第十章

结构体、共用体、位操作和枚举类型

- 主要内容

10.1 概述
10.2 定义结构体类型变量的方法
10.3 结构体变量的引用
10.4 结构体变量的初始化
10.5 结构体数组
10.6 指向结构体类型数据的指针
10.7 共用体
10.8 枚举类型
10.9 用typedef定义类型

§10.1 概述

问题定义：

将不同类型的数据组合成一个有机的整体，以便于引用。如：

一个学生有学号/姓名/性别/年龄/地址等属性

int num; char name[20]; char sex; int age; char addr[30];

应当把它们组织成一个组合项，在一个组合项中包含若干个类型不同（当然也可以相同）的数据项。

num	name	sex	age	score	addr
100101	LiFun	M	18	87.5	Beijing

§10.1 概述

声明一个结构体类型的一般形式为：

```
struct   结构体名
    {成员表列};
```

如：

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

§10.2　定义结构体类型变量的方法

可以采取以下3种方法定义结构体类型变量：

(1)先声明结构体类型再定义变量名

例如：struct student　　student1, student2;

定义了student1和student2为struct student类型的变量，即它们具有struct student类型的结构.

student1	100102	WangLi	F	20	98	Beijing
student2	100101	ZhangXin	M	19	90.5	shanghai

§10.2　定义结构体类型变量的方法

在定义了结构体变量后，系统会为之分配内存单元。例如：student1和student2在内存中各占59个字节(2+20+1+2+4+30=59)。

注意：

将一个变量定义为标准类型（基本数据类型）与定义为结构体类型不同之处在于后者不仅要求指定变量为结构体类型，而且要求指定为某一特定的结构体类型，因为可以定义出许许多多种具体的结构体类型。

§10.2　定义结构体类型变量的方法

(2)在声明类型的同时定义变量

这种定义的一般形式为:

```
struct   结构体名
{
    成员表列

} 变量名表列;
```

§10.2　定义结构体类型变量的方法

例如：

```
struct student
 {
        int num;
     char name[20];
     char sex;
     int age;
     float score;
     char addr[30];
} student1,student2;
```

它的作用与第一种方法相同，即定义了两个struct student 类型的变量student1, student2

§10.2 定义结构体类型变量的方法

(3) 直接定义结构体类型变量

其一般形式为:

```
struct
 {
    成员表列
 } 变量名表列;
```

即不出现结构体名。

注意:

(1) 类型与变量是不同的概念，不要混同。只能对变量赋值、存取或运算，而不能对一个类型赋值、存取或运算。在编译时，对类型是不分配空间的，只对变量分配空间。

(2)对结构体中的成员（即“域”），可以单独使用，它的作用与地位相当于普通变量。

(3)成员也可以是一个结构体变量。

(4) 成员名可以与程序中的变量名相同,二者不代表同一对象。

§10.2 定义结构体类型变量的方法

先声明一个struct date类型，它代表“日期”，包括3个成员：month（月）、day（日）、year（年）。

然后在声明struct student类型时，将成员birthday指定为struct date类型。

例如：

```
struct date  /*声明一个结构体类型*/
{
 int num;
 char name[20];
 char sex;
 int age;
 float score;
 struct date birthday;  /*birthday是struct date类型*/
 char addr[30];
} student1,student2;
```

num	name	sex	age	birthday			addr
				Monday	day	year	

§10.3结构体变量的引用

在定义结构体变量之后,就可以引用这个变量。但应注意:

(1)不能将一个结构体变量作为一个整体进行输入和输出。

例如: 已定义student1和student2为结构体变量并且它们已有值。

```
printf("%d,%s,%c,%d,%f,% \ n",student1);
```

§10.3结构体变量的引用

结构体变量中成员的引用方式：

结构体变量名.成员名

例如：student1.num表示student1变量中的num成员,即student1的num(学号)项。

可以对变量的成员赋值,例如:student1.num=10010;

“.”是成员(分量)运算符,它在所有的运算符中优先级最高,因此可以把student1.num作为一个整体来看待。

上面赋值语句的作用是将整数10010赋给student1变量中的成员num。

§10.3结构体变量的引用

(2) 如果成员本身又属一个结构体类型，则要用若干个成员运算符,一级一级地找到最低的一级的成员。只能对最低级的成员进行赋值或存取以及运算。

例如: 对上面定义的结构体变量student1, 可以这样访问各成员:

student1.num

student1.birthday.month

注意：

不能用student1.birthday来访问student1变量中的成员birthday,因为birthday本身是一个结构体变量。

§10.3结构体变量的引用

(3) 对结构体变量的成员可以像普通变量一样进行各种运算（根据其类型决定可以进行的运算）。

例如：

```
student2.score=student1.score;
sum=student1.score+student2.score;
student1.age++;
++student2.age;
```

由于“.”运算符的优先级最高，因此student1.age++是对student1.age进行自加运算，而不是先对age进行自加运算。

§10.3结构体变量的引用

(4) 可以引用结构体变量成员的地址，也可以引用结构体变量的地址。

例如：

```
scanf("%d"，&student1.num);
(输入student1.num的值)
printf("%o"，&student1）；
(输出student1的首地址)
```

§10.3结构体变量的引用

但不能用以下语句整体读入结构体变量，

例如：

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

结构体变量的地址用作函数参数，传递结构体变量的地址。

§10.4结构体变量的初始化

例10.1

```
#include <stdio.h>
int main()
{struct student
   {     long int num;
         char name[20];
         char sex;
         char addr[20];
    }a={10101,"LiLin",'M',"123 Beijing Road"};
       /* 对结构体变量a赋初值*/
         printf("No.:%ld\nname:%s\nsex:%c\naddress:%s\n",
                              a.num,a.name,a.sex,a.addr);
}
```

运行结果：

No.：10101

name：LiLin

sex：M

address：123 Beijing Road

§10.5 结构体数组

一个结构体变量中可以存放一组数据（如一个学生的学号、姓名、成绩等数据）。

如果有10个学生的数据需要参加运算，就要用数组，这就是结构体数组。

结构体数组与数值型数组不同之处在于：每个数组元素都是一个结构体类型的数据，它们都分别包括各个成员（分量）项。

§10.5 结构体数组

定义结构体数组的一般形式

（1）struct 结构体名

{成员表列}数组名[数组长度]；

（2）先声明一个结构体类型（如struct Person）然后用此类型定义结构体数组

结构体类型 数组名[数组长度]；

struct Person leader[3]

§10.5 结构体数组-应用举例

例10.2 有3个候选人，每次输入一个得票候选人的名字，要求最后输出各人得票结果。

解决思路：设一个结构体数组，数组中包含3个元素，每个元素中包含候选人的姓名（字符型）和得票数（整型）。输入被选人姓名，然后与数组元素中的姓名成员比较，如果相同，就给这个元素中的“得票数”成员的值+1。最后输出所有元素的信息。

§10.5 结构体数组-应用举例

程序定义一个全局的结构体数组leader，它有 3 个元素，每一个元素包含两个成员name（姓名）和count（票数）。在定义数组时并初始化，使3位候选人的票数都先置零.

在主函数中定义字符数组leader-name，它代表被选人姓名，在10次循环中每次先输入一个被选人具体人名，然后把它与3个候选人姓名相比，看它和哪一个候选人的名字相同。在输入和统计结束之后，将3人的名字和得票数输出。

name	count
Li	0
Zhang	0
Fun	0

图10-6

§10.5 结构体数组-应用举例

例10.2

```
#include <string.h>
#include <stdio.h>
struct person
{
        char name[20];
        int count;
}leader[3]={"Li",0,"Zhang",0,"Sun",0};
```

```
int main()
{       int i,j;
        char leader_name[20]; // 投票给谁
        for(i=1;i<=10;i++)// 投票的次数
        {
        scanf("%s", leader_name);
        for(j=0;j<3;j++)
        if(strcmp(leader_name,leader[j].name)==0)
        leader[j].count++;
        }
        printf("\n");
        for(i=0;i<3;i++)
        printf("%5s:%d\n",leader[i].name,leader[i].count);
}
```

运行结果：

```
Li↙
Sun↙
Zhang↙
Zhang↙
Sun ↙
Li↙
Sun↙
Zhang↙
Li↙
Li:4
Zhang:3
Sun:3
```

§10.6 结构体指针

结构体指针: 指向结构体变量的指针，一个结构体变量的起始地址就是这个结构体变量的指针。如果把一个结构体变量的起始地址存放在一个指针变量中，这个指针就指向该结构体变量。

10.6.1 指向结构体变量的指针

指向结构体对象的指针变量既可指向结构体变量，也可指向结构体数组中的元素。指针变量的基类型必须与结构体变量的类型相同。

```
struct student *ptr
```

§10.6 结构体指针

例9.3 通过指向结构体变量的指针变量输出结构体变量的成员。

拟解决的问题：

如何对结构体变量成员赋值；

怎样通过指向结构变量的指针访问结构体变量中的成员。

例10.3指向结构体变量的指针应用

```
#include <string.h>
#include <stdio.h>
int main()
{struct student
        {long num;
        char name[20];
        char   sex;
        float score;};
        struct student stu_1;
        struct student * p;
        p=&stu_1;
        stu_1.num=89101;
        strcpy(stu_1.name,"LiLin");
        stu_1.sex='M';
        stu_1.score=89.5;
        printf("No.:%ld\nname:%s\nsex:%c\nscore:%f\n",
                stu_1.num,stu_1.name,stu_1.sex,stu_1.score);
        printf("No.:%ld\nname:%s\nsex:%c\nscore:%f\n",
                (*p).num,(*p).name,(*p).sex,(*p).score);
}
```

运行结果：

```
No.: 89101
name: LiLin
sex: M
score: 89.500000
No.: 89101
name: LiLin
sex: M
score: 89.500000
```

§11.6结构体指针

程序分析：

在函数的执行部分将结构体变量stu_1的起始地址赋给指针变量p，也就是使p指向stu_1,然后对stu_1的各成员赋值。第一个printf函数是输出stu_1的各个成员的值。用stu_1.num表示stu_1中的成员num，依此类推。第二个printf函数也是用来输出stu_1各成员的值，但使用的是（*p）.num的形式。

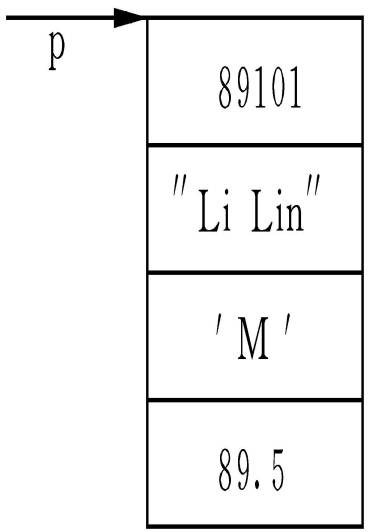


图10-7

§11.6结构体指针

如果p指向一个结构体变量，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

其中->称为指向运算符。

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。**

§11.6结构体指针

2 指向结构体数组的指针

例10.4 有三个学生的信息，放在结构体中，要求输出全部学生的信息。

解题思路：用指向结构体变量的指针来处理。

例10.4

```
#include <stdio.h>
struct student
{       int num;
        char name[20];
        char sex;
        int age;};
        struct student stu[3]={{10101,"Li Lin",'M',18},
                               {10102,"Zhang Fun",'M',19},
                               {10104,"WangMing",'F',20}};

int main()
{   struct student  *p;
    printf("  No. Name          sex       age\n");
    for(p=stu;p<stu+3;p++)
      printf("%5d %-20s %2c %4d\n",
                    p->num, p->name, p->sex, p->age);
}
```

运行结果：

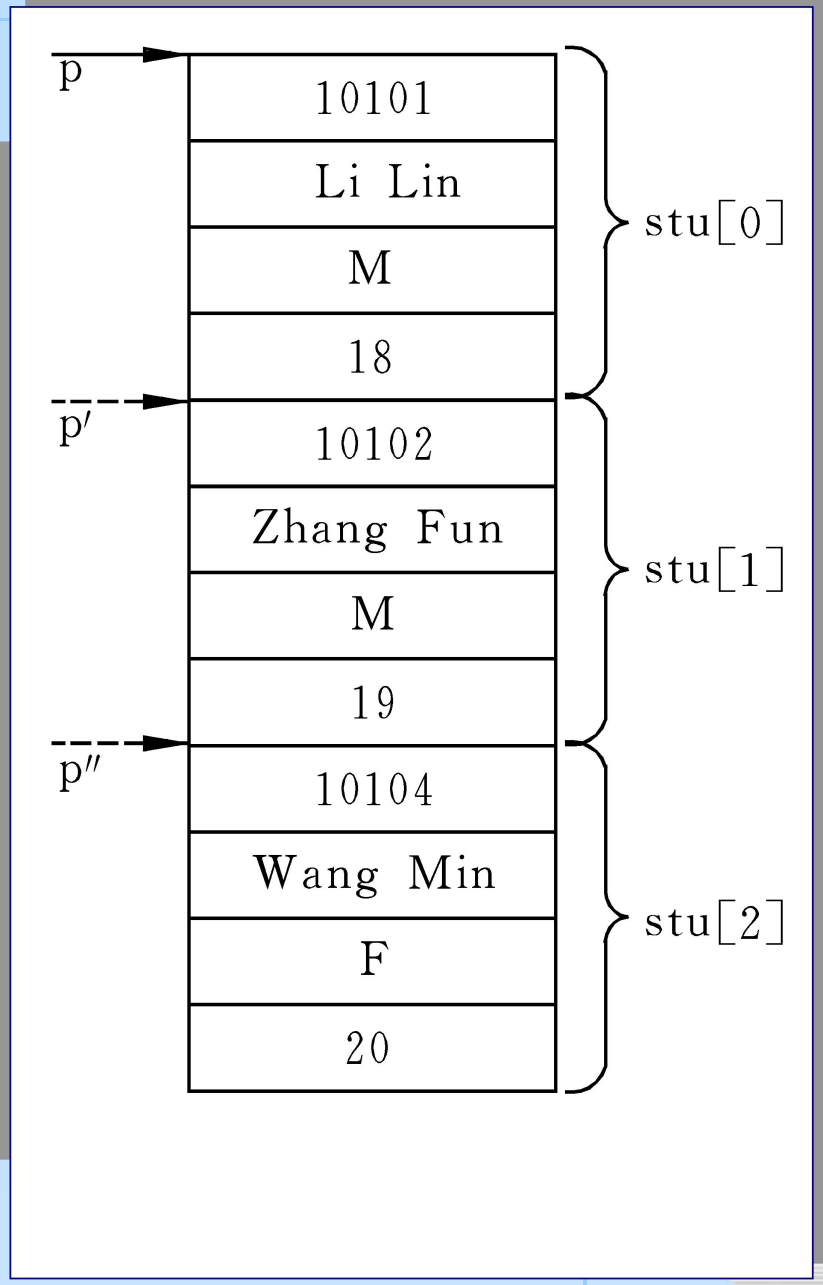
```
  No.          Name          sex     age
10101       LiLin             M        18
10102    Zhang Fun            M        19
10104    WangMing             F        20
```

§10.6结构体指针

程序分析：

p是指向struct student结构体类型数据的指针变量。在for语句中先使p的初值为stu，也就是数组stu第一个元素的起始地址。

在第一次循环中输出stu[0]的各个成员值。然后执行p++，使p自加1。p加1意味着p所增加的值为结构体数组stu的一个元素所占的字节数。执行p++后p的值等于stu++，p指向stu[1]。在第二次循环中输出stu[1]的各成员值。在执行p++后,p的值等于stu+2，再输出stu[2]的各成员值。在执行p++后，p的值变为stu+3，已不再小于stu+3了，不再执行循环。



§10.6结构体指针

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)->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++)->num 先得到p->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。

§10.6结构体指针

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=stu[1].name;**

如果要将某一成员的地址赋给p，可以用强制类型转换，先将成员的地址转换成p的类型。

例如： p = (struct student *) stu[0].name;

§10. 6结构体指针

10.6.3 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.

§10.6结构体指针

例10.5 有一个结构体变量stu，包含学生学号、姓名和3门课程成绩。要求在main函数中赋值，在利用print函数将它们输出。用结构体变量作函数参数。

```
#include <stdio.h>
#include<string.h>
#define FORMAT "%d\n%s\n%f\n%f\n%f\n"
struct student
{
    int num;
    char name[20];
    float score[3];
};
```

§10.6结构体指针

```
int main()
{
    void print(struct student);
    struct student stu;
    stu.num=12345;
    strcpy(stu.name, "LiLin");
    stu.score[0]=67.5;
    stu.score[1]=89;
    stu.score[2]=78.6;
    print(stu);
}
void print(struct student stu)
{
 printf(FORMAT,stu.num,stu.name, stu.score[0], stu.score[1], stu.score[2]);
    printf("\n");
}
```

运行结果：

```
1 2 3 4 5
L i L i
67.500000
89.000000
78.599998
```

例10.6 将上题改用指向*结构体变量的指针*作实参。

```
#include <stdio.h>
#include<string.h>
#define FORMAT "%d\n%s\n%f\n%f\n%f\n"
struct student
{   int num;
    char name[20];
    float score[3];
}stu={12345,"LiLin",67.5,89,78.6};

int main()
{  void print(struct student *);
    print(&stu);                          /*实参改为stu的起始地址*/
}
void print(struct student *p)             /*形参类型修改指向结构体的指针变量*/
{   printf(FORMAT,p->num,p->name,p->score[0],p->score[1],p->score[2]);
                                          /*用指针变量调用各成员的值*/
    printf("\n");
}
```

运行结果：

```
1 2 3 4 5
L i  L i n
67.500000
89.000000
78.599998
```

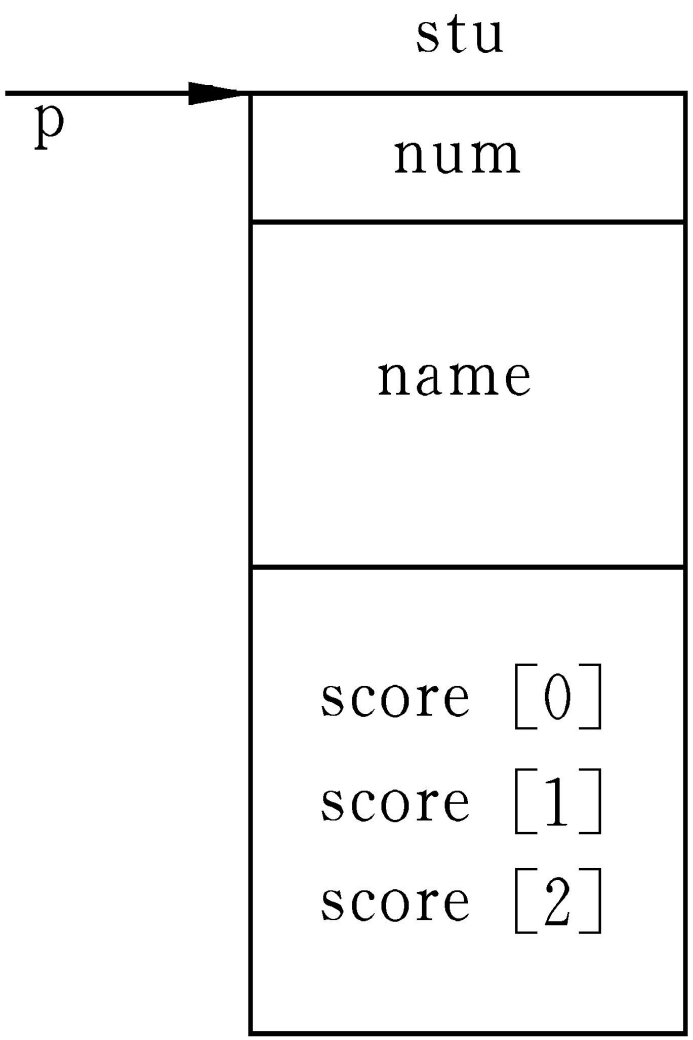
§10.6结构体指针

程序分析：

此程序改用在定义结构体变量stu时赋初值，这样程序可简化些。print函数中的形参 p 被定义为指向struct student类型数据的指针变量。

注意在调用print函数时，用结构体变量str的起始地址&stu作实参。在调用函数时将该地址传送给形参p(p是指针变量）。这样 p 就指向stu。在print函数中输出 p 所指向的结构体变量的各个成员值，它们也就是stu的成员值。

main函数中的对各成员赋值也可以改用scanf函数输入。



Questions & Answers

§10.8 共用体

10.8.1共用体的概念

几个不同的变量共占同一段内存的结构称为共用体类型的结构.

定义共用体类型变量的一般形式为:

```
union   共用体名
        {
          成员表列
        } 变量表列;
```

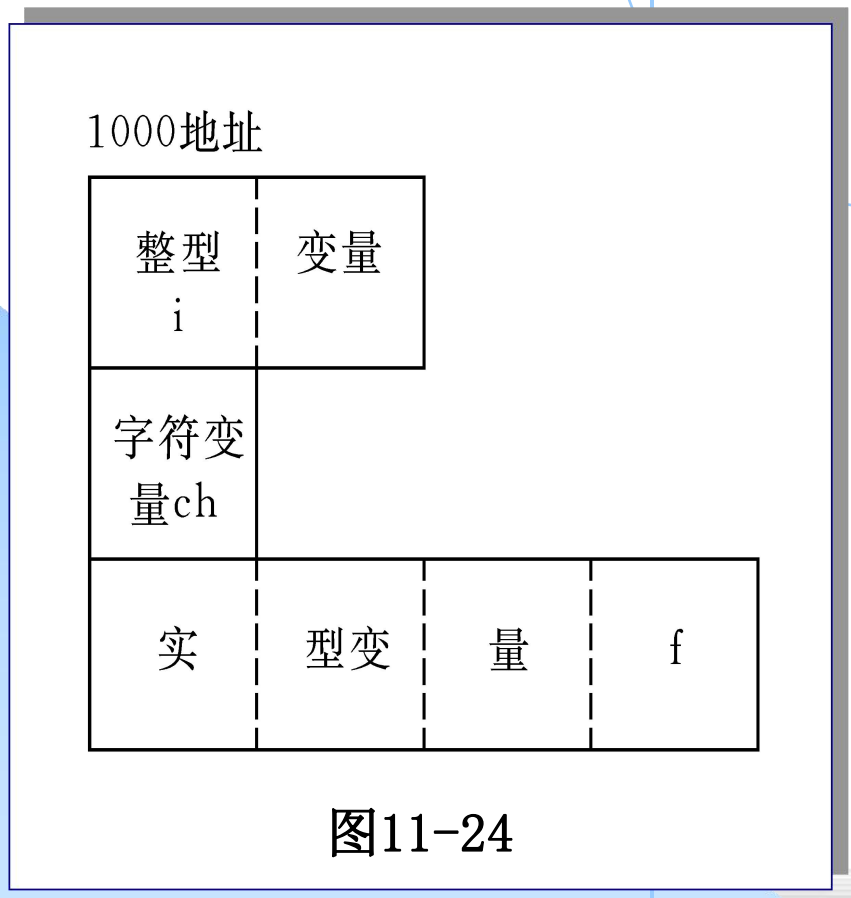


图11-24

§10.8 共用体

例如：

```
union data
  {   int i;
      char ch;   或
      float f;
} a,b,c;
union data a,b,c;
```

```
union data
{   int i;
    char ch;
    float f;
};
```

§ 10.8 共用体

共用体和结构体的比较：

结构体变量所占内存长度是各成员占的内存长度之和。每个成员分别占有其自己的内存单元。

共用体变量所占的内存长度等于最长成员的长度。

例如：**上面定义的“共用体”变量a、b、c各占4个字节（因为一个实型变量占4个字节），而不是各占2 + 1 + 4 = 7个字节。**

§ 10.8 共用体

10.8.2 共用体变量的引用方式

只有先定义了共用体变量才能引用它，而且不能引用共用体变量，而只能引用共用体变量中的成员。

例如：前面定义了a、b、c为共用体变量

a.i （引用共用体变量中的整型变量 i ）

a.ch（引用共用体变量中的字符变量 c h）

a.f （引用共用体变量中的实型变量 f ）

§ 10.8 共用体

10.8.3 共用体类型数据的特点

(1)同一个内存段可以用来存放几种不同类型的成员，但在每一瞬时只能存放其中一种，而不是同时存放几种。

(2) 共用体变量中起作用的成员是最后一次存放的成员，在存入一个新的成员后原有的成员就失去作用。

(3) 共用体变量的地址和它的各成员的地址都是同一地址。

§ 10.8 共用体

(4) 不能对共用体变量名赋值，也不能企图引用变量名来得到一个值，也不能在定义共用体变量时对它初始化。

(5) 不能把共用体变量作为函数参数，也不能使函数带回共用体变量，但可以使用指向共用体变量的指针

(6) 共用体类型可以出现在结构体类型定义中，也可以定义共用体数组。反之，结构体也可以出现在共用体类型定义中，数组也可以作为共用体的成员。

§ 10.8 共用体

例10.12 设有若干个人员的数据，其中有学生和教师。

学生的数据中包括：姓名、号码、性别、职业、班级。

教师的数据包括： 姓名、号码、性别、职业、职务。

学生和教师所包含的数据是不同的。现要求把它们放在同一表格中。

num	name	sex	job	class(班) position(职务)
101	Li	f	s	501
102	Wang	m	t	prof

可以用共用体来处理第5项。

§ 10.8 共用体

例10.12

```
#include <stdio.h>
struct
{
        int num;
        char name[10];
        char sex;
        char job;
        union
        {
            int banji;
            char position[10];
        } category;
}person[2];   /*先设人数为2*/
```

运行情况如下：

101 Li f s 501↙

102 Wang m t professor↙

No.	Name	sex	job	class/position
101	Li	f	s	501
102	Wang	m	t	professor

```
int main()
{    int i;
     for(i=0;i<2;i++)
            {
            scanf("%d %s %c %c", &person[i].num, &person[i].name, &person[i].sex,
                                                    &person[i].job);
            if(person[i].job == 'S')  scanf("%d", &person[i].category.banji);
            else if(person[i].job == 'T') scanf("%s", person[i].category.position);
            else printf("Input error!");
            }
            printf("\n");
            printf("No. name sex job class/position\n");
            for(i=0;i<2;i++)
            {
            if (person[i].job == 'S')
               printf("%-6d%-10s%-3c%-3c%-6d\n",person[i].num,
                person[i].name, person[i].sex, person[i].job, person[i].category.banji);
            else  printf("%-6d%-10s%-3c%-3c%-6s\n",person[i].num,
                person[i].name,person[i].sex,person[i].job, person[i].category.position);
            }
}
```

§10.9 枚举类型

枚举：将变量的值都列举出来，变量的值只限于列举出来的值的范围内。

声明枚举类型用enum

enum weekday{sun，mon，tue，wed，thu，fri，sat};

定义变量：

enum weekday workday，week-day;

enum{sun，mon，tue，wed，thu，fri，sat} workday;

变量值只能是sun到sat之一

§10.9 枚举类型

说明:

在C编译中，对枚举元素按常量处理，故称枚举常量。它们不是变量，不能对它们赋值。

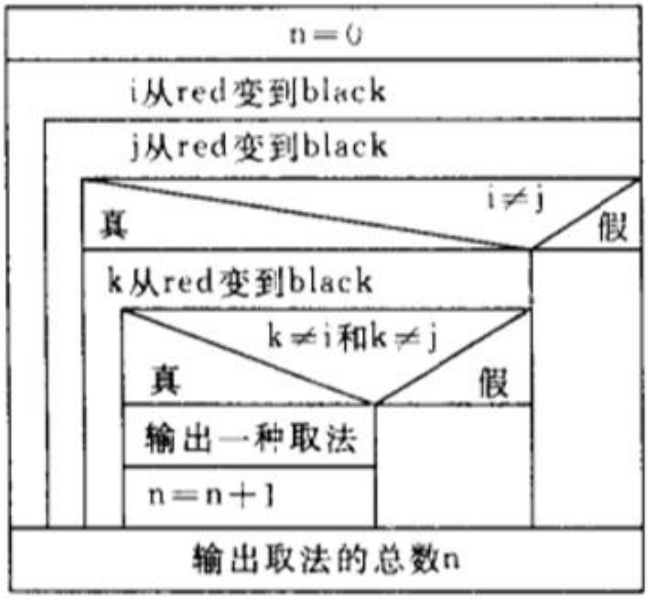
(2) 枚举元素作为常量，它们是有值的，C语言编译按定义时的顺序使它们的值为0，1，2...

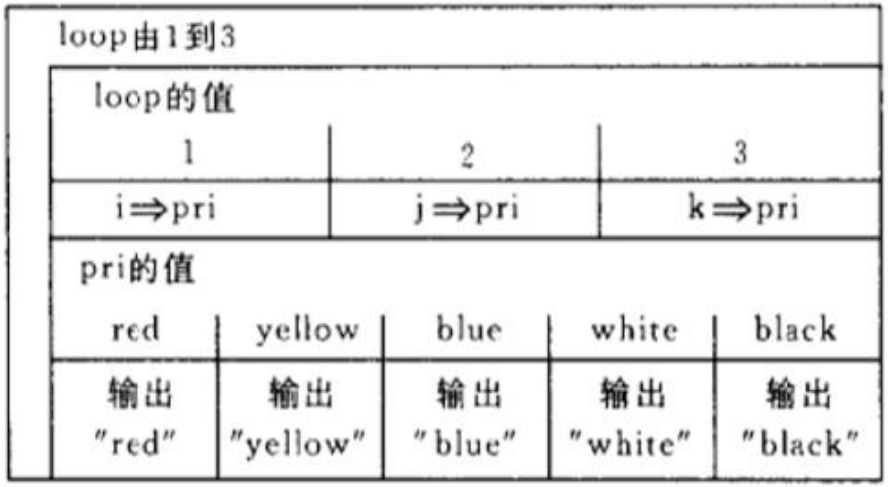
(3) 枚举值可以用来作判断比较。

(4) 一个整数不能直接赋给一个枚举变量。

§10.9 枚举类型

例10.13 口袋中有红、黄、蓝、白、黑5种颜色的球若干个。每次从口袋中先后取出3个球，问得到3种不同色的球的可能取法，输出每种排列的情况。





§10.9 枚举类型

例10.13解决思路

用 n 累计得到 3 种不同色球的次数。外循环使第 1 个球 i 从 red 变到 black。中循环使第 2 个球 j 也从 red 变到 black。如果 i 和 j 同色则不可取，只有 i、j 不同色(i≠j)时才需要继续找第 3 个球，此时第 3 个球 k 也有 5 种可能(red 到 black)，但要求第 3 个球不能与第 1 个球或第 2 个球同色，即 k≠i，k≠j。满足此条件就得到 3 种不同色的球。输出这种 3 色组合方案，然后使 n 加 1。外循环全部执行完后，全部方案就已输出完了。最后输出总数 n。

下面的问题是如何实现图 11-28 中的“输出一种取法”。这里有一个问题：如何输出 red、blue……单词。不能写成 printf("%s",red)来输出“red”字符串。可以采用图 11-28 的方法。

为了输出 3 个球的颜色，显然应经过 3 次循环，第 1 次输出 i 的颜色，第 2 次输出 j 的颜色，第 3 次输出 k 的颜色。在 3 次循环中先后将 i、j、k 赋予 pri。然后根据 pri 的值输出颜色信息。在第 1 次循环时，pri 的值为 i，如果 i 的值为 red，则输出字符串“red”，其他的类推。

```
#include <stdio.h>
int main()
{
        enum color{red,yellow,blue,white,black}; //声明枚举类型color
        enum color i,j,k,pri;
        int n=0,loop;
        for(i=red;i<=black;i++)          {          //当i为某一颜色时
                for (j=red;j<=black;j++){          //当j为某一颜色时
                        if (i!=j){                 //若前两个球的颜色不同
                        for (k=red;k<=black;k++)
                                if ((k!=i) && (k!=j))
                                {
                                n=n+1;
                                printf("%-4d",n);
                                for (loop=1;loop<=3;loop++)
                                {
                                switch (loop){
                                        case 1: pri=i; break;
                                        case 2: pri=j; break;
                                        case 3: pri=k; break;
                                        default:break;
                }
```

```
                switch (pri){
                        case red:       printf("%-10s","red"); break;
                        case yellow:    printf("%-10s","yellow"); break;
                        case blue:      printf("%-10s","blue"); break;
                        case white:     printf("%-10s","white"); break;
                        case black:     printf("%-10s","black"); break;
                        default :break;
                }
            }
                printf("\n");
        }
    }
  }
}
printf("\ntotal:%5d\n",n);
}
```

1	red	yellow	blue
2	red	yellow	white
3	red	yellow	black
4	red	blue	yellow
5	red	blue	white
6	red	blue	black
7	red	white	yellow
8	red	white	blue

49	black	red	yellow
50	black	red	blue
51	black	red	white
52	black	yellow	red
53	black	yellow	blue
54	black	yellow	white
55	black	blue	red
56	black	blue	yellow
57	black	blue	white
58	black	white	red
59	black	white	yellow
60	black	white	blue

total: 60

§10.10 用typedef定义类型

用typedef声明新的类型名来代替已有的类型名

声明INTEGER为整型

```
typedef int INTEGER
```

声明结构类型

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

§10.10 用typedef定义类型

声明NUM为整型数组类型

typedef int NUM[100]

声明STRING为字符指针类型

typedef char *STRING;

声明POINTER为指向函数的指针类型，该函数返回整型值

typedef int (*POINTER)()

§10.10 用typedef定义类型

用typedef定义类型的方法

① 先按定义变量的方法写出定义体（如：int i）。

② 将变量名换成新类型名（例如：将i换成COUNT）。

③ 在最前面加typedef

(例如：typedef int COUNT)。

④ 然后可以用新类型名去定义变量。

§10.10 用typedef定义类型

用typedef定义类型的方法（举例）

① 先按定义数组变量形式书写：int n[100];

② 将变量名 n 换成自己指定的类型名：

int NUM[100]] ；

③ 在前面加上typedef，得到

typedef int NUM[100];

④ 用来定义变量：NUM n;

§10.10 用typedef定义类型

说明:

(1) 用typedef可以声明各种类型名，但不能用来定义变量。

(2) 用typedef只是对已经存在的类型增加一个类型名，而没有创造新的类型。

(3) 当不同源文件中用到同一类型数据时，常用typedef声明一些数据类型，把它们单独放在一个文件中，然后在需要用到它们的文件中用#include命令把它们包含进来。

(4) 使用typedef有利于程序的通用与移植。

§10.10 用typedef定义类型

(5) typedef与#define有相似之处，例如：

typedef int COUNT；#define COUNT int的作用都是用COUNT代表int。但事实上，它们二者是不同的。

#define是在预编译时处理的，它只能作简单的字符串替换，而typedef是在编译时处理的。实际上它并不是作简单的字符串替换，而是采用如同定义变量的方法来声明一个类型

Questions & Answers