第九章 结构体与共用体

【教学目标】

使用结构体解决简单问题。

【教学内容】

结构体的概念和特点；
结构体变量的定义及初始化；
结构体数组；
结构体指针；
共用体的概念、定义和使用方法；
枚举类型的概念、定义。

【教学重点和难点】

结构体类型的定义、结构体变量的初始化方法；
共用体变量的使用方法；
结构体指针的应用；
共用体变量的使用。

【问题的提出】

前面的课程我们学习了一些简单数据类型（整型、实型、字符型）的定义和应用，还学习了数组（一维、二维）的定义和应用，这些数据类型的特点是：当定义了某一特定数据类型，就限定该类型变量的存储特性和取值范围。对简单数据类型来说，既可以定义单个的变量，也可以定义数组。而数组的全部元素都具有相同的数据类型，或者说是相同数据类型的一个集合。

在日常生活中，我们常会遇到一些需要填写的登记表，如住宿表、成绩表、通讯地址等。在这些表中，填写的数据是不能用同一种数据类型描述的，在住宿表中我们通常会登记上姓名、性别、身份证号码等项目；在通讯地址表中我们会写下姓名、邮编、邮箱地址、电话号码、E-m a i l等项目。这些表中集合了各种数据，无法用前面学过的任一种数据类型完全描述，因此C语言引入一种能集中不同数据类型于一体的新的数据类型—结构体类型。结构体类型的变量可以拥有不同数据类型的成员，是不同数据类型成员的集合。

所谓共用体类型是指将不同的数据项组织成一个整体，它们在内存中占用同一段存储单元。

【教学要点】

1. 结构体变量的使用

【例 9.1】定义一个结构变量, 其中每个成员都从键盘接收数据, 然后对结构中的浮点数求和, 并显示运算结果, 同时将数据以文本方式存入一个名为 wage.dat 的磁盘文件中。

```
#include <stdio.h>
```

```
main()
{
      struct{                                          /*定义一个结构变量*/
      char name[8];
      int age;
      char sex[2];
      char depart[20];
      float wage1, wage2, wage3, wage4, wage5;
} a;
FILE *fp;
float wage;
char c='Y';
fp=fopen("wage.dat", "w");                    /*以只写方式创建一个文件*/
   while(c=='Y'||c=='y')
      {
          printf("\nName:");
          scanf("%s",a.name);                          /*输入姓名*/
          printf("Age:");
          scanf("%d",&a.age);                          /*输入年龄*/
          printf("Sex:");
          scanf("%s",a.sex);                           /*输入性别*/
          printf("Dept:");
          scanf("%s",a.depart);                        /*输入部门*/
          printf("Wage1:");
          scanf("%f",&a.wage1);                        /*输入工资*/
          printf("Wage2:");
          scanf("%f",&a.wage2);
          printf("Wage3:");
          scanf("%f",&a.wage3);
          printf("Wage4:");
          scanf("%f",&a.wage4);
          printf("Wage5:");
          scanf("%f",&a.wage5);
          wage=a.wage1+a.wage2+a.wage3+a.wage4+a.wage5;
    printf("The sum of wage is %6.2f\n", wage);              /*显示结果*/
    fprintf(fp, "%10s%4d%4s%30s%10.2f\n",a.name, a.age, a.sex, a.depart, wage);
                                                          /*结果写入文件*/
          while(1)
         {
              printf("Continue?<Y/N>");                   /*循环退出判断*/
              c=getchar();
              if(c=='Y'||c=='y'||c=='N'||c=='n')
               break;
         }
      }
fclose(fp);
```

}

说明:

结构体是一种新的数据类型, 因此结构体变量也可以象其它类型的变量一样赋值、运算, 不同的是结构体变量以成员作为基本变量。结构体变量成员的输入和输出必须采用各成员独立进行的形式，而不能将结构体变量以整体的形式输入和输出。

结构成员的表示方式为:

结构体变量.成员名

如果将"结构体变量.成员名"看成一个整体, 则这个整体的数据类型与结构中该成员的数据类型相同, 这样就可象以前章节所讲的基本变量那样使用。

2. 指向结构体类型数组的指针的使用

【例 9.2】.定义结构体数组，并且分别采用数组法，指针法和地址法输出数组的元素。

```
struct data                    /*定义结构体类型*/
{
int day,month,year;
};
struct stu                     /*定义结构体类型*/
{
char name[20];
long num;
struct data birthday;
};
main()
{
    int i;
    struct stu *p,student[4]={{"liying",1,1978,5,23},
                    {"wangping",2,1979,3,14},
                    {"libo" ,3,1980,5,6} ,
                    {"xuyan",4,1980,4,21}};
/*定义结构体数组并初始化*/
    p=student;
/*将数组的首地址赋值给指针p , p指向了一维数组student*/
    printf("\n1----Output name,number,year,month,day\n" );
    for(i=0;i<4;i++)
/*采用指针法输出数组元素的各成员*/
    printf("%20s%10ld%10d//%d//%d\n",(p+i)->name,(p+i)->num,
    (p+i)->birthday.year,(p+i)->birthday.month,(p+i)->birthday.day) ;
    printf("\n2----Output name,number,year,month,day\n" );
for(i=0;i<4;i++)
/*采用指针法输出数组元素的各成员*/
    printf("%20s%10ld%10d//%d//%d\n",(*(p+i)).name ,(*(p+i)).num,
    (*(p+i)).birthday.year ,(*(p+i)).birthday.month ,(*(p+i)).birthday.day) ;
    printf("\n3----Output name,number,year,month,day\n" );
    for(i=0;i<4;i++,p++)
```

```
/*采用指针法输出数组元素的各成员*/
    printf("%20s%10ld%10d//%d//%d\n",p->name,p->num,p->birthday.year,
    p->birthday.month,p->birthday.day);
    printf("\n4-----Output name,number,year,month,day\n" );
  for(i=0;i<4;i++)
/*采用数组法输出数组元素的各成员*/
printf("%20s%10ld%10d//%d//%d\n",(student+i)->name,(student+i)->num ,
(student+i)->birthday.year,(student+i)->birthday.month ,(student+i)->
birthday.day);
    p=student;
/*将数组的首地址赋值给指针p , p指向了一维数组student*/
    printf("\n5-----Output name,number,year,month,day\n" );
    for(i=0;i<4;i++)
/*  采用指针的数组描述法输出数组元素的各成员*/
    printf("%20s%10ld%10d//%d//%d\n",p[i].name,p[i].num,p[i].birthday.year,
    p[i].birthday.month,p[i].birthday.day);
}
```

说明:

p是指向一维结构体数组的指针，对数组元素的引用可采用三种方法。

1. 地址法

student+i和p+i均表示数组第i个元素的地址，数组元素各成员的引用形式为：(student+i)->num、(student+i)->num和(p+i)->name、(p+i)->num等。student+i和p+i与&student[i]意义相同。

2. 指针法

若p指向数组的某一个元素，则p++就指向其后续元素。对数组成员的引用描述为:(*p).name、(*p).num或者p->name,p->num,其中->称为指向运算符

3. 指针的数组表示法

若p=student，我们说指针p指向数组student，p[i]表示数组的第i个元素，其效果与student[i]等同。对数组成员的引用描述为: p[i].name、p[i].num等。

3. 结构体变量、结构体指针变量作函数参数

【例 9.3】定义函数，实现对结构体数组中年龄在 19 岁以下（含 19 岁）同学的成绩增加 10 分（结构体变量地址作为函数参数）。

```
struct student
{
   int num;
   char name[20];
   char sex;
   int age;
   float score;
};
   struct student stu[3]={{11302,"Wang",'F',20,483},
                {11303,"Liu",'M',19,503},
                {11304,"Song",'M',19,471.5}};
   print(struct student s)
```

```
{
    printf("%s,%d,%5.1f\n",s.name,s.age,s.score);
}
add10(struct student *ps)
{
    if(ps->age<=19)
    ps->score=ps->score+10;
}
main()
{
    struct student *p;
    int i;
    for(i=0;i<3;i++)
  print(stu[i]);
    for(i=0,p=stu;i<3;i++,p++)
  add10(p);
    for(i=0,p=stu;i<3;i++,p++)
  print(*p);
}
```

或：

```
for(i=0;  i<3;  i++)print(stu[i]);
for(i=0;  i<3;  i++)add10(&stu[i]) ;
for(i=0;  i<3;  i++)print(stu[i]);
```

说明：

函数print的形参s属于结构体类型，所以实参也用结构体类型stu[i]或*p。

函数add10的形参ps属于结构体指针类型，所以实参用指针类型&stu[i]或p。

【例 9.4】定义函数，实现对结构体数组中年龄在 19 岁以下（含 19 岁）同学的成绩增加 10 分（结构体变量作为函数参数）。

```
struct student
{
  int num;
  char name[20];
  char sex;
  int age;
  float score;
};
  struct student stu[3]={{11302,"Wang",'F',20,483},
            {11303,"Liu",'M',19,503},
            {11304,"Song",'M',19,471.5}};
  print(struct student s)
  {
    printf("%s,%d,%5.1f\n",s.name,s.age,s.score);
  }
struct student add10(struct student s)
  {
    if(s.age<=19)
    s.score=s.score+10;
    return s;
  }
```

```
main()
{
    struct student *p;
int i;
    for(i=0;i<3;i++)
  print(stu[i]);
    for(i=0,p=stu;i<3;i++,p++)
 stu[i]=add10(stu[i]);
    for(i=0,p=stu;i<3;i++,p++)
  print(stu[i]);
}
```

或：

```
for(i=0,p=stu;  i<3;  i++) print(*(p+i));
for(i=0,p=stu;i<3;i++,p++) *p=add10(*p);
for(i=0,p=stu;i<3;i++,p++) print(*p);
```

说明：

函数 add10 修改为返回值为结构体类型的函数，那么形参的传址就不必要了。主函数调用 add10 时，将返回值赋值给结构体数组元素。

4. 结构体与联合体的联系与差别

定义一个镶嵌结构体成员的共用体结构，分析其内存使用情况。

```
【例9.5】main()
{
  union{                                          /*定义一个联合*/
  int i;
  struct{                                      /*在联合中定义一个结构*/
  char first;
  char second;
       }half;
  }number;
  number.i=0x4241;                                /*联合成员赋值*/
  printf("%c%c\n", number.half.first, number.half.second);
  number.half.first='a';                          /*联合中结构成员赋值*/
  number.half.second='b';
  printf("%x\n", number.i);
  getch();
}
```

说明：

结构和联合有下列区别:

1. 结构体和联合体都是由多个不同的数据类型成员组成, 但在任何同一时刻, 联合中只存放了一个被选中的成员, 而结构的所有成员都存在。
2. 对于联合的不同成员赋值, 将会对其它成员重写, 原来成员的值就不存在了, 而对于结构的不同成员赋值是互不影响的。

上例输出结果为:

AB

6261

从上例结果可以看出: 当给 i 赋值后, 其高八位和低八位也就是 first 和 second 的值; 当给 first 和 second 赋字符后, 这两个字符的 ASCII 码也将作为 i 的低八位和高八位。

【例 9.6】定义一个镶嵌结构体成员的共用体结构，分析其内存使用情况。

```
struct data
{
int a1;
int a2;
int a3;
} ;
union dig
{
struct data a;
char byte[6];
} ;
main()
{
union dig unit;
int i;
printf("enter a1:\n");
scanf("%d",&unit.a.a1);
printf("enter a2:\n");
scanf("%d",&unit.a.a2);
printf("enter a3:\n");
scanf("%d",&unit.a.a3);
printf("a1=%d a2=%d a3=%d\n", unit.a.a1,unit.a.a2,unit.a.a3);
for(i=0;i<6;i++)
printf("%d,",unit.byte[i]);
printf("\n");
}
```

运行程序：

```
enter a1:
600
enter a2:
300
enter a3:
100
a1=600 a2=300 a3=100
88,2,44,1,100,0
```

说明：

int型数据在内存中占2个字节， char型数据在内存中占1个字节。

从程序的输出结果来看， 600占两个字节，由第0、1字节构成，即2×256+88=800。

300同样占两个字节，由第2、3字节构成， 1×256+44=400，

100由第4、5字节构成， 0×256+100=100.

5. 枚举类型

【例9.7】定义两个枚举类型变量，分析其每个枚举元素的值

```
enum{black,blue,red,green} color;
enum{mon,thu=4,fri,sat}day;
main()
{
for(color=black;color<=green;color++)
      printf("%3d",color);
printf("\n");
for(day=mon;day<=sat;day++)
      printf("%3d",day);
}
```

说明：

理解枚举类型的要点是，每一个用符号表示的枚举数据代表一个整数值，且每个符号的取值都比前面的符号大，第一个符号的默认值是0。每一个符号都可用于任何使用整型值的场合，也就是说每一个符号都可作为整型常量是使用。在枚举变量color的枚举表中，枚举元素的值依次为0，1，2，3，因此第1条for语句通过枚举变量color循环输出0，1，2，3。枚举变量day的枚举表中,由于thu的值为4。因此fri和sat的值分别为5和6；第2条for语句依次输出0，1，2，3，4，5，6。

6. 链表

【例9.8】创建一个存放正整数（输入负数做结束标志）的单链表，并打印输出。

```
#include <stdlib.h>                          /*包含*含malloc()  的头文件*/
#include <stdio.h>
struct node                                          /*链表节点的结构*/
{
int num;
struct node *next;
};
main()
{
struct node *creat();                                    /*函数声明*/
void print2();
struct node *head;                                   /*  定义头指针*/
clrscr();
head=NULL;                                           /*  建一个空表*/
head=creat(head);                                    /*  创建单链表*/
print2(head);                                         /*打印单链表*/
}
struct node creat(struct node head) /*函数返回的是与节点相同类型的指针*/
{
struct node *p1,*p2;
p1=p2=(struct node*) malloc(sizeof(struct node));            /*申请新节点*/
scanf("%d",&p1->num);                                /*  输入节点的值*/
p1->next=NULL;                                   /*  将新节点的指针置为空*/
while(p1->num>0)                                  /*  输入节点的数值大于0 */
{
```

```
if (head==NULL) head=p1;                          /*  空表，接入表头*/
else p2->next=p1;                                 /*  非空表，接到表尾*/
p2=p1;
p1=(struct node )malloc(sizeof(struct node));     /*申请下一个新节点*/
scanf("%d",&p1->num);                             /*输入节点的值*/
}
p2->next=NULL;
return head;
}
void print2(struct node *head)     /*输出以he a d  为头的链表各节点的值*/
{
struct node *temp;
temp=head;                                        /*取得链表的头指针*/
while (temp!=NULL)                                /*只要是非空表*/
{
printf("%-6d",temp->num);                         /*输出链表节点的值*/
temp=temp->next;                                  /*跟踪链表增长*/
}
}
```

说明：

在链表的创建过程中，链表的头指针是非常重要的参数。因为对链表的输出和查找都要从链表的头开始，所以链表创建成功后，要返回一个链表头节点的地址，即头指针。

运行程序：

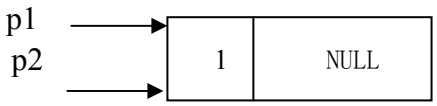
1 2 3 4 5 6 7 –9

1 2 3 4 5 6 7

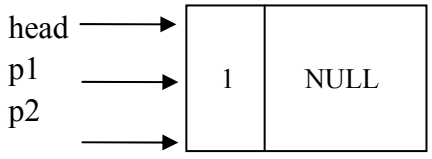
链表的创建过程用图示如下：

第一步，创建空表： head ⟶ N U L L

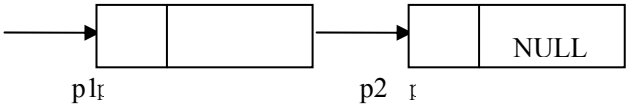
第二步，申请新节点：



第三步，若是空表，将新节点接到表头：

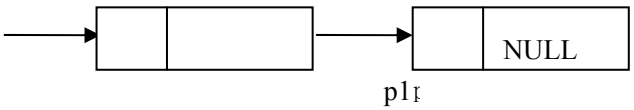


若是非空表，head

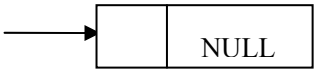


p2->next=p1

第四步，p2=p1: head…



第五步，申请新节点：p1



若数值为负，则结束；否则转到第三步。

【小结】

简单的来说，结构体就是一个可以包含不同数据类型的一个结构，它是一种可以自己定义的数据类型，在实际问题中，一组数据往往具有不同的数据类型。例如， 在学生登记表中，姓名应为字符型；学号可为整型或字符型； 年龄应为整型；性别应为字符型；成绩可为整型或实型。 显然不能用一个数组来存放这一组数据。 因为数组中各元素的类型和长度都必须一致，以便于编译系统处理。为了解决这个问题， C 语言中给出了另一种构造数据类型——结构体。 结构体是一种构造类型，这种类型的变量可以拥有不同数据类型的成员，即不同数据类型成员的集合，从而为解决具有这种数据类型的问题提供了方便。