学生应知信息知识

〔语言简明教程（下）

狄登峰 主编

目　　录

第七章　结构体、联合和枚举

7.1　结构体的说明和定义

7.1.1　什么是结构体

前面的章节已讲过数组。数组是具有相同数据类型的数据所组成的集合体，它的引入为程序设计者带来了很大的方便。但是，在程序设计中，还经常遇到一些关系密切、数据类型不同的数据。对于这些数据，为了处理方便，经常把它们组织在一起，并为其取一个名字。这类数据在 Pascal 和 COBOL 语言中叫记录，而在 C 语言中叫结构体。

结构体通常是由不同类型的数据所组成的集合体。构成结构体的数据称之为结构体成员（或称结构体元素），每一个成员具有不同的名字和数据类型（特殊情况下，也可具有相同的数据类型）。为了处理方便，每个结构体都有一个名字，而所有的成员都组织在该名字之下。结构体的典型例子是学生的记录，它包括学号、姓名、性别、年龄、地址和电话号码等，这些数据的类型都可以不同。因此，结构体的每一个成员都是通过其名字来引用的，而不像数组是通过下标来引用。

结构体的引入为处理复杂的数据结构体提供了有力的手段，也为函数间传递一组不同数据类型的数据提供了方便。特别是对于数据结构体比较复杂的大型程序提

供了方便。

7.1.2 结构体的说明及结构体变量的定义

1．结构体的说明

由于结构体是由不同数据类型的数据所组成的集合体，它包含若干成员。因此，在使用结构体进行数据处理时，首先应对结构体的组成进行描述。这种描述称之为说明。结构体的说明实质上是宣布该结构体是由哪些成员所组成，以及成员的数据类型。

结构体说明的一般格式如下：

```
struct      结构体名
 {
结构体成员表列；
}
```

其中 struct 是保留字，“ struct 结构体名 ”称结构体类型标识符，或称结构体类型名。大括号中的结构体成员表列称结构体体。成员表包含若干成员，每一个成员都具有如下的形式：

```
数据类型标识符 结构体成员名；
```

例如：

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

其中，date 是结构体名。而 year、month、day、yearday 均是该结构体的成员。

有关结构体说明请注意如下几点：

- 结构体说明描述了结构体的组织形式，但在编译时并不为它分配存储空间。只是规定了一种特定的数据结构体类型及它所占用的存储模型。
- 结构体的成员可以是简单变量、数组、指针、结构体或联合等。
- 结构体说明可以在函数内部，也可以在函数外部。在内部说明的结构体，只在函数内部可见，在外部说明的结构体，从说明点到源文件尾之间的所有函数都可见。
- 结构体可以嵌套使用，即一个结构体也可以成为另一个结构体的成员。
- 结构体成员的名字可以同程序中的其他变量同名，二者不会相混。

2．结构体变量的定义

结构体变量同其他变量一样也必须先定义或说明，然后才能引用。在说明了一个结构体之后，就可定义该结构体的对象了。一个结构体的对象或实例称之为该结构体类型的结构体变量。可以采用如下三种方式定义结构体变量。

（1）在结构体说明的同时定义结构体变量

其一般形式如下：

```
［存储类型］ struct 结构体名
{
结构体成员表列；
} 结构体变量名表列；
```

其中，存储类型表示结构体变量的存储类型（下同）。

例如：

```
struct example
 {
char * name；
int age；
double salary；
char * address；
} guo，zhang，xiao，yang；
```

该例的结构体名为 example，**用它定义了** guo、zhang、xiao **和** yang **四个结构体变量，这四个变量具有相同的结构体组成，即具有相同的类型。这种定义方法的特点是：定义一次结构体变量之后，在该定义之后任何位置还可用该结构体类型来定义其他结构体变量。**

（2**）直接定义结构体变量**

其一般形式如下：

```
［存储类型］struct
 {
结构体成员表列；
} 结构体变量名表列；
```

例如：

```
struct
{
char * name;
int age;
double salary;
char * address;
}guo,zhang,xiao,yang;
```

这里定义的四个结构体变量与（1**）中定义的完全相**

同。这种定义方式的特点是：不能在别处用来另行定义别的结构体变量，要想定义就得将“ struct{…} ”这部分重新写。

（3）把定义和说明分开

其定义的一般形式如下：

［存储类型］ struct 结构体名 结构体变量表；

例如：

```
struct example
 {
char * name；
int age；
double salary；
char * address；
} ；
```

利用结构体类型“ struct example ”来定义结构体变量 guo、zhang、xiao 和 yang 的形式为：

```
struct example guo，zhang，xiao，yang；
```

这种定义方式的特点是：可把其结构说明部分作为文件存放起来，这样就可借助于“ # include ”语句把它复制到任何源文件中，用以定义同类型的其他结构体变量。

有关结构体变量的定义还需如下几点：

- 结构体变量分配内存，而结构体说明不分配内存。
- 结构体变量一般不用 register 型。
- 结构体变量的定义一定要在结构体说明之后或与结构体说明同时进行，对尚未说明的结构体类型，不能用它来定义结构体变量。
- 结构体变量占用实际内存的大小可用 sizeof () 运算来求出。

sizeof（）的一般格式如下：

sizeof（运算符）

其中，运算量可以是简单变量、数组、结构体或数据类型名。

例如：

sizeof（struct example）=15

sizeof（int）=2

- 结构体变量中的成员可以单独使用，其地位与一般变量相同。

7.2 结构体成员的引用与结构体变量的初始化

7.2.1 结构体成员的引用

在对结构体进行引用时，一般只能对其成员进行直接操作，而不准许结构体变量整体直接进行操作。

引用有两种方式：

（1）用指针方式

即定义一个指针，使它指向该结构体变量，这时就可用指针和成员名来引用结构体成员了。

（2）用结构体成员运算符方式。

用结构体成员运算符引用结构体成员的一般形式如下：

结构体变量名.成员名

其中“.”是结构体成员运算符，其结合性是自左至右。下面是用结构体成员运算符引用结构体成员的例子：

struct example

```
 {
long int idnumber;
char * name;
char address［100］;
} guo;
```

各成员的引用形式如下：

```
guo.idnumber
guo.name
guo.address 或 guo.address［i］
```

对于* guo.name **形式，由于运算符“ . ”优先于“ * ”所以，*** guo.name **等价于*（**guo.name**）。其含义是访问** guo.name **的目标变量。**

【例 7-1**】将某年某月某日转换成该年的第几天。**

这里我们把与时间有关的变量定义为结构体变量，其程序如下：

```
main()
{
  struct date                    / * 定义结构体 * /
   {
    int year;                    / * 定义结构体成员及其数据类型
* /
int month;
int day;
int yearday;
char * monthname;
}date1;                          / * 定义结构体变量 * /
int leap;
int i;
```

```
static int month[2][13] ={
{0,31,28,31,30,31,30,31,31,30,31,30,31},    /*定义静态数组 */
{0,31,29,31,30,31,30,31,31,30,31,30,31}};
printf(" \ n Please enter year、month and day:");        /* 提示输入年、月、日 */
scanf(" % d,% d,% d",& date1.year,&date1.month,&date1.day);   /*提示年、月、日*/
i =date1.year;
leap =0;
if(( i %4= =0)& & (i % 100! =0))leap=1;              /* 判断是否闰年 */
   if(i%400= =0)leap =1;
   date1.yearday = date1.day;
   for(i=1;i<date1.month;i+ +)
date1.yearday + = month[leap][i];printf("yearday = % d \ n",date1.yearday);
}
```

通过该例，可以看出结构体变量的成员，即成员变量，可以像一般变量一样参与各种操作。

7.2.2 结构体变量的初始化

所谓结构体变量的初始化，就是在定义结构体变量的同时，对其成员变量赋初值。结构体变量初始化的一般形式如下：

struct **结构体名 结构体变量名**= **｛初始数据｝；**

例如：

```
struct date
```

```
{
int year；
int month；
int day；
int yearday；
char monthname［3］；
}；
struct date date1=｛1979，10，29，"OCT"｝；
```

在对结构体变量初始化时，应注意如下几点：

- 只能对外部和静态结构体变量初始化，不能对 auto 类型的结构体变量初始化。
- 初始化数据之间用逗号（,）隔开。
- 初始化数据的个数要与成员的个数相同。
- 初始化数据的类型要与相应的成员变量的类型一致。

7.3 结构体数组

7.3.1 结构体数组的定义及初始化

1．结构体数组的定义

结构体数组是其元素都是具有相同结构体变量。同一般数组一样，结构体数组也必须先定义或说明，才能引用。结构体数组定义的一般格式如下：

［存储类型］struct 结构体名 结构体数组名［元素个数］［结构体数组名］［元素个数］，…］；

其中“ struct 结构体名 ”是已定义过的结构体类型。存储类型是结构体数组的存储类型。

2．结构体数组的初始化

一个外部的或静态的结构体数组在定义的同时可以初始化。其一般格式是在定义之后 紧跟一个用花括号括起来的一组初始数据：

［存储类型］struct 结构体名 结构体数组名［］=｛初始数据｝；

其中“ struct 结构体名 ”是预先说明的结构体类型。

或：

```
［存储类型］struct 结构体名
{
结构体成员表列；
｝结构体成员表［］=｛初始数据｝；
```

例如，初始化前面已说明过的 keytab ［］结构体数组：

```
struct key keytab［］= { {"shift"，0} ，
                        {"ctrl"，0} ，
                        {"del"，0} ，
                          …
                        {"pause"，0}
                      } ；
```

在初始化时，应当使初始数据的个数与结构体数组的元素个数以及每个数组元素的成员个数匹配。为了增强可读性，最好使每一个数组元素的初始数据都用花括号括起来。

7.3.2 结构体数组的应用举例

结构体数组适合于处理一组具有相同结构体类型的数据，下面举例说明其应用。

【例 7-2】统计全年级男、女生人数及 1977 年到 1980

年出生的人数（含 1977 年和 1980 年）。

```
main
{
  struct stud                          / * 定义学生登记表一结构体
数组 * /
  {
char name[30];
char sex;
int year;
  };
  struct stud grad[300];
  int malenumber,femalenumber,count;
  char sex[2],year[5];
  int i;
  countm = countf = count78 = 0;          / * 建立学生名册 * /
for(i =0;i <50;i + +)
{
readline(grade[i].name,sex,year);
grade[i].sex = sex[0];
grade[i].year = atoi(year);
}
for(i=0;i <300;i+ +)          / * 统计男女生人数和 1977 年到 1980
年出生人数* /
{
if((grade[i].year> =1977)& &(gradep[i].year< =1980))
count ++;
if(grade[i].sex = =' m '||grade[i].sex = =' m ')
malenumber + +;
```

```
else
femalenumber + ;
}
printf(" \ n male: % d",countm);          / * 输出统计结果* /
printf(" \ n 女生人数：% d",femalenumber);
printf(" \ n1977 年到 1980 年出生的人数：% d",count);
}
readlin(pname,* psex, * pyear)            / * 读记录函数 * /
char * pname, * psex, * pyear;
{
printf(" \ nPlease enter name:");         / * 读入记录的内容 * /
scanf(" % s",pname);
printf(" \ n Please enter sex:");
scanf(" % s",psex);
printf(" \ n Please enter birth year:");
scanf(" % s",pyear);
}
```

【例 7-3】用结构体数组处理通讯录。

```
# include < stdio.h >                     / * 嵌入头文件 * /
# include < stdlib.h >
# define MAXIMUM 20                       / * 宏定义 * /
struct stud
{
char name[30];                            / * 定义结构体 * /
int age;
char sex;
char telnumber[8];
char address[100];
```

```
}
main()
{
struct stud student[MAXIMUM];          /* 定义结构体数组 student[MAXIMUM]* /
char str[10];
int i;
for(i=0;i<MAXIMUM;i + +)            / * 给结构体成员初始化 * /
{
   printf(" \ n Please enter name:");
   gets(student[i].name);
   printf(" \ n Please enter sex:");
   gets(str);
   student[i].sex = str[0];
   printf(" \ n Please enter age?");
   gets(str);
  student[i].age = atoi(s);
   printf(" \ n Please enter telnumber:");
   gets(student[i].telnumber);
   printf(" \ n Please enter address:");
   gets(student[i].address);
}
printf(" \ n name age sex telnumber address \ n");
                    ----------------------------------------
printf("                                                \ n");
for(i =0; i<MAXIMUM;i + +)
{
printf(" % -14s %-7d",student[i].name,student[i].age);   /*输出结果 * /
```