# **结构体**

**构造数据类型：用户自己建立的数据类型（自定义数据类型）。**

**C语言中的构造数据类型有：数组类型、结构体类型和共用体类型。**

## **定义和使用结构体变量**

### **什么是结构体？**

**C语言允许用户根据需要自己建立的由不同类型数据组成的组合型的数据类型，我们把它称之为结构体（struct）。**

### **为什么要用结构体？**

**在日常生活中有许多事物用单一的数据类型可能没办法完全表示出来，例如：学校要存储学生的学号、姓名、性别、年龄、成绩和家庭地址等信息，这些信息需要用不同的数据类型来存储，显然用我们一个普通的单一的数据类型是无法全部存储起来的，就比如说数组，我们常用它来存储一串连续的信息，但是它的数据类型单一，显然无法把学生的这些信息全部保存。**

**所以，结构体这种数据类型就诞生了，它能根据用户需要来更方便的存储各种各样的信息。**

### **结构体类型的声明和结构体变量的定义**

**（1）声明一个结构体类型的一般形式为：**

**struct 结构体类型名{成员列表}；**

**（2）定义一个结构体类型的变量：**

**可以在声明的时候直接定义结构体变量,也可以先声明后定义结构体变量。**

**例如：学生类结构体**

**struct student{**

**int id; //学生学号**

**char name[10]; //学生姓名**

**char sex[4]; //学生性别**

**int age; //学生年龄**

**int score; //学生成绩**

**char address[20]; //学生家庭住址**

**}S1,S2,S3; //声明时定义结构体变量S1,S2,S3**

**//这里定义了一个结构体，其中struct为结构体关键字，srtuct student为结构体类型名，id、name、sex、age、score、address为结构体成员名，S1、S2、S3为结构体变量名。**

**# 注意：一定要区分清楚什么是结构体类型名、结构体成员名和结构体变量名。**

**int main()**

**{**

**struct student S4,S5;//声明之后使用结构体类型名定义的结构体变量S4,S5**

**return 0;**

**}**

1. **不指定结构体类型名而直接定义结构体类型的变量**

**其一般形式为：**

**struct //这里缺省了结构体类型名**

**{**

**成员列表；**

**}变量名列表；**

**如：srtuct //这里没有规定结构体类型名**

**{**

**int id;**

**char name[10];**

**char sex[4];**

**int age;**

**int score;**

**char address[20];**

**}t1,t2,t3; //由于没有规定类型名，所以变量只能声明时定义**

**# 注意：以此方式定义结构体，由于没有结构体类型名，只能在声明时定义此结构体的变量，而不能再以此结构体类型名去定义其他变量了。（这种方式用得不多）**

### **结构体变量的初始化和引用**

**（1）结构体变量的初始化：**

**srtuct teacher{**

**int id;**

**char name[10];**

**char sex[4];**

**int age;**

**int score;**

**char address[20];**

**}t1={666,”小威”,”男”,18,0,”六星教育长沙总部”},t2,t3;**

**//这里给srtuct teacher类型的结构体变量t1赋初值（初始化）**

**//在定义结构体变量的同时给变量赋初值，应按次序给每一个成员或部分成员赋值**

**（2）结构体变量和成员的引用**

**相同结构体类型的结构体能够相互赋值：如有struct student s1,s2; 就可以有s1=s2;**

**不同结构体类型的结构体不能相互赋值：如有struct student s; struct teacher t; 不能有s=t;**

**也不能有student = teacher;**

**结构体成员引用符：’.’，结构体成员引用的一般格式：结构体变量名.成员名**

**如：s1.id=1;、s2.name、s3.score**

### **使用typedef关键字自定义类型名**

**typedef：简单来说就是用一个新的类型名代替原有的类型名。**

**如：typedef int INT; //这里用INT代替int,之后定义整型变量就可以用INT了**

**typedef的使用方法与#define INT int 类似，相当于给数据类型关键字取别名。**

**但是需要注意的是：typedef只能用于给数据类型关键字取别名，除此之外没有其他用途。**

**使用typedef取的别名也需要遵循标识符的命名规则**

**可以给一个数据类型取多个别名(没有意义)**

**如：typedef int INT,I,inter;**

**或者：typedef int INT;**

**typedef int I;**

**typedef一般使用在数据类型名比较长的情况下，**

**如：声明结构体**

**typedef struct student{**

**int id; //学生学号**

**char name[10]; //学生姓名**

**char sex[4]; //学生性别**

**int age; //学生年龄**

**int score; //学生成绩**

**char address[20]; //学生家庭住址**

**}S; //这里的S是代表struct student这个数据类型的别名，而不是结构体的变量名了。**

**我们可以用struct student定义此结构体类型的变量，如：struct student s1,s2;**

**也可以用struct student的别名S等于此结构体类型的变量，如：S s3,s4;**

**思考一下：结构体变量所占的内存大小怎么计算？(内存的对齐)**

## **用一个结构体类型作为另一个结构体类型的成员**

**struct test1**

**{**

**int x;**

**};**

**struct test2**

**{**

**int y;**

**struct test1 z;**

**}n;**

**通过结构体变量n引用成员变量x：n.z.x=10; //给成员变量x赋值为10**

## **结构体数组**

1. **结构体数组的定义**

**例如：struct student S[3];**

**//这是一个struct student类型的结构体数组，此数组中有三个struct student结构体类型的元素，分别为S[0],S[1],S[2]。**

1. **结构体数组元素的使用**

**使用结构体数组元素成员的一般格式：结构体数组名[数组下标].成员变量名;**

**例如：S[0].id=1;**

**//这里引用了struct student类型的结构体数组S的第1个元素S[0]的成员id，使其值等于1**

**printf(“%s\n”S[2].name);**

**//这里是输出struct student类型的结构体数组S的第3个元素S[2]的成员name的值**

**思考一下：结构体数组所占的内存大小怎么计算？**

**如有以下定义：**

**struct student s1[3],s2[3];**

**那么可以有 s1[0]=s2[1];吗？**

**可以有 s1=s2;吗？**

# **共用体**

## **一、什么是共用体？**

**共用体关键字：union**

**有时候想用同一段内存单元存放不同类型的变量。如：把一个整型变量、浮点型变量和字符型变量放在同一个内存单元中，它们在内存中所占字节数不同，但是共用同一段内存地址，也就是共用体了。**

## **二、共用体类型的声明和共用体变量的定义**

**共用体类型声明和定义的一般格式为：**

**union 共用体名**

**{**

**成员列表;**

**}变量列表;**

**如：union DATA**

**{**

**int a;**

**double b;**

**char c[10];**

**}d1={1},d2,d3; //可以对共用体变量进行初始化，但是初始化列表中只能有一个常量。**

**d1.a=10; d1.b=6.6; strcpy(d1.c,”abcdefg”); //这里分别给共用体成员赋值**

**#注意：由于共用体中的每个成员共用一段内存空间，所以共用体在同一时刻只能保存一个成员的值，也就是保存最后一个赋值的成员的值。**

## **共用体类型所占内存**

**由于共用体类型中所有成员是共用一段内存的，所以整个共用体类型所占内存是最大成员所占的内存空间（也存在内存对齐）。**

**如：union DATA**

**{**

**int a;**

**double b;**

**char c[10];**

**}d1,d2,d3;**

**printf(“%d\n”,sizeof(d1)); //这里输出共用体的内存大小为16（两个double的大小）**

# 枚举类型

## 枚举类型的概念

**如果一个变量只有几种可能的值，就可以定义为枚举类型，所谓的“枚举”就是指吧可能的值一一列举出来，枚举变量的取值范围只限于列举出来的值的范围内，也就是只能在值的集合内选择。**

## **枚举类型的声明**

**枚举类型关键字：enum**

**enum 类型名{枚举元素列表};**

**例如：enum Weekday{sun,mon,tue,wed,thu,fri,sat};**

**//以上声明了一个枚举类型的变量Weekday，用于表示星期，其变量的取值范围为星期天至星期六。**

## **三、枚举变量的定义**

**根据以上声明我们可以定义此枚举类型的变量。**

**例如：enum Weekday workday,weekend;**

**其中workday和weekend是此枚举类型的变量，其每个变量的取值范围为集合{sun,mon,tue,wed,thu,fri,sat}中的一个值。**

**例如：workday=mon; weekend=sun;**

**workday=abc; //错误，不存在枚举值abc**

**可以在声明时同时定义变量。**

**如：enum Weekday{sun,mon,tue,wed,thu,fri,sat}workday,weekend;**

**声明的枚举类型也可以没有类型名，但如果需要使用此枚举类型的话，就必须在声明时同时定义变量。**

**如：enum {sun,mon,tue,wed,thu,fri,sat}workday,weekend;**

**说明：**

**（1）在C语言中枚举类型的枚举元素是按照常量进行处理的，所以又称为枚举常量。不要因为它们是标识符而把它们当作变量来使用，在使用时是不能对它们赋值的。**

**例如：sun=0;mon=1;sun = 7; //错误，不能给枚举常量赋值**

1. **每一个枚举元素都代表一个整数，C语言编译器按照定义时的顺序默认它们的值为0，1，2，3，4，5，6……。在上面的声明中，sun的值为0，mon的值为1，tue的值为2，依次类推。**

**如果有赋值语句：workday=mon;**

**就相当于：workday=1;**

**枚举常量的引用和输出：**

**如：printf(“%d”,sun);//输出整数0**

**枚举变量的引用和输出：**

**如：workday=mon;**

**printf(“%d”,workday);//输出整数1**

1. **我们也可以人为的指定枚举元素的数值，在声明的枚举类型的时候显式指定。**

**例如：enum Weekday{sun=7,mon=1,tue,wed,thu,fri,sat}workday,weekend;**

**指定枚举常量sun的值为7，mon的值为1，之后的值顺序加1，也就是tue的值为2，sat的值为6。**

**由于枚举类型的值为整数，因此C99把枚举类型也作为整型数据的一种，即用户自己定义的整数类型。**

1. **枚举类型可以用来比较和判断。**

**例如：**

**if(workday==mon){……}**

**If(workday<=sat){……}**

**枚举类型的比较规则是按照其初始化时指定的整数大小来进行比较的。如果声明时没有人为指定，就按默认规则处理，即第一个枚举元素的值为0，第二个枚举元素的值为1，依次类推，所以mon>sun,sat>fri。**

**#简单的枚举类型应用举例：**

**根据你输入的数字输出对应星期几**