计算机基础程序设计丛书

C 语言程序设计题典

李春葆　编著

目　录

第 9 章　结构体

§9.1　本章摘要

9.1.1　知识点

1. 结构体的概念

- 结构体定义
- 结构体变量说明
- 引用结构中成员
- 结构变量的初始化

2. 结构体与数组的关系

- 数组作为结构体成员
- 结构体数组

3. 结构体与指针的关系

- 指针作为结构体成员
- 结构体指针
- 通过指针引用结构体成员

4. 结构体的应用

- 单链表的基本运算

9.1.2　内容概要

1．结构体类型的定义

结构体是由基本数据类型构成的、并用一个标识符来命名的各种变量的组合。结构体中可以使用不同的数据类型。结构体名是结构体的标识符不是变量名。其定义方法为：

```
struct结构体名 {
    类型  变量名;
    类型  变量名;
    …
}
```

2．结构体变量的定义

结构体也是一种数据类型，可以使用结构体变量，因此，结构体变量也可以像其他类型的变量一样赋值和进行相应的运算，不同的是结构体变量以成员作为基本变量。在使用结构体变量时要先对其定义。

结构体变量定义的几种方式如下：

```
struct  结构体名 {
    类型 变量名;
    类型 变量名;
        …
} 结构体变量名;
```

或

```
struct  结构体名  结构体变量名;
```

例如，以下语句定义了一个student的结构体：

```
struct student {
    int no;
    char name[10];
    int sex;
    char class[5];
}
```

注意：如果需要定义多个具有相同形式的结构体变量时，先作结构体说明，再使用后者来定义结构体变量比较方便。结构体可以引用自身，即当在一个结构体中有一个成员是指向本结构体的指针时，可以把若干个相同的结构体存储单元连在一起，用以建立如链表、树、图等各种数据结构。

3．结构体成员

构成结构体的每一个类型变量称为结构体成员，它像数组的元素一样，．但数组中元

素是以下标来访问的，而结构体是按变量名字来访问成员的。结构体变量成员的引用方式如下：

结构体变量.成员名

4．结构体数组的定义和引用

结构体数组就是具有相同结构体类型的变量集合。如：

```
struct student st[40];
```

其中，student是前面定义的结构体，st是一个包含40个元素的结构体数组。
结构体数组成员的访问是以数组元素为结构体变量的，其形式如下：

结构数组元素.成员名

例如：

```
st[0].no、st[1].name等
```

5. 结构体数据指针变量的定义和引用

结构体指针是指向结构体的指针。其定义方式如下：

struct 结构体 *结构体指针变量；

结构体指针是由一个加在结构体变量名前的“*”操作符来定义，例如用前面已说明的结构体定义一个结构体指针如下：

```
struct student *sp;
```

使用结构体指针对结构体成员的访问，与结构体变量对结构体成员的访问在表达方式上有所不同。结构体指针对结构体成员的访问表示如下：

结构体指针变量名->成员名

或

(*结构体指针变量名).成员名

注意：(1)结构体指针是指向结构体的一个指针，即结构体中第一个成员的首地址，因此在使用之前应该对结构体指针初始化，即分配整个结构体长度的字节空间，可用下面函数完成，例如使用以下命令：

```
sp=(struct student *)malloc(sizeof(struct student));
```

为sp分配一个大小为结构体长度的内存区域，然后将其首地址作为结构体指针返回。

（2）结构体变量名不是指向该结构体的地址，这与数组名的含义不同，因此若需要求结构体中第一个成员的首地址应该是：“&结构体变量名”。

6．指向结构体指针的数组

如同可以定义指向一般指针的数组一样，也可以定义指向结构体指针的数组。例如：

```
struct student *sp[40];
```

其中，sp是一个包含有40个元素的数组，每个元素是指向结构体student的指针。

7．指向结构体数组的指针

如同可以定义指向一般数组的指针一样，也可以定义指向结构体数组的指针。例如：

```
struct student (*sp)[40];
```

其中，sp是一个指针，指向包含有40个元素的数组，该数组的每个元素的类型为student。

§9.2 例题解析

【例9.1】 分析以下程序的运行结果。

```
#include <stdio.h>
void main()
{
   struct
   {
     int x;
     int y;
   } s[2]={{1,2},{3,4}},*p=s;
    printf("%d,",++p->x);
    printf("%d\n",(++p)->x);
}
```

【解】 由定义得：s[0].x=1，s[0].y=2，s[1].x=3，s[1].y=4。p指向s。在++p->x表达式中，->优先级高于++，p->x返回1，再执行++，故返回2。在(++p)->x表达式中，由于有括号，先执行++，p指向s[1]，(++p)->x=s[1]->x，故返回3。所以输出为：2,3。

【例9.2】 分析以下程序的运行结果。

```
#include <stdio.h>
struct stru
{
    int x;
   char c;
};
void func(struct stru b);
```

```
void main()
{
    struct stru a={10, 'x'};
   func(a);
   printf("%d,%c\n",a.x,a.c);
}
void func(struct stru b)
{
    b.x=20;
   b.c='y';
}
```

【解】 定义结构变量a时，a.x=10，a.y='x'，**调用**func()**函数时，由于是传值调用，实参不改变。所以输出为：**10,x。

【例9.3**】 分析以下程序的运行结果。**

```
#include <stdio.h>
struct st
{
    int x;
   int *y;
} *p;
int s[]={10,20,30,40};
struct st a[]={1,&s[0],2,&s[1],3,&s[2],4,&s[3]};
void main()
{
    p=a;
   printf("%d,",p->x);
   printf("%d,",(++p)->x);
   printf("%d,",*(++p)->y);
   printf("%d\n",++(*(++p)->y));
}
```

【解】 在定义结构数组a**之后，**a[0].x=1，a[0].y**指向**10，a[1].x=2，a[1].y**指向**20，a[2].x=3，a[2].y**指向**30，a[3].x=4，a[3].y**指向**40。p**指向**a。p->x=a[0].x=1；(++p)->x=a[1].x=2；*(++p)->y=*(a[2].y)=30；++(*(++p)->y)=++(a[3].y)=41。**所以输出为：**1,2,30,41。

【例9.4**】 编写一个程序，以结构变量存放日期，输入今天的日期，输出明天的日期。**

【解】 设计以下函数：leap()**用于判断指定的日期所在年份是否为闰年；**numdays()**返回指定日期所在月份的天数，其中用到**leap()**函数确定**2**月份的天数。在求明天日期时，考虑这些情况：今天日期不是月底的情况（肯定也不是年底），是年底的情况和非年底但是月底的情况，分别进行相应处理。**

本题程序如下：

```
#include <stdio.h>
struct ydate
{
```

```
    int day;
    int month;
    int year;
};
int leap(struct ydate d)  /* 判断d日期所在年份是否为闰年 */
{
    if ((d.year%4==0 && d.year%100!=0) || (d.year%400==0))
        return 1;
    else
        return 0;
}
int numdays(struct ydate d) /* 返回d日期所在月份的天数 */
{
    int day;
    static int daytab[]={31,28,31,30,31,30,31,31,30,31,30,31};
    if (leap(d) && d.month==2)
        day=29;
    else
        day=daytab[d.month-1];
    return day;
}
void main()
{
    struct ydate today,tomorrow;
    printf("日期格式为:年.月.日\n");
    printf("    今天是:");
    scanf("%d.%d.%d",&today.year,&today.month,&today.day);
    if (today.day!=numdays(today))  /* 不是月底的情况 */
    {
        tomorrow.year=today.year;
        tomorrow.month=today.month;
        tomorrow.day=today.day+1;
    }
    else if (today.month==12)      /* 是年底的情况 */
    {
        tomorrow.year=today.year+1;
        tomorrow.month=1;
        tomorrow.day=1;
    }
    else                          /* 非年底但是月底的情况 */
    {
        tomorrow.year=today.year;
        tomorrow.month=today.month+1;
        tomorrow.day=1;
    }
    printf("    明天是:%d.%d.%d\n\n",
```

```
        tomorrow.year,tomorrow.month,tomorrow.day);
}
```

本程序的执行结果如下：

```
日期格式为：年.月.日
    今天是：2001.7.31
    明天是：2001.8.1
```

【例9.5】　编写一个程序，输入n个学生的英语、语文和数学三门课程的成绩，然后计算平均分并输出。

【解】　本题使用结构嵌套，即在定义student结构时，其deg成员又是一个degree结构类型。嵌套结构变量的使用方法与非嵌套结构变量的使用方法相似。其程序如下：

```
#include <stdio.h>
#define N 5
struct degree
{
    float eng;
    float cult;
    float math;
   float avg;
};
struct student
{
    char name[10];
    struct degree deg;
};
void main()
{
    int i;
    struct student st[N];
    printf("输入学生成绩:\n");
    for (i=0;i<N;i++)
    {
        printf("第%d个学生姓名:",i+1);
        scanf("%s",st[i].name);
        printf("    英语,语文,数学:");

scanf("%f,%f,%f",&st[i].deg.eng,&st[i].deg.cult,&st[i].deg.math);
        st[i].deg.avg=(st[i].deg.eng+st[i].deg.cult+st[i].deg.math)/3;
    }
    printf("输出结果:\n");
    printf("姓名      英语  语文  数学  均分\n");
    for (i=0;i<N;i++)
    {
```

```
        printf("%-8s  %g    %g    %g    %g\n",st[i].name,st[i].deg.eng,
               st[i].deg.cult,st[i].deg.math,st[i].deg.avg);
    }
}
```

本程序的执行结果如下：

输入学生成绩：
第1上学生姓名：li
英语，语文，数学：86,88,89
第2个学生姓名：Zheng
英语，语文，数学：76,65,68
第3个学生姓名：Chen
英语，语文，数学：64,68,54
第4个学生姓名：Xu
英语，语文，数学：89,92,85
第5个学生姓名：Ma
英语，语文，数学：82,85,82
输出结果：

姓名	英语	语文	数学	均分
Li	86	88	89	87.6667
Zheng	76	65	68	69.6667
Chen	64	68	52	61.3333
Xu	89	92	85	88.6667
Ma	82	85	82	83

【例9.6】 编写一个程序，用户输入一串整数，以0标识结束，将用户输入的整数构成一个单链表并输出。

【解】 单链表的每个节点有一个data成员和一个next成员，前者存放数据，后者指向下一个节点。create()**用于创建一个单链表并返回第一个节点的指针，**disp()**用于输出单链表各节点的**data**成员值。本题程序如下：**

```
#include <stdio.h>
#include <malloc.h>
struct node
{
    int data;
    struct node *next;
};
struct node *create()
{
```

```
    int d;
    struct node *h=NULL,*r,*s;
    printf("创建单链表\n");
    while (1)
    {
        printf("  输入整数:");
       scanf("%d",&d);
        if (d==0) break;
        s=(struct node *)malloc(sizeof(node));
        s->data=d;s->next=NULL;
        if (h==NULL)
        {
            h=s;r=s;        /* h指向单链表第一个节点 */
        }
        else
        {
            r->next=s;r=s;   /* r始终指向最后一个节点 */
        }
    }
    return h;
}
void disp(struct node *p)
{
    if (p==NULL)
        printf("空表\n");
    else
        while (p!=NULL)
        {
            printf("%d ",p->data);
            p=p->next;
        }
    printf("\n");
}
void main()
{
    struct node *p;
    p=create();
    printf("输出单链表:\n   ");
    disp(p);
}
```

本程序的执行结果如下：

```
创建单链表
  输入整数：10
  输入整数：20
  输入整数：30
  输入整数：40
  输入整数：50
  输入整数：60
  输入整数：0
输出单链表：
   10   20   30   40   50   60
```

§9.3 习题实践

【题1】 分析以下程序的运行结果。

```
#include <stdio.h>
void main()
{
   struct
   {
      int x;
      int y;
   } s[2]={{1,2},{3,4}},*p=s;
    printf("%d,%d\n",++p->x,(++p)->x);
}
```

【题2】 分析以下程序的运行结果。

```
#include <stdio.h>
void main()
{
    struct complx
   {
        int x,y;
    } cnum[2]={1,3,2,7};
   printf("%d\n",cnum[0].y/cnum[0].x*cnum[1].x);
}
```

【题3】 分析以下程序的运行结果。

```
#include <stdio.h>
void main()
{
    struct st
    {
         int n;
        struct st *next;
    };
    struct st a[3],*p;
    a[0].n=5;a[0].next=&a[1];
    a[1].n=7;a[1].next=&a[2];
    a[2].n=9;a[2].next='\0';
    p=&a[0];
    printf("%d,",p->n);
    printf("%d,",p->n++);
    printf("%d,",p->next->n);
    printf("%d\n",++p->n);
}
```

【题4】　分析以下程序的运行结果。

```
#include <stdio.h>
struct stru
{
    int x;
    char c;
};
void func(struct stru *b);
void main()
{
    struct stru a={10, 'x'};
    func(&a);
    printf("%d,%c\n",a.x,a.c);
}
void func(struct stru *b)
{
    b->x=20;
    b->c='y';
}
```

【题5】　分析以下程序的运行结果。

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

```
struct s
{
    int x,*y;
};
int data[5]={10,20,30,40,50};
struct s array[5]={100,&data[0],200,&data[1],
                300,&data[2],400,&data[3],500,&data[4]};
void main()
{
    int i=0;
    struct s s_var;
    s_var=array[0];
    printf("%d,",s_var.x);
   printf("%d,",*s_var.y);
   printf("%d,",array[i].x);
   printf("%d,",*array[i].y);
   printf("%d,",++array[i].x);
   printf("%d,",++*array[i].y);
   printf("%d,",array[++i].x);
   printf("%d,",*++array[i].y);
   printf("%d,",(*array[i].y)++);
   printf("%d,",*(array[i].y++));
   printf("%d,",*array[i].y++);
   printf("%d,",*array[i].y);
}
```

【题6】 分析以下程序的运行结果。

```
#include <stdio.h>
struct s
{
    int x,*y;
} *p;
int data[5]={10,20,30,40,50};
struct s array[5]={100,&data[0],200,&data[1],
                300,&data[2],400,&data[3],500,&data[4]};
void main()
{
    p=array;
    printf("%d,",p->x);
   printf("%d,",(*p).x);
   printf("%d,",*p->y);
   printf("%d,",*(*p).y);
   printf("%d,",++p->x);
   printf("%d,",(++p)->x);
   printf("%d,",p->x++);
   printf("%d,",p->x);
   printf("%d,",++(*p->y));
   printf("%d,",++*p->y);
```