

# 第7讲

聚合数据组织与处理：

结构体

# 主要内容

7.1 结构体的概念

7.2 结构体的定义、初始化及引用

7.3 结构体数组、结构体作为函数参数和返回值

7.4 共用体与枚举

# 主要内容

7.1 结构体的概念

7.2 结构体的定义、初始化及引用

7.3 结构体数组、结构体作为函数参数和返回值

7.4 共用体与枚举

## 7.1 结构体的概念

### 结构体的引入

- 在处理大批量的数据时往往使用数组，但数组有其局限性，即只有在所有数据类型相同的情况下才能使用数组
- 数组无法组织和处理由多种不同数据类型构成的混合数据，由此引出一种新的数据类型——**结构体**

- 结构体就是这样一种数据类型，它包含了一组数据，这组数据可以具有不同的类型

```
struct Student      // 结构体类型名
{
    int num;          // 成员1类型及名称
    char name[20];    // 成员2类型及名称
    float score;      // 成员3类型及名称
};
```

- 结构体通常又被称为**聚合数据类型**，或记录类型、表单类型
  - ✓ 它将一组数据值聚合成一个单独的数据类型，用这组数据共同描述一个对象，形成一个整体
  - ✓ 结构体中的每个数据值叫做**结构体成员**，每个成员都有一个名字

- 结构体实现了“**数据封装**”
  - ✓ **数组**：将**类型相同**的一组**有序数据**封装成一个整体，需要时可以从数组中选出**元素**，单独进行操作
  - ✓ **结构体**：将**类型不同**的一组**无序数据**封装成一个整体，需要时可以从结构体变量中选出**成员**，单独进行操作

## ● 结构体类型与结构体变量

- ✓ 结构体类型：系统对之不分配存储单元，无具体数据
- ✓ 结构体变量：分配存储单元，可以存放具体的数据
- ✓ 在构建结构体类型后，就可以定义结构体变量、结构体数组、结构体指针



## ● 结构体与指针

- ✓ 两者联合使用时，可以实现链表、堆栈、队列和树等复杂的数据结构
- ✓ 可以实现动态数据组织与处理
- ✓ 很多情况下，使用结构体指针比结构体本身更加合理

## 7.2 结构体变量的定义、初始化与引用

- 在定义结构体变量之前，首先需要构建（定制）结构体类型：
  - ✓ 构建结构体类型：定制该结构体类型的变量由哪些成员（字段）组成，每个成员的名字及类型
  - ✓ 定义结构体变量：编译器按照构建的结构体类型给变量分配相应的内存空间，存入数据

10

- 结构体类型的构建

格式: **struct 结构体类型名**  
**{ 成员表列; };**

例如: **struct Student**  
**{**  
    **int num;**  
    **char name[20];**  
    **float score;**  
**};**

## 说明:

- (1) 成员名可以与程序中的变量名相同，不同的结构体中也可以有相同的成员名；
- (2) 结构体成员的类型可以是任意类型，可以为整型、字符型、浮点型，也可以是数组或结构体类型；
- (3) 结构体类型构建可放在函数之外，也可以放在函数内部。一般放在main函数之前，通常将构建结构体类型的代码保存在头文件中
- (4) 系统对结构体类型不分配内存

- 结构体变量的定义

- (1) 类型构建与变量定义分开

**struct 结构体类型名 变量名;**

例: **struct Student** //构建了一个结构体类型

```
{  
    int num;  
    char name[20];  
    float score;  
};
```

**struct Student st1, st2;** //定义了两个结构体变量

## (2) 在构建结构体类型的同时定义变量

```
struct 结构体类型名  
{ 成员表列; }结构体变量名;
```

```
例: struct Student  
    {  
        int num;  
        char name[20];  
        float score;  
    }st1, st2;
```

### (3) 不构建结构体类型，直接定义结构体变量

**struct**

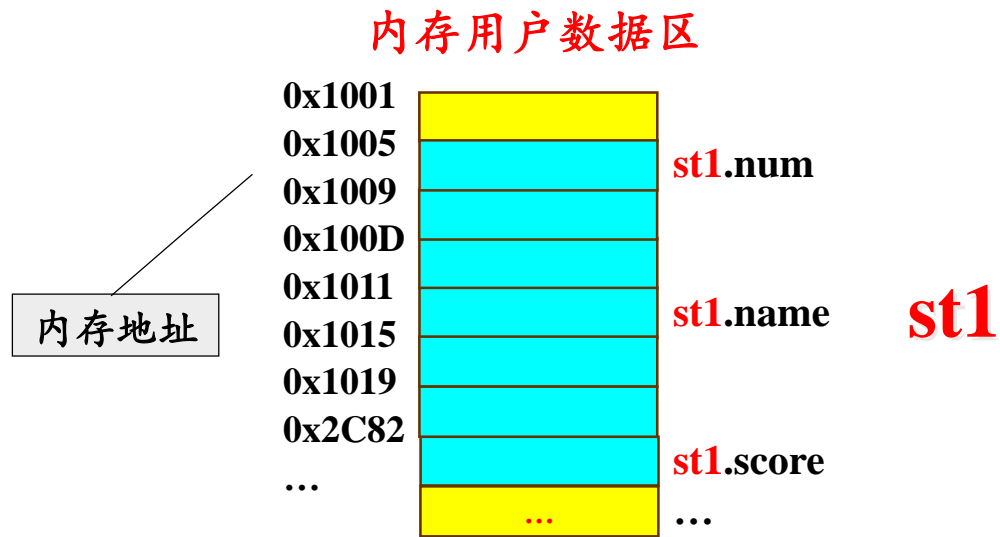
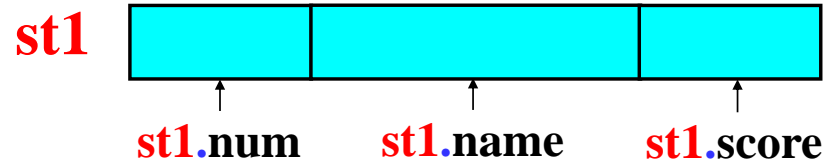
**{ 成员表列; }结构体变量名;**

例: **struct**  
    {  
        int num;  
        char name[20];  
        float score;  
    }**st1, st2;**

## 说明:

- (1) 形式(1)最常见，适用于需要大量引用该结构体类型的情况；形式(2)是简略形式，此后该结构体类型还可以再次引用，适用于引用该结构体类型不太多的情况；形式(3)适用于一次性定义变量的场合
- (2) 定义结构体变量后，系统分配一片连续的存储单元，结构体变量的大小为各成员长度之和。可以用sizeof运算符测量该结构体类型所占内存空间的大小





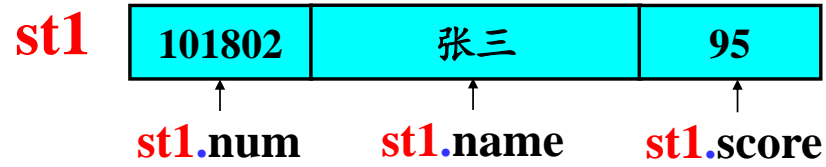
## ● 结构体变量的初始化

- (1) `struct 结构体类型名 变量名={初值表列};`
- (2) `struct 结构体类型名{ 成员表列; }结构体变量名={初值表列};`
- (3) `struct{ 成员表列; }结构体变量名={初值表列};`

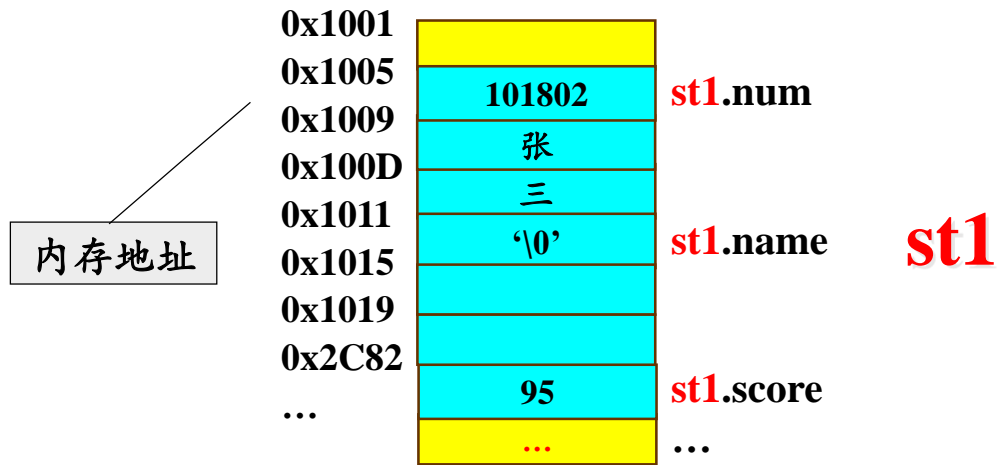
**例：** `struct Student st1={101802,"张三",95},  
st2={101806,"李四"};`

`struct Student  
{ int num; char name[20]; float score;  
}st1={101802,"张三",95}, st2={101806,"李四",89};`

**struct**  
`{ int num; char name[20]; float score;  
}st1={101802,"张三",95}, st2={101806,"李四", 89};`



内存用户数据区



- 结构体变量的引用

格式： 结构体变量名 . 成员名

如： `st1.num`, `st1.name`, `st1.score`

例如：

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

struct Student
{
    int num;
    char name[20];
    struct date birthday;
    float score;
}st1={101802, "张三", 2000,3,5,95}, st2;
```

## 说明:

- (1) 成员运算符“.”的级别最高;
- (2) 如果成员本身是一个结构体变量, 则只能一级一级对最低级的成员进行存取与运算  
如: `st1.birthday.year=2000;`
- (3) 对结构型变量的整体操作只限于赋值操作、作为函数参数或返回值传递, 但要求必须是同一类型的结构体变量, 且不能对结构型变量整体进行其他操作  
如: `st2=st1;` // 允许

## 7.3 结构体数组、 结构体作为函数参数和返回值

- **结构体数组：**

- ✓ 结构体数组中每个元素都是同一种结构体类型的变量，可以存放批量的数据
- ✓ 结构体数组的定义与引用和基本类型的数组相同



## ● 结构体数组的定义

格式：

- (1) `struct 结构体类型名 数组名[数组长度];`
- (2) `struct 结构体类型名{ 成员表列; }数组名  
[数组长度];`
- (3) `struct{ 成员表列; }数组名[数组长度];`

```
struct Student  
{ int num; char name[20]; float score; };  
struct Student st[10];
```

```
struct Student  
{ int num; char name[20]; float score; }st[10];
```

```
struct  
{ int num; char name[20]; float score; }st[10];
```

## ● 结构体数组的初始化

- ✓ 与普通数组一样，结构体数组也可以在定义的同时对其进行初始化
- ✓ 初始化的形式是在定义的结构体数组名后加上“**= {初值表列};**”

如： **struct Student st[3]**={{101802,"张三", 95},  
                                  {101806,"李四", 89},  
                                  {101809,"王五", 74}};

- 结构体数组的使用

- ✓ 引用结构体数组元素

如： `struct Student st[2], tmp;`

`tmp=st[0]; st[0]=st[1]; st[1]=tmp;`

- ✓ 引用结构体数组元素的成员

如： `struct Student st[2]; gets(st[1].name);`

- ✓ 引用结构体数组地址

**例7-1:** 从键盘上输入5名学生的信息，计算每个学生四门课程的平均成绩，用选择排序法按平均成绩由大到小进行排序，输出排好序的学生的所有信息（学号、姓名、四门课程的成绩、平均成绩），输出平均成绩时保留二位小数。

● **分析**

本题用结构体方法比较合适，数据的输入、处理和输出均在main函数中完成

```
#include <stdio.h>  
#define N 5  
struct Student  
{ int num;  
    char name[20];  
    float score[4];  
    float aver;  
};
```

```
void main( )  
{  
    int i,j,k; struct Student stu[N], temp;  
    printf("\n请输入学生信息：学号、姓名、四门课成绩:\n");  
    for(i=0;i<N;i++)  
    {  
        scanf("%d %s %f %f %f%f",  
                &stu[i].num, stu[i].name, &stu[i].score[0],  
                &stu[i].score[1], &stu[i].score[2], &stu[i].score[3]);  
        stu[i].aver=(stu[i].score[0]+stu[i].score[1]+stu[i].score[2]  
                    + stu[i].score[3])/4.0;  
    }  
}
```

```

for(i=0;i<N-1;i++) // 选择法排序
{
    k=i;
    for(j=i+1;j<N;j++)
        if(stu[j].aver>stu[k].aver) k=j;
    temp=stu[k]; stu[k]=stu[i]; stu[i]=temp; // 结构体整体的使用
}
printf("The final result:\n"); // 输出结果
for(i=0; i<N; i++)
    printf("%6d %8s %6.1f%6.1f%6.1f%6.1f%7.2f\n", stu[i].num,
        stu[i].name,stu[i].score[0], stu[i].score[1], stu[i].score[2],
        stu[i].score[3], stu[i].aver);
}

```

```

请输入各学生的信息：学号、姓名、四门课成绩：
1001 张三 89 95 78 100
1002 李四 90 67 99 92
1003 王五 89 93 88 74
1004 赵六 100 77 84 97
1005 周七 79 94 96 86
The final result:
1001 张三 89.0 95.0 78.0 100.0 90.50
1004 赵六 100.0 77.0 84.0 97.0 89.50
1005 周七 79.0 94.0 96.0 86.0 88.75
1002 李四 90.0 67.0 99.0 92.0 87.00
1003 王五 89.0 93.0 88.0 74.0 86.00

```



## ● 结构体与函数：

- ✓ 结构体变量成员作为函数参数（传值引用）
- ✓ 结构体变量作为函数参数（传值引用）
- ✓ 结构体作为函数返回值（传值引用）
- ✓ 结构体指针作为函数参数（地址引用）

## 说明:

- (1) 结构体变量成员作为函数参数时，是单向值传递；
- (2) 结构体变量作为函数参数时，是单向值传递，将实参中每个成员的值传给形参。实参、形参类型需相同；
- (3) 结构体作为函数返回值时，整个结构体类型值会返回到主调函数中
- (4) 结构体变量作为函数参数和返回值时，会增加系统开销，一般采用地址引用方式

`#include<stdio.h>`     **//例7-2： 结构体变量成员作为函数参数**

`int max(int x,int y)`  
`{    return x>y?0:1; }`

`void main( ) {`  
`struct Student {`

`int num;    char name[20];      float score;`  
`}st[2]={101802,"张三", 89}, {101806,"李四", 95}};`

`if(max(st[0].score, st[1].score)==0) printf("The info. of %s is:`  
`No.=%d Score=%.2f\n",st[0].name,st[0].num,st[0].score);`  
`else if(max(st[0].score, st[1].score)==1) printf("The info. of %s`  
`is: No.=%d Score=%.2f\n",st[1].name,st[1].num,st[1].score);`  
`else printf("Error!\n");`

`}`

The info. of 李四 is: No.=101806 Score=95.00

**例7-3：**读入5名学生的学号、姓名、四门课程的成绩；编写一个函数Max找出这5名学生中平均成绩最高的学生，输出其相关信息。

● **分析**

本题用结构体结合函数方法实现，数据的输入、输出均在main函数中完成；查找平均成绩最高的功能采用“打擂台”方法，由自定义函数Max实现；main函数调用Max函数

```
#include <stdio.h>  
#define N 5  
struct Student  
{ int num;  
    char name[20];  
    float score[4];  
    float aver;  
};
```

```

void main( )
{
    struct Student Max(struct Student stu_1[ ]);
    struct Student stu[N], t;    int i, j;
    printf("\n 请输入学生信息：学号、姓名、四门课成绩:\n");
    for(i=0;i<N;i++)
    {
        scanf("%d %s %f %f %f%f",&stu[i].num, stu[i].name,
            &stu[i].score[0],&stu[i].score[1],&stu[i].score[2],&stu[i].score[3]);
        stu[i].aver=0;
        for(j=0; j<4; j++)  stu[i].aver += stu[i].score[j];
        stu[i].aver /= 4.0;
    }
    t=Max(stu);    // stu是什么?
    printf(" The info. of student who average score is greatest:\n");
    printf(" %6d %8s %6.1f%6.1f%6.1f%6.1f%7.2f\n",
        t.num, t.name, t.score[0], t.score[1], t.score[2], t.score[3], t.aver);
}

```

38

```

struct Student Max(struct Student stu_1[ ])
{
    int i;
    struct Student t = stu_1[0];
    for(i=1;i<=N-1;i++)
        if(stu_1[i].aver>t.aver) t = stu_1[i];
    return t;
}

```

```

请输入学生信息：学号、姓名、四门课成绩：
1001 张三 89 95 78 100
1002 李四 90 67 99 92
1003 王五 89 93 88 74
1004 赵六 100 77 84 97
1005 周七 79 94 96 86
The info. of student who average score is greatest:
1001 张三 89.0 95.0 78.0 100.0 90.50

```

## 7.4 共用体与枚举

### ● 共用体

- ✓ 在处理大批量的聚合数据时往往使用结构体，但结构体有其局限性，即只有在所有成员都有自己的空间的的情况下才能使用结构体
- ✓ **共用体**是一种新的数据类型，其特点是：不同类型的成员共享同一段空间，但各成员在不同的时刻使用，起始地址相同



- 共用体的技术基础是覆盖存储技术  
例：一张床可供大家轮流睡觉、一个储物柜大家轮流存放物品...
- 共用体又称联合体、公用体
- 共用体的类型构建、变量定义及使用类似于结构体，但它有自己的特点

- 共用体类型的构建（定制）及变量定义

格式1: `union 共用体类型名`  
`{ 成员表列; };`  
`union 共用体类型名 变量表列;`

格式2: `union 共用体类型名`  
`{ 成员表列; }变量表列;`

格式2: `union`  
`{ 成员表列; }变量表列;`

如：

```
union data
{
    int a;
    char b;
    double c;
};
union data x, y, z;
```

```
union data
{
    int a;
    char b;
    double c;
}x, y, z;
```

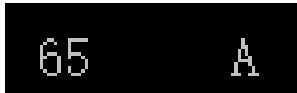
```
union
{
    int a;
    char b;
    double c;
}x, y, z;
```

## ● 共用体类型与结构体类型的比较

### 不同之处:

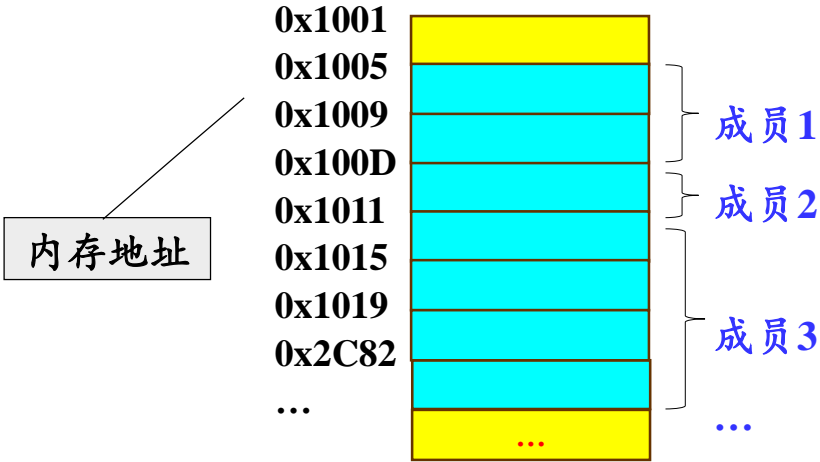
- (1) 存储结构不同
- (2) 结构体每个成员拥有自己**独立**的存储空间，所有成员可以**同时**存储数据；共用体中所有成员拥有**同一个**存储空间，同一时刻只能存储**一个**数据
- (3) 初始化时，结构体变量可以对所有成员进行初始化，而共用体变量只能对一个成员进行初始化
- (4) 可以单独引用结构体变量的各个成员；共用体变量只能引用最后一次存入的成员

```
#include<stdio.h>
void main( )
{
    union data{ int a;  char b; double c;}x={'A'}, y;
    y=x;
    printf("%5d %5c\n", y.a, y.b);
}
```

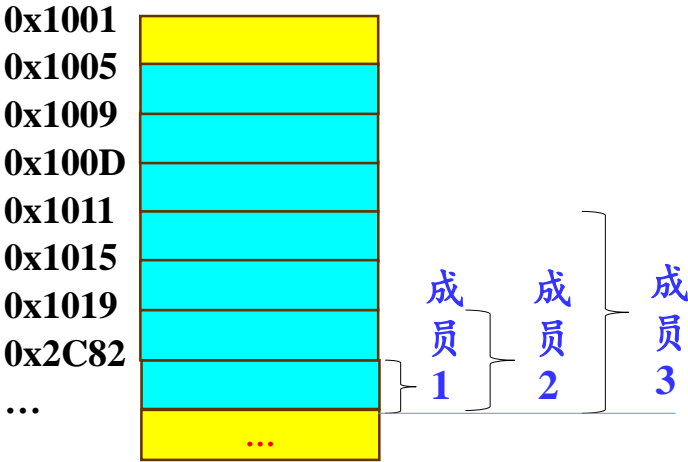


```
65  A
```

结构体变量的存储分配



共用体变量的存储分配



## 相同之处:

(1) 两个相同类型的共用体变量可以整体赋值

例: `#include<stdio.h>`  
`void main( ) {`  
 `union data{ int a; char b; double c;}x,y;`  
 `x.a=128; x.b='A';`  
 `y=x;`  
 `printf(" %5d %5c\n", y.a, y.b);`  
`}`

65	A
----	---

(2) 允许将共用体变量作为函数参数

**例7-4：**计2018级体育课测试，男生测试1500米跑（成绩用分钟表示），女生测试仰卧起坐次数，测试结果放到一张表格中，表中包括该学生的学号、姓名、性别和体育成绩。编写程序输出该表格

- **分析**

本题需要利用结构体+共用体方法



```
#include<stdio.h>  
#define N 4  
struct Student  
{  
    int num;  
    char name[20];  
    char sex;  
    union  
    {  
        float run;  
        int situp;  
    }score;  
};
```

```
void main( ) {  
    int i, n;  
    struct Student stu[N];  
    for(i=0; i<N; i++)  
    {  
        printf(" 请输入：学号、姓名、性别： \n");  
        scanf("%d%s %c", &stu[i].num, stu[i].name, &stu[i].sex);  
        if(stu[i].sex=='m' || stu[i].sex=='M')  
        { printf(" 请输入他的1500米成绩(分钟): ");  
          scanf("%f", &stu[i].score.run);  
        }  
        else  
        { printf(" 请输入她的仰卧起坐次数: ");  
          scanf("%d", &stu[i].score.situp);  
        }  
    }  
}
```

```
for(i=0; i<N; i++)  
    if(stu[i].sex=='m' || stu[i].sex=='M')  
        printf("%-6d %-10s 男 1500米成绩(分):%.4f\n",  
                stu[i].num, stu[i].name, stu[i].score.run);  
    else  
        printf("%-6d %-10s 女 仰卧起坐次数: %d\n",  
                stu[i].num, stu[i].name, stu[i].score.situp);  
}
```

```

请输入：学号、姓名、性别：
1001 张三 m
请输入他的1500米成绩(分钟)： 5.18
请输入：学号、姓名、性别：
1002 李四 f
请输入她的仰卧起坐次数： 30
请输入：学号、姓名、性别：
1003 王五 M
请输入他的1500米成绩(分钟)： 5.27
请输入：学号、姓名、性别：
1004 赵六 F
请输入她的仰卧起坐次数： 53
1001 张三 男 1500米成绩(分)： 5.1800
1002 李四 女 仰卧起坐次数： 30
1003 王五 男 1500米成绩(分)： 5.2700
1004 赵六 女 仰卧起坐次数： 53

```

## ● 枚举类型

- ✓ 变量的取值被限定在**有限的范围**  
如：体育比赛结果（赢、输、平）、一周（周一~周日）、一年（春、夏、秋、冬）...
- ✓ 变量的取值可以用整数表示（如：周一用1表示、周日用7表示），但这种表示可读性差；  
若定义为符号常量，则无法体现这些常量间的内在联系，且不能作为一个完整的逻辑整体

- 枚举类型的构建（定制）及变量定义

格式1: `enum 枚举类型名`  
`{ 枚举值表列; };`  
`enum 枚举类型名 变量表列;`

格式2: `enum 枚举类型名`  
`{ 枚举值表列; }变量表列;`

格式2: `enum`  
`{ 枚举值表列; }变量表列;`

如：

```
enum week{ Sun, Mon, Tue, Wed, Thu, Fri, Sat };
```

```
enum week x, y, z;
```

```
enum week{ Sun, Mon, Tue, Wed, Thu, Fri, Sat }x, y, z;
```

```
enum{ Sun, Mon, Tue, Wed, Thu, Fri, Sat }x, y, z;
```

## 说明:

- (1) 枚举类型属于简单类型，不是构造类型
- (2) 枚举值（枚举元素）是程序员自行定义的标识符，枚举值不能重复，每个枚举值有一个序号，默认从0开始按自然数顺序增长；但可以改变该序号  
`enum week{ Sun=7, Mon=1, Tue, Wed, Thu, Fri, Sat };`
- (3) 枚举变量的值只能在枚举值中取，可以将枚举值或枚举变量赋给一个枚举变量，但不能将一个整数赋给一个枚举变量
- (4) 枚举值可以比较大小，按序号进行比较



### 例7-5:

输入一个整数，输出其所对应的英文名称

- 分析

本题要求利用枚举方法

```

#include<stdio.h>
void main( )
{
    enum week{ Sun=7, Mon=1, Tue, Wed, Thu, Fri, Sat };
    enum week day;
    int n;
    printf("\n 输入一个整数: ");
    scanf("%d", &n);
    day=(enum week)n;
    switch(day) {
        case Sun: printf(" Sunday\n"); break;
        case Mon: printf(" Monday\n"); break;
        case Tue: printf(" Tuesday\n"); break;
        case Wed: printf(" Wednesday\n"); break;
        case 4: printf(" Thursday\n"); break;
        case 5: printf(" Friday\n"); break;
        case 6: printf(" Saturday\n"); break;
        default: printf("Data Error\n"); break;
    }
}

```

```

输入一个整数: 3
Wednesday

```

```

输入一个整数: 5
Friday

```

# Homework

- 实践

实验5 聚合数据程序设计

- 作业

《教材》： P329 2、4、5