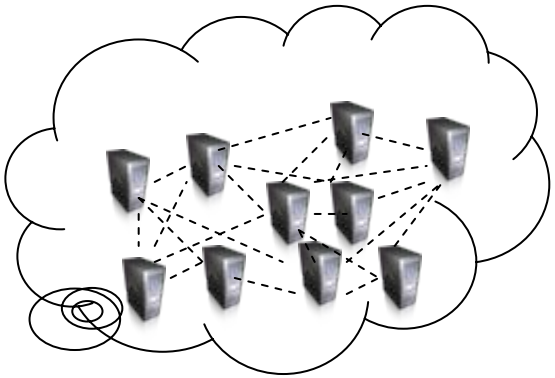
厦门大学非计算机专业本科生公共课

（2012-2013第2学期）



《C语言程序设计》

第9章 结构体、共用体和枚举类型

林子雨

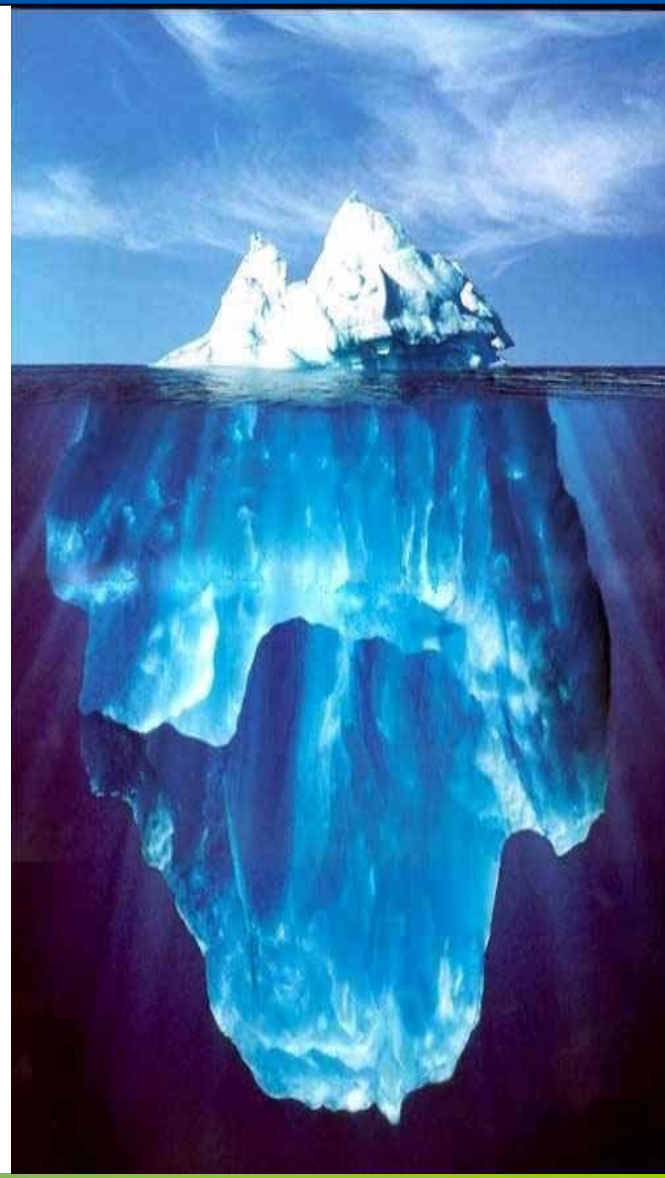
厦门大学计算机科学系

E-mail: ziyulin@xmu.edu.cn

个人主页：**http://www.cs.xmu.edu.cn/linziyu**

课程提要

- 第一章 绪论
- 第二章 C语言基础
- 第三章 结构化程序设计
- 第四章 选择结构
- 第五章 循环结构程序设计
- 第六章 函数
- 第七章 编译预处理
- 第八章 数组
- 第九章 结构体、共用体和枚举类型
- 第十章 指针



第9章 结构体、共用体和枚举类型

9.1 结构体

9.1.0 结构体概述

- 数组类型用于表示一组相关联的、同类型的数据集合。
- 一个数组不仅存储一批同类型数据，而且能表达各数据元素之间的线性关系。
- 在实际应用中，有时要将不同类型的数据元素组合成为一个有机整体。例如：一种商品包括商品编号、商品名称、商品价格、商品库存量等信息，这些数据项是相互关联的。
- 在C语言中，为了将这些相互联系而类型不同的数据作为一个整体处理，引入了结构体类型。
- 结构体类型由一组相互关联的数据元素（元素类型可以相同或不相同）构造而成。

9.1.1 结构体类型的定义

- 结构体类型比较复杂，系统无法事先为用户定义一种统一的结构体类型。在定义结构体变量之前，用户要先定义结构体类型，即用自己定义的结构体类型定义结构体变量。
- 定义结构体类型，应该指出该结构体类型叫什么名字，包含哪些数据元素，各数据元素叫什么名字，属于什么数据类型等。
- 定义结构体类型的一般格式如下：

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

实例：

```
struct Commodity
{
        char Name[20];
        int Price,Count;
        char Provenance[30];
};
```

注意：结构体类型定义是一个语句，要以分号结束。

9.1.2 结构体变量定义和初始化

- 9.1.2.1 结构体变量的定义
- 9.1.2.2 结构体变量的初始化

9.1.2.1 结构体变量的定义

- 结构体类型反映的是所处理对象的抽象特征，而要描述具体对象时，就需要定义结构体类型的变量，简称结构体变量或结构体。
- 结构体变量的定义必须在结构体类型定义之后，它的一般形式是：

  struct 结构体类型名 结构体变量名;
- 例如，前面已经定义了结构体类型struct Commodity，可以用它来定义变量：

  struct Commodity TV;

Name	Price	Count	Provenance
Television	2100	20	Fujian

图 结构体变量存储分配示意图

9.1.2.1 结构体变量的定义

- 结构体变量定义也可以和结构体类型定义同时进行，其形式如下：

```
struct 结构体名
{
    数据类型 成员项1;
    数据类型 成员项2;
    …
    数据类型 成员项n;
} 结构变量表;
```

例：

```
struct Commodity
{
        char Name[20];
        int Price;
        int Count;
        char Provenance[30];
} c1,c2,c3;
```

注:当结构体类型和结构体变量同时定义时，可以省略结构体名。但是，由于省略了结构体名，在程序的其他位置就不能再使用这种结构体类型定义其他结构体变量。

9.1.2.2 结构体变量的初始化

•如果结构体变量的值已知，在定义结构体变量的同时，可以给它的成员赋初值，这就是结构体的初始化。它的一般形式为：

struct 结构体类型 结构体变量={初始化数据表};

•其中，初始化数据表中各数据之间用逗号隔开，初始化数据表中的数据个数必须和结构体成员项的个数相同，而且数据类型必须适合于相对应的成员项。例如：

Commodity TV={“Television”,2100,30,”Fujian”};

注意：如果定义结构体变量时没有初始化，其各成员项取系统设定的默认值，即自动变量取随机值，静态变量取0。

9.1.3 结构体变量的引用

- 9.1.3.1 对结构体变量成员项的引用
- 9.1.3.2 对结构体变量的整体引用

9.1.3.1 对结构体变量成员项的引用

- 在程序设计中，对结构体变量的引用，主要是引用它的成员项。对结构体变量成员的引用形式为：

  结构变量名.成员名

  其中，“.”为分量运算符。因为运算符“.”在C语言所有运算符中优先级别最高，所以可以把“结构体变量名.成员名”看作是一个整体。

  例如上面定义的结构体变量TV，它的成员引用形式是：

  TV.Name, TV.Provenance

  例如：

  TV.Count=30;

  TV.Provenance=“Fujian”;

- 结构体的成员项也称为成员变量，其作用和简单变量一样，可以作为表达式的运算对象，进行各种操作运算。例如：

  scanf(“%s”,TV.Name);

  scanf(“%d”,&TV.Price);

  TV.Price-=100;

9.1.3.1 对结构体变量成员项的引用

- 例9.1.1编写一个程序，统计库存电视机的总价格。程序清单如下：

```
struct          Commodity          //定义结构体类型，作为外部标识符
{
      char      Name[20];
      int       Price;
      int       Count;
      char      Provenance[30];
};
void  main()
{
      int Total;
       //定义并初始化结构体变量
      Commodity  TV=  {“Television”, 2100,20,“Fujian”};
      Total=TV.Price*TV.Count;
      printf("The Total  Prices of  %s  is  %d\n ",TV.Name,Total);
}
```

注意：结构体类型定义语句可以放置在函数内部，也可以放置在函数外部。如果结构体类型名作为全局标识符，可以在各函数中使用。

9.1.3.2 对结构体变量的整体引用

- 同类型的结构体变量可以相互赋值（包括作为函数参数的值传递），除此之外，不能对结构体变量进行整体引用。

例9.1.2 结构体整体赋值示例。

```
void  main()
{
        struct  {int a;char b;} st1={20,'a'},st2;
        st2=st1;
        printf("%d\n%c\n",st2.a,st2.b);
}
```

9.1.3.2 对结构体变量的整体引用

- 例**9.1.3**结构体变量作为函数参数示例。

```
struct  st {int a;char b;} ;
void print(st st1)
{
    printf("st.a=%d\nst.b=%c\n",st1.a,st1.b);
}
void  main()
{
    st st2={20,'a'};
    print(st2);//调用函数print时，把实际参数st2的值赋给形式参数st1
}
```

9.1.3.2 对结构体变量的整体引用

所谓同类型的结构体变量是指在同一语句定义的变量（如例9.1.2），或由同一个结构体类型标识符定义的变量（如例9.1.3）。下面程序是错误的：

```
void  main()
{
        struct  {int a;char b;} st1={20,'a'};
        struct  {int a;char b;} st2;
        st2=st1; //错误
        printf("%d\n%c\n",st2.a,st2.b);
}
```

9.1.4 结构体数组

- **1、结构体数组的定义**
- 结构体数组的定义一般形式是：
  struct 结构体名 结构体数组名[整常量表达式];
- 例如：

```
struct Commodity
{
    char Name[20];
    int price;
};
struct Commodity aTV[3];
```

上面定义了一个结构体数组aTV。 aTV有三个元素，分别是aTV[0]、 aTV[1]和aTV[2]。aTV各元素类型是Commodity，各元素有两个成员。

9.1.4 结构体数组

- **2、结构体数组的初始化**
- 结构体数组在定义的同时也可以进行初始化，一般形式为：
  struct 结构体类型 结构体数组名[整常量表达式]={初始化数据表};
- 例如：

```
struct Commodity
{
    char Name[20];
    int price;
} aTV[3]={
    {“TV21”,1500},
    {“TV27”,1800},
    {“TV29”,2100}
};
```

•在对结构体数组进行初始化时，方括号中的数组长度可以省略。编译时，系统会根据初始值个数来确定结构体数组元素的个数；
•对结构体数组的初始化，初始化数据表中内层花括号可以省略。

9.1.4 结构体数组

- 例9.1.4 在学生成绩表中求各学生的总成绩和全班各课程的平均成绩。

序号	姓名 (name)	数学 (math)	语文 (Chinese)	英语 (English)	总成绩 (sum)
1	Zhang San	90	86	92	
2	Li Si	88	75	78	
3	Wang Wu	91	74	65	
全班平均					

9.1.4 结构体数组

例**9.1.4**在学生成绩表中求各学生的总成绩和全班各课程的平均成绩。

```
#include<stdio.h>
#define  N  3
struct  student
{    char name[10];
          int maths , chinese , english , sum ;
} ;
void  main()
{
    student  stu[N+1]; //stu[0]存储全班平均成绩，stu[i]存储第i个学生的成绩
    int i, m_ave=0,c_ave=0,e_ave=0;
    printf("请输入各学生的姓名、数学、语文和英语成绩\n");
    for(i=1;i<=N;i++)
    {       printf("第%d个学生：",i);
            scanf("%s",stu[i].name);
            scanf("%d%d%d",&stu[i].maths,&stu[i].chinese,&stu[i].english);
            stu[i].sum=stu[i].maths+stu[i].chinese+stu[i].english;//计算各学生总成绩
            m_ave+=stu[i].maths;                    //计算全班数学课总分
            c_ave+=stu[i].chinese;                  //计算全班语文课总分
            e_ave+=stu[i].english;                  //计算全班英语课总分
    }
```

9.1.4 结构体数组

（续）例**9.1.4**在学生成绩表中求各学生的总成绩和全班各课程的平均成绩。

```
    stu[0].maths=m_ave/N;                    //计算全班数学课平均分
    stu[0].chinese=c_ave/N;                  //计算全班语文课平均分
    stu[0].english=e_ave/N;                  //计算全班英语课平均分
    for(i=1;i<=N;i++)
            printf("%s\t%d\t%d\t%d\t%d\n",stu[i].name,stu[i].maths,stu[i].chinese,
stu[i].english,stu[i].sum);
printf("平均：\t%d\t%d\t%d\n",stu[0].maths,stu[0].chinese,stu[0].english);
}
```

9.1.4 结构体数组

- 例**9.1.5** 在学生成绩表中求各学生的总成绩和全班各课程的平均成绩，用不同于例**9.1.4**中的数据结构来处理。

序号	姓名(name)	成绩(score)			
		数学(1)	语文(2)	英语(3)	总成绩(0)
1	Zhang San	90	86	92	
2	Li Si	88	75	78	
3	Wang Wu	91	74	65	
全班平均					

9.1.4 结构体数组

例9.1.5代码

```
#include<stdio.h>
#define  N  3
#define  M  3
struct  student
{     char name[10];
      int score[M+1] ;//每个学生有M个课程，score[j]是第j课程成绩，score[0]是总成绩
} ;
void  main()
{
      student  stu[N+1];  //stu[0]存储全班平均成绩，stu[i]存储第i个学生的成绩
      int i,j;
      for(i=0;i<=N;i++)
            stu[i].score[0]=0;          //各学生的总成绩赋初值0
      for(j=0;j<=M;j++)
            stu[0].score[j]=0;          //各课程的总成绩赋初值0
      printf("请输入各学生的姓名、数学、语文和英语成绩\n");
      for(i=1;i<=N;i++)
      {     printf("第%d个学生：",i);
            scanf("%s",stu[i].name);                  //输入第i个学生的姓名
            for(j=1;j<=M;j++)
            {           scanf("%d",&stu[i].score[j]);          //输入第i个学生的第j门课程成绩
                        stu[i].score[0]+=stu[i].score[j];      //求第i个学生的总成绩
                        stu[0].score[j]+=stu[i].score[j];      //求第j门课程的总成绩
            }
            stu[0].score[0]+=stu[i].score[0];      //求各学生总成绩的总和
      }
```

9.1.4 结构体数组

（续例**9.1.5**代码）

```
    for(j=0;j<=M;j++)
            stu[0].score[j]/=N;                    //计算各门课程的平均成绩
    for(i=1;i<=N;i++)
    {
            printf("%s\t",stu[i].name);
            for(j=1;j<=M;j++)
                    printf("%d\t",stu[i].score[j]);
            printf("%d\n",stu[i].score[0]);
    }
    printf("平均：\t");
    for(j=1;j<=M;j++)
            printf("%d\t",stu[0].score[j]);
    printf("%d\n",stu[0].score[0]);
}
```

9.2 共用体

- 有时为了节省空间，把不同用途的数据存放在同一个存储区域，这种数据类型称为共同体类型，又称联合体类型。构成共用体变量的各成员项的数据类型可以是相同的，也可以是不同的。
- 共用体类型和共用体变量的定义方式与结构体的定义方法相似，共用体成员的引用也和结构体成员的引用方法类似。二者主要区别在于对成员项的存储方式上。

9.2.1 共用体的类型定义

- 可以先定义共用体类型，然后用已定义的共用体类型定义共用体变量；也可以把共用体类型和共用体变量放在一个语句中一次定义。
- 定义共用体类型，使用关键字union。共用体类型定义的一般形式为：

```
union 共用体名
{
    数据类型  成员项1;
    数据类型  成员项2;
    …
    数据类型  成员项n;
};
```

- 和结构体类型定义一样，在没有定义共用体变量之前，共用体类型定义只是说明了共用体变量使用的内存模式，并没有分配具体的存储空间。
- 例如

```
union Variable
{
        short int i;
        char c;
        float f;
};
```

- 定义了共用体类型union Variable，union Variable类型的变量由i、c和f3个成员项组成，这三个成员项在内存中使用共同的存储空间，即它们在内存中具有相同的首地址。

9.2.2 共用体变量的定义、初始化和引用

- 1、共用体变量的定义
- 定义共用体类型之后，就可以用已定义的数据类型定义具体的共用体变量，定义共用体变量的一般形式为：

  union 共用体类型名 共用体变量列表;
- 例如，在上面定义了共用体类型Variable之后，就可以用它定义变量：

  union Variable unVar1,unVar2;
- 定义共用体变量后，系统为共用体变量分配存储空间。共用体变量存储空间的长度不是该变量所有成员项的空间长度之和，而是把长度最大的成员项的存储空间作为共用体变量的存储空间。

9.2.2 共用体变量的定义、初始化和引用

- **2、共用体变量的初始化**
- 可以对共用体变量进行初始化，一般格式为：
  union 共用体类型名 共用体变量={初始值};
- 例如：
  Variable unVar1={65};
- 注意，大括号不能省略，大括号中只能提供一个值。
- **3、共用体变量的引用**
- 共用体变量unVar1的成员项是unVar1.i， unVar1.c和unVar1.f。
- 由于共用体的各个成员项公用一块存储空间，所以，共用体的存储空间在某一个时刻只能保存某个成员项的数据。

9.3 枚举类型

- 9.3.1 枚举类型的定义
- 9.3.2 枚举变量的定义、初始化和使用
- 9.3.2.1 枚举变量的定义
- 9.3.2.2 枚举变量初始化
- 9.3.2.3 枚举变量的使用

9.3.1 枚举类型的定义

- 定义枚举类型就是定义该类型的值集合，即枚举变量可能的取值范围。
- 枚举类型定义的一般形式为：
  enum 枚举类型名 {枚举元素表};
- 例如：
  enum Day {Mon, Tue, Wed, Thu, Fri, Sat, Sun};
- **说明**：标识符Mon,Tue等称为枚举元素，也称为枚举常量。枚举元素是该枚举型变量可能的取值。枚举元素是标识符，必须符合标识符的构成规则。
- 在字符型中，一个字符和一个整数对应，如'A'对应65。同样，在枚举类型中，一个枚举元素也和一个整数对应，默认情况下，第一个枚举元素的值为0，然后按序递增1.例如上面的定义中，Mon和0对应，Tue和1对应，Sun和6对应。与枚举元素对应的整数称为枚举元素的序号。
- 定义枚举类型，可以对枚举元素表中的枚举元素指定序号，这可以通过在该枚举元素之后加一个等号和一个整数来实现，例如：
  enum Day{Mon=1,Tue,Wed,Thu,Fri,Sat,Sun=0};
- 这样， Mon的序号为1，Tue的序号为2， Sat的序号为6，Sun的序号为0.
- 如果某枚举元素没有指定序号，则该元素的序号为前一元素序号加1。枚举元素表中任何两个元素的序号不能相同。
- 定义枚举类型而不直接使用整数，是因为便于记忆，增加程序可读性。

9.3.2.1 枚举变量的定义

- 定义某枚举类型之后，就可以定义该类型的变量，定义枚举变量的一般形式为：

  enum 枚举类型名 枚举变量列表;
- 例如：

  enum Day enDay;
- 上面语句定义了枚举变量enDay，变量enDay的取值范围只能是Mon,Tue,Wed,Thu,Fri,Sat,Sun之一。
- 在VC++中，当用已定义的枚举类型定义枚举变量时，保留字enum可以省略，即上面枚举变量定义语句也可以写成：

  Day enDay;
- 枚举类型和枚举变量的定义也可以同时进行，例如：

  enum Day{Mon=1,Tue,Wed,Thu,Fri,Sat,Sun=0} enDay;
- 当枚举类型和枚举变量同时定义时，枚举类型可以省略，例如：

  enum {Mon=1,Tue,Wed,Thu,Fri,Sat,Sun=0} enDay;

9.3.2.2 枚举变量初始化

- 可以使用枚举元素或整数对枚举变量进行初始化。但使用整数时必须进行类型转换。例如：

  Day enDay =Sun;

  或

  Day enDay=(Day)0;

9.3.2.3 枚举变量的使用

- 枚举值是简单型的数据，和整型、字符型数据一样，可以作为运算对象出现在表达式中。不过枚举值一般只限于以下运算：
- （1）可以将枚举值（枚举元素或枚举变量）赋给枚举变量和整型变量。例如：

```
Day d;
int I;
d=Mon;
i=Mon;
```

- （2）可以将整数（包括整型表达式的值）赋给枚举变量，但赋值前应进行类型转换。例如：
- d=(Day)(d+1);
- （3）可以对枚举值进行关系运算，系统以枚举元素序号大小作为比较依据。
- （4）可以输出枚举值的序号。由于不能对枚举变量直接输入和输出，使得枚举类型的应用受到影响。

9.3.2.3 枚举变量的使用

- **例9.3.1** 枚举数组（数组元素类型为枚举型的数组）使用示例，程序清单如下：

```
#include <stdio.h>
enum Day{Mon=1,Tue,Wed,Thu,Fri,Sat,Sun=0};
void main()
{
    //定义枚举数组并初始化
    enum Day enDay[]={Mon,Tue,Wed,Thu,Fri,Sat,Sun };
    for(int i=0;i<7;i++)
            printf("%d\t",enDay[i]);   //输出数组元素enDay[i]的值
    printf("\n");
}
```

程序的运行结果为：

1 2 3 4 5 6 0

9.3.2.3 枚举变量的使用

- **例9.3.2** 枚举类型和字符串对比示例，程序清单如下：

```
#include <stdio.h>
#include <string.h>
enum Day{Mon=1,Tue,Wed,Thu,Fri,Sat,Sun=0};
void main()
{
    printf("enum:  ");
if(Thu>Fri)                          //枚举值比较
       printf("Thu>Fri\n");
    else
       printf("Thu<Fri\n");
printf("string:  ");
    if(strcmp("Thu","Fri")>0)  //字符串比较
       printf("Thu>Fri\n");
    else
       printf("Thu<Fri\n");
}
```

程序的运行结果：

enum: Thu<Fri

string: Thu>Fri

9.4 typedef语句

- typedef是type define的缩写。其实，typedef语句并不是用于定义新的数据类型，而是为已定义的数据类型定义别名。
- typedef语句的一般格式是：

  typedef 现有的类型名 新的类型名;
- 例如：typedef int INTEGER;
- 定义类型名INTEGER是int的别名。在该语句之后的程序中，标识符INTEGER和保留字int的作用相同。例如：

  INTEGER i,j;
- 编译时系统将把它当做int i,j;来处理，也就是当i,j定义为整型变量。

附件：课程教材 （2012-2013第2学期）

- 《C语言程序设计（第2版）》
- 清华大学出版社，黄保和，江弋 编著
- 版次：2011年10月第2版
- ISBN:978-7-302-26972-4
- 定价：35元

附件：课程和班级网站 （2012-2013第2学期）

- 课程介绍网站：
  http://dblab.xmu.edu.cn/node/124
- 班级网站：
  http://dblab.xmu.edu.cn/node/347

附件：课程教师和助教（2012-2013第2学期）

主讲教师：林子雨

单位：厦门大学信息科学与技术学院计算机科学系
办公地点：福建省厦门市思明区厦门大学海韵园
E-mail:ziyulin@xmu.edu.cn
个人主页：http://www.cs.xmu.edu.cn/linziyu

助教：刘颖杰

单位：厦门大学计算机科学系2012级硕士研究生
E-mail: 376339705@qq.com
手机：18020761782

Department of Computer Science, Xiamen University, May 9, 2013