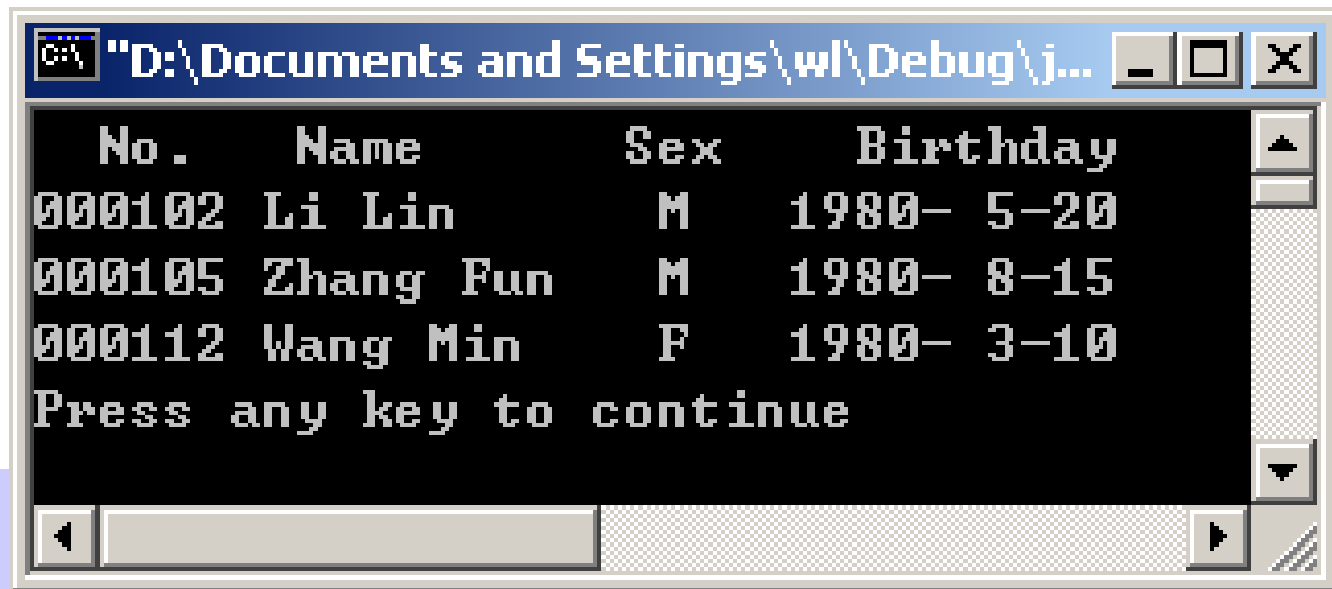
答案

```
for(i=0;i<3;i++)
{
    scanf("%d",&p->num);
    scanf("%s",p->name);
    getchar();
    scanf("%c",&p->sex);
    scanf("%d",&p->age);
    p++;
}
p=stu;
for(i=0;i<3;i++)
{
    printf("%12d%8s%4c%4d",p->num,p->name,p->sex,p-
>age);
    p++;
}
```


练习

```
/*struct.h*/
struct date
{
    int year;
    int month;
    int day;
};

struct std_info
{
    char no[20];    /*学号*/
    char name[10]; /*姓名*/
    char sex;       /*性别*/
    struct date birthday; /*出生日期*/
};
```



The screenshot shows a Windows command prompt window with the title bar "D:\Documents and Settings\wl\Debug\j...". The window contains a table of student data with four columns: No., Name, Sex, and Birthday. The data is as follows:

No.	Name	Sex	Birthday
000102	Li Lin	M	1980- 5-20
000105	Zhang Fun	M	1980- 8-15
000112	Wang Min	F	1980- 3-10

Below the table, the text "Press any key to continue" is displayed. The window has a scrollbar on the right and a status bar at the bottom.

用结构体变量和指向结构体的指针作函数参数

练习

- ▶ 定义一个函数，函数的功能是计算一个学生的平均分；
- ▶ 学生的信息包括学号、姓名、5门课程成绩、平均分、总分

引导

方法A：函数的定义

```
struct student func(struct student stu)
{
    //求stu中成绩的总和和平均值

    return stu;
}
```

引导

方法B：函数的定义

```
void func(struct student *stu)
{
    //求*stu中成绩的总和和平均值

}
```

练习

- ▶ 定义一个函数，函数的功能是计算n个学生的平均分， $1 \leq n \leq 1000$ ；
- ▶ 学生的信息包括学号、姓名、5门课程成绩、平均分、总分

引导

方法A：函数的定义

```
void func(struct student stu[])  
{  
    //求stu数组中成绩的总和和平均值  
  
}
```

引导

方法B：函数的定义

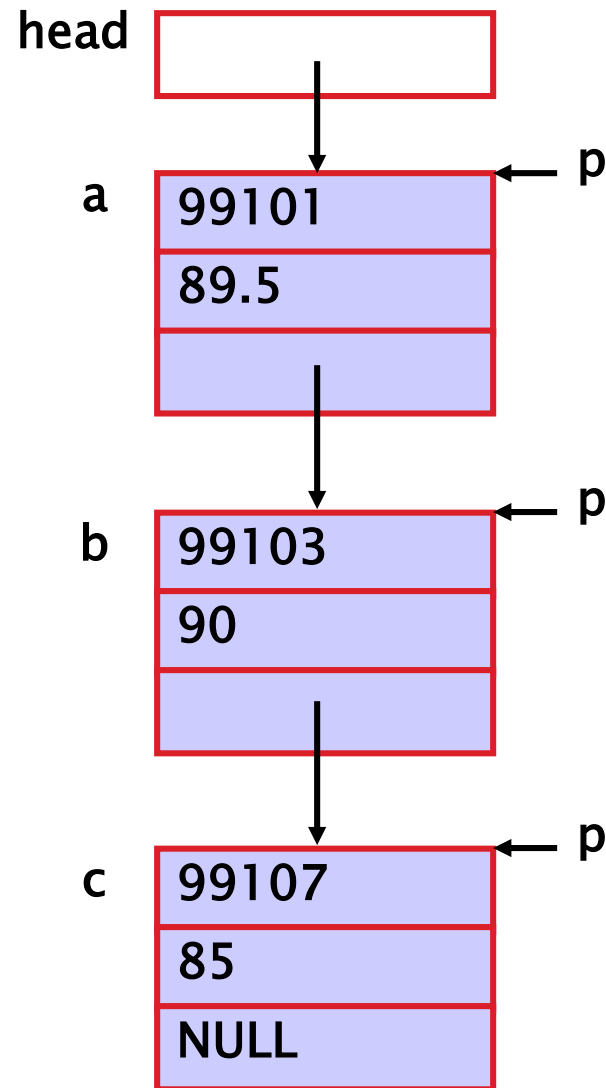
```
void func(struct student *stu)
{
    //求stu指向的数组中成绩的总和和平均值

}
```


链表

简单链表

```
#include <stdio.h>
struct student{
    long num;
    float score;
    struct student *next;
}
void main()
{
    struct student a,b,c,*head,*p;
    a.num=99101;a.score=89.5;
    b.num=99103;b.score=90;
    c.num=99107;c.score=85;
    head=&a;a.next=&b;b.next=&c;c.next=NULL;
    p=head;
    do{
        printf("%ld %5.1f\n",p->num,p->score);
        p=p->next;
    }while(p!=NULL);
}
```



动态操纵内存函数之一

▶ **#include <malloc.h>**

□ **void *malloc(unsigned int size);**

作用：在内存的动态存储区中分配一个长度为**size**的连续空间。返回值为指向分配域起始地址的指针。分配不成功时返回**NULL**。

简单链表2

怎样构造3个节点的链表?

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

struct student{
    long num;
    float score;
    struct student
    *next;
};

int main()
{
    struct student
    *head,*p;
```

```
p=malloc(sizeof(struct student));
p->num=1000;
p->score=77;
p->next=NULL;

head=p;

do{
    printf("%ld %5.1f\n",p->num,p-
    >score);
    p=p->next;
}while(p!=NULL);
return 0;
}
```

方案A

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

struct student{
    long num;
    float score;
    struct student *next;
};

int main()
{
    struct student *head,*p,*p1,*p2,*p3;
    p1=malloc(sizeof(struct student));
    p1->num=1001;
    p1->score=77;
    p1->next=NULL;

    head=p1;
```

```
p2=malloc(sizeof(struct student));
p2->num=1002;
p2->score=88;
p2->next=NULL;

p1->next=p2;

p3=malloc(sizeof(struct student));
p3->num=1003;
p3->score=99;
p3->next=NULL;

p2->next=p3;

p=head;
do{
    printf("%ld %5.1f\n",p->num,p->score);
    p=p->next;
}while(p!=NULL);

return 0;
}
```

方案B

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

struct student{
    long num;
    float score;
    struct student *next;
};

int main()
{
    struct student *head,*p,*pa[3];
    int i;
```

```
    for(i=0;i<3;i++)
    {
        pa[i]=malloc(sizeof(struct
        student));
        scanf("%d",&pa[i]->num);
        scanf("%f",&pa[i]->score);
        pa[i]->next=NULL;
    }
    head=pa[0];
    pa[0]->next=pa[1];
    pa[1]->next=pa[2];

    p=head;
    do{
        printf("%ld %5.1f\n",p->num,p->score);
        p=p->next;
    }while(p!=NULL);

    return 0;
}
```

方案C

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

struct student{
    long num;
    float score;
    struct student *next;
};

int main()
{
    struct student *head,*p,*pa[3];
    int i;
    p=malloc(sizeof(struct
student));
    scanf("%d",&p->num);
    scanf("%f",&p->score);
    p->next=NULL;
    head=p;
```

```
p=malloc(sizeof(struct
student));
scanf("%d",&p->num);
scanf("%f",&p->score);
p->next=head;
head=p;

p=head;
do{
    printf("%ld %5.1f\n",p-
>num,p->score);
    p=p->next;
}while(p!=NULL);

return 0;
}
```



共用体

共用体的定义及使用

```
union data
```

```
{
```

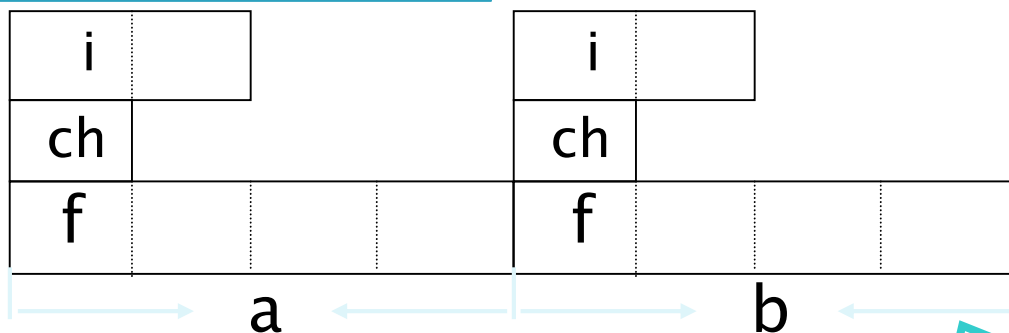
```
    int i;
```

```
    char ch;
```

```
    float f;
```

```
};
```

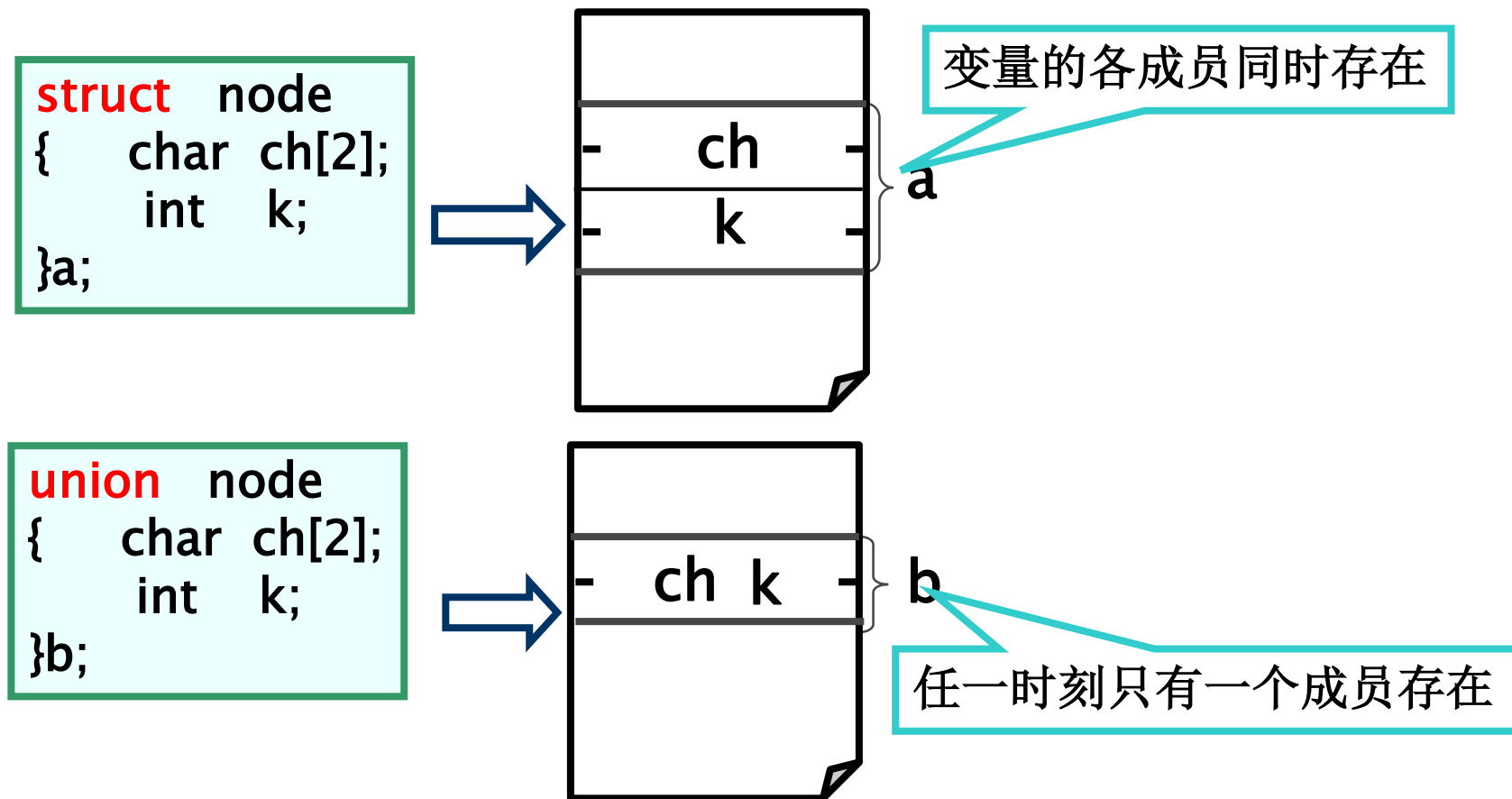
```
union data a,b,c,*p,d[3];
```



共用体变量任何时刻
只有一个成员存在

共用体变量定义分配内存，
长度=最长成员所占字节数

- 结构体与共用体
 - 区别: 存储方式不同
 - 联系: 可相互嵌套



思考

- ▶ 学校的管理系统中管理教工和学生两种人，他们的类别分别为T和S；
- ▶ 学生信息包括：姓名,号码,类别,性别,班级
- ▶ 教工信息包括：姓名,号码,类别,性别,职位
- ▶ 班级为整型
- ▶ 职位为字符数组

name	num	sex	type	class
				position
Li	1011	F	S	501
Wang	2086	M	T	prof

类型定义

```
union banjiorzhiwei  
{  int class;  
    char position[10];  
};
```

```
struct person  
{  int num;  
    char name[10];  
    char sex;  
    char type;  
    union banjiorzhiwu category;  
};
```

```
struct person p[100];
```

练习

- ▶ 尝试对上述

name	num	sex	type	class / position
Li	1011	F	S	501
Wang	2086	M	T	prof

循环n次			
读入姓名、号码、性别、职务			
读入class		job=='s'	
		真	假
		job=='t'	
		真	假
		读入 position	输出 “输入错”

循环n次	
job=='s'	
真	假
输出:姓名,号码,性别,职业,班级	输出:姓名,号码,性别,职业,职务

枚举类型

- ▶ 如果一个变量的值只有几种可能的值，可以定义为“枚举”类型。所谓“枚举”是指将变量的值一一列举出来，变量的值只限于列举出来的值的范围内。

1. 枚举类型的定义

`enum 枚举类型名 {取值表};`

例: `enum weekdays {Sun,Mon,Tue,Wed,Thu,Fri,Sat};`

2. 枚举变量的定义——与结构体变量类似

常用形式: `enum weekdays workday;`



3. 说明

(1) 枚举型仅适应于取值有限的数。

例如，根据现行的历法规定，1周7天，1年12个月。

(2) 取值表中的值称为枚举元素，对枚举元素按常量处理，故称枚举常量。因此不能够对其赋值。

(3) 枚举元素作为常量是有值的——定义时的顺序号（从0开始）。所以枚举元素可以进行比较，比较规则是：序号大者为大！

例如，上例中的Sun=0、Mon=1、.....、Sat=6，所以Mon>Sun、Sat最大。

例：

```
workday=mon;
printf("%d",workday);
```

(4) 枚举元素的值也是可以人为改变的：在定义时由程序指定。

例如，如果enum weekdays {Sun=7, Mon=1, Tue, Wed, Thu, Fri, Sat}; 则Sun=7, Mon=1, 从Tue开始，依次增1。

(5) 不能将整数赋给枚举变量，如workday=1; 是错误的。但可以通过强制类型转换后赋值，如workday=(enum weekdays)1; 相当于workday=mon;

typedef

起外号

`typedef` 原类型名 新类型名;

用typedef定义类型

除可直接使用C提供的标准类型和自定义的类型（如结构体类型）外，也可使用typedef定义已有类型的别名。该别名与标准类型名一样，可用来定义相应的变量。

格式

typedef **类型定义** 类型名;

可以是标准类型，
也可以是自定义类型。

举例

```
typedef int INTEGER;  
typedef float REAL;  
INTEGER i,j;  
REAL a,b;
```

```
typedef struct student{  
    char name[10];  
    int score;  
    struct student *next;  
}STU,*PSTU;
```

```
STU stu;  
PSTU pstu;
```