

# 第 9 章

用户自己建立数据类型

# 定义和使用结构体变量

# 自己建立结构体类型

**struct 结构体名  
{成员表列};**

C语言允许用户自己建立由不同类型数据组成的组合型的数据结构，它称为**结构体**（structre）。

在程序中建立一个结构体类型：

num	name	sex	age	score	addr
10010	Li Fang	M	18	87.5	Beijing

```
struct Student
{
    int num;           //学号为整型
    char name[20];     //姓名为字符串
    char sex;          //性别为字符型
    int age;           //年龄为整型
    float score;       //成绩为实型
    char addr[30];     //地址为字符串
};                    //注意最后有一个分号
```

结构体类型的名字是由一个关键字**struct**和结构体名组合而成的。结构体名由用户指定，又称“结构体标记”（structure tag）。

花括号内是该结构体所包括的子项，称为结构体的成员（member）。对各成员都应进行类型声明，即 **类型名 成员名;**

“成员表列”（member list）也称为“域表”（field list），每一个成员是结构体中的一个域。成员名命名规则与变量名相同。

# 自己建立结构体类型

- (1) 结构体类型并非只有一种，而是可以设计出许多种结构体类型，各自包含不同的成员。
- (2) 成员可以属于另一个结构体类型。

num	name	sex	age	birthday			addr
				month	day	year	

```
struct Date //声明一个结构体类型 struct Date
{
    int month; //月
    int day; //日
    int year; //年
};
```

```
struct Student //声明一个结构体类型 struct Student
{
    int num;
    char name[20];
    char sex;
    int age;
    struct Date birthday; //成员birthday属于struct Date类型
    char addr[30];
};
```

# 定义结构体类型变量

1. 先声明结构体类型，再定义该类型的变量

```
struct Student
{
    int num;           //学号为整型
    char name[20];     //姓名为字符串
    char sex;          //性别为字符型
    int age;            //年龄为整型
    float score;       //成绩为实型
    char addr[30];     //地址为字符串
};                    //注意最后有一个分号
```

```
struct Student student1, student2;
```

结构体类型名      结构体变量名

2. 在声明类型的同时定义变量

```
struct Student
{
    int num;
    char name[20];
    char sex;
    int age;
    float score;
    char addr[30];
} student1, student2;
```

struct 结构体名

{

成员表列

} 变量名表列;

3. 不指定类型名而直接定义结构体类型变量

```
struct
{
    成员表列
} 变量名表列;
```

sutdent1:	10001	Zhang Xin	M	19	90.5	Shanghai
student2:	10002	Wang Li	F	20	98	Beijing

# 定义结构体类型变量

```
struct Student
{   int num;
    char name[20];
    char sex;
    int age;
    float score;
    char addr[30];
} student1, student2;
```

(1) 结构体类型与结构体变量是不同的概念，不要混淆。只能对变量赋值、存取或运算，而不能对一个类型赋值、存取或运算。在编译时，对类型是不分配空间的，只对变量分配空间。

```
student1.num = 1001;    // 正确
```

```
student Student = ...;  // 错误，等同于：int = ...;一样是错误的
```

(2) 结构体类型中的成员名可以与程序中的变量名相同，但二者不代表同一对象。

```
int num = 1000;
```

```
student1.num = 1001; // 结构体成员中的num与上面的num不同
```

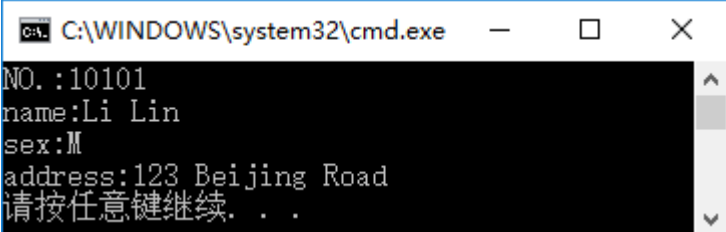
(3) 对结构体变量中的成员（即“域”），可以单独使用，它的作用与地位相当于普通变量。

```
int num = student1.num;
```

# 结构体变量的初始化和引用

【例9.1】 把一个学生的信息(包括学号、姓名、性别、住址)放在一个结构体变量中，然后输出这个学生的信息。

```
#include <stdio.h>
int main()
{
    struct Student      //声明结构体类型struct Student
    {
        long int num;    //以下4行为结构体的成员
        char name[20];
        char sex;
        char addr[20];
    } a={10101, "Li Lin", 'M', "123 Beijing Road"}; //定义结构体变量a并初始化
    printf("NO. :%ld\nname:%s\nsex:%c\naddress:%s\n", a.num, a.name, a.sex, a.addr);
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
NO. :10101
name:Li Lin
sex:M
address:123 Beijing Road
请按任意键继续. . .
```

# 结构体变量的初始化和引用

- (1) 在定义结构体变量时可以对它的成员初始化。初始化列表是用花括号括起来的一些常量，这些常量依次赋给结构体变量中的各成员。

**注意**

对结构体变量初始化，不是对结构体类型初始化

- (2) 可以引用结构体变量中成员的值，引用方式为 **结构体变量名. 成员名**

```
student1.num=10010;
```

/\* 已定义了 student1 为 student 类型的结构体变量，则 student1.num 表示 student1 变量中的 num 成员，即 student1 的 num(学号) 成员 \*/

“.” 是成员运算符，它在所有的运算符中优先级最高，因此可以把 student1.num 作为一个整体来看待，相当于一个变量。

```
printf(" %s\n" , student1); // 企图用结构体变量名输出所有成员的值
```



不能企图通过输出结构体变量名来达到输出结构体变量所有成员的值。只能对结构体变量中的各个成员分别进行输入和输出。



# 结构体变量的初始化和引用

- (3) 如果成员本身又属一个结构体类型，则要用若干个成员运算符，一级一级地找到最低的一级的成员。只能对最低级的成员进行赋值或存取以及运算。

```
student1.num=10010;           //结构体变量student1中的成员num  
student1.birthday.month=6;    //结构体变量student1中的成员birthday中的成员month
```

- (4) 对结构体变量的成员可以像普通变量一样进行各种运算（根据其类型决定可以进行的运算）。

```
student2.score=student1.score; //赋值运算  
sum=student1.score+student2.score; //加法运算  
student1.age++;                //自加运算
```

- (5) 同类的结构体变量可以互相赋值。

```
student1=student2; //假设student1和student2已定义为同类型的结构体变量
```

- (6) 可以引用结构体变量成员的地址，也可以引用结构体变量的地址（结构体变量的地址主要用作函数参数，传递结构体变量的地址）。但不能用以下语句整体读入结构体变量。

```
scanf("%d",&student1.num); //输入student1.num的值  
printf("%o",&student1); //输出结构体变量student1的起始地址
```

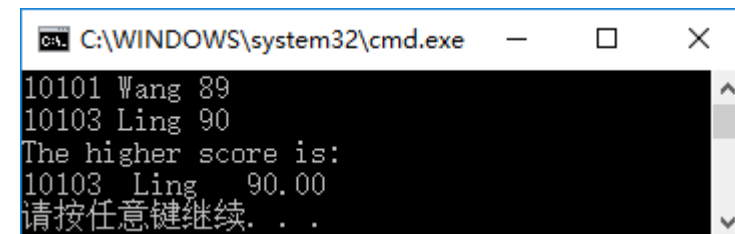
```
scanf("%d,%s,%c,%d,%f,%s\n",&student1);
```



# 结构体变量的初始化和引用

【例9.2】输入两个学生的学号、姓名和成绩，输出成绩较高的学生的学号、姓名和成绩。

```
#include <stdio.h>
int main()
{
    struct Student      //声明结构体类型struct Student
    {
        int num;
        char name[20];
        float score;
    } student1, student2; //定义两个结构体变量student1, student2
    scanf("%d%s%f", &student1.num, student1.name, &student1.score); //输入学生1的数据
    scanf("%d%s%f", &student2.num, student2.name, &student2.score); //输入学生2的数据
    printf("The higher score is:\n");
    if(student1.score > student2.score)
        printf("%d %s %6.2f\n", student1.num, student1.name, student1.score);
    else if(student1.score < student2.score)
        printf("%d %s %6.2f\n", student2.num, student2.name, student2.score);
    else
    {
        printf("%d %s %6.2f\n", student1.num, student1.name, student1.score);
        printf("%d %s %6.2f\n", student2.num, student2.name, student2.score);
    }
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
10101 Wang 89
10103 Ling 90
The higher score is:
10103 Ling 90.00
请按任意键继续...
```

# 使用结构体数组

# 定义结构体数组

【例9.3】有3个候选人，每个选民只能投票选一人，要求编一个统计选票的程序，先后输入被选人的名字，最后输出各人得票结果。

name	count
Li	0
Zhang	0
Sun	0

```
C:\WINDOWS\system32\cmd.exe
Li
Li
Sun
Zhang
Zhang
Sun
Li
Sun
Zhang
Li

Result:
  Li:4
  Zhang:3
  Sun:3
请按任意键继续. . .
```

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

struct Person                                //声明结构体类型struct Person
{
    char name[20];                          //候选人姓名
    int count;                              //候选人得票数
} leader[3]={"Li", 0, "Zhang", 0, "Sun", 0}; //定义结构体数组并初始化

int main()
{
    int i, j;
    char leader_name[20]; //定义字符数组
    for(i=1; i<=10; i++)
    {
        scanf("%s", leader_name); //输入所选的候选人姓名
        for(j=0; j<3; j++)
            if(strcmp(leader_name, leader[j].name)==0) leader[j].count++;
    }
    printf("\nResult:\n");
    for(i=0; i<3; i++)
        printf("%5s:%d\n", leader[i].name, leader[i].count);
    return 0;
}
```

# 定义结构体数组

(1) 定义结构体数组一般形式是

① **struct 结构体名  
{成员表列} 数组名[数组长度];**

```
struct Person
{
    char name[20];
    int count;
} leader[3];
```

② 先声明一个结构体类型，然后再用此类型定义结构体数组

**结构体类型 数组名[数组长度];**

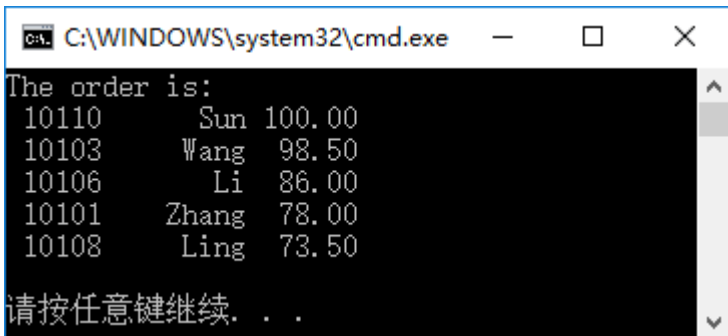
```
struct Person
{
    char name[20];
    int count;
}
struct Person leader[3]; //leader是结构体数组名
```

(2) 对结构体数组初始化的形式是在定义数组的后面加上: **= {初值表列};**

```
struct Person leader[3]= {"Li", 0, "Zhang", 0, "Sun", 0};
```

# 结构体数组的应用举例

【例9.4】有n个学生的信息(包括学号、姓名、成绩)，要求按照成绩的高低顺序输出各学生的信息。



```
C:\WINDOWS\system32\cmd.exe
The order is:
10110    Sun 100.00
10103    Wang 98.50
10106    Li 86.00
10101    Zhang 78.00
10108    Ling 73.50
请按任意键继续. . .
```

```
#include <stdio.h>
struct Student //声明结构体类型struct Student
{
    int num;
    char name[20];
    float score;
};
int main()
{
    struct Student stu[5]={ {10101, "Zhang", 78}, {10103, "Wang", 98.5}, {10106, "Li", 86},
    {10108, "Ling", 73.5}, {10110, "Sun", 100}}; //定义结构体数组并初始化
    struct Student temp; //定义结构体变量temp，用作交换时的临时变量
    const int n=5; //定义常量n
    int i, j, k;
    printf("The order is:\n");
    for(i=0; i<n-1; i++) // 选择法排序（降序）
    {
        k=i;
        for(j=i+1; j<n; j++)
            if(stu[j].score>stu[k].score) k=j; //进行成绩的比较
        temp=stu[k]; stu[k]=stu[i]; stu[i]=temp; //stu[k]和stu[i]元素互换
    }
    for(i=0; i<n; i++)
        printf("%6d %8s %6.2f\n", stu[i].num, stu[i].name, stu[i].score);
    printf("\n");
    return 0;
}
```

# 结构体指针



# 结构体指针

---

所谓结构体指针就是指向结构体变量的指针，一个结构体变量的起始地址就是这个结构体变量的指针。如果把一个结构体变量的起始地址存放在一个指针变量中，那么，这个指针变量就指向该结构体变量。



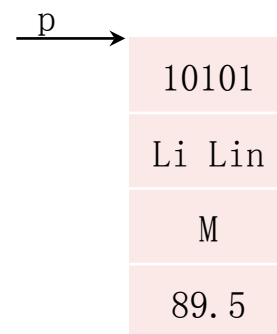
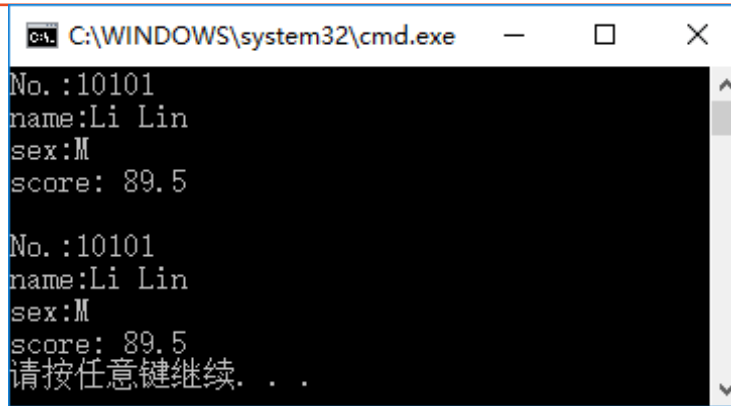


(\*p). num也可表示为p->num

# 指向结构体变量的指针

【例9.5】通过指向结构体变量的指针变量输出结构体变量中成员的信息。

```
#include <stdio.h>
#include <string.h>
int main()
{
    struct Student //声明结构体类型struct Student
    {
        long num;
        char name[20];
        char sex;
        float score;
    };
    struct Student stu_1; //定义struct Student类型的变量stu_1
    struct Student *p; //定义指向struct Student 类型数据的指针变量p
    p=&stu_1; //p指向stu_1
    stu_1.num=10101; //对结构体变量的成员赋值
    strcpy(stu_1.name, "Li Lin"); //用字符串复制函数给stu_1.name赋值
    stu_1.sex='M';
    stu_1.score=89.5;
    printf("No. :%ld\nname:%s\nsex:%c\nscore:%5.1f\n", stu_1.num, stu_1.name, stu_1.sex, stu_1.score); //输出结果
    printf("\nNo. :%ld\nname:%s\nsex:%c\nscore:%5.1f\n", (*p).num, (*p).name, (*p).sex, (*p).score);
    return 0;
}
```



如果p指向一个结构体变量stu，以下3种用法等价：

① stu. 成员名

stu.num

② (\*p). 成员名

(\*p). num

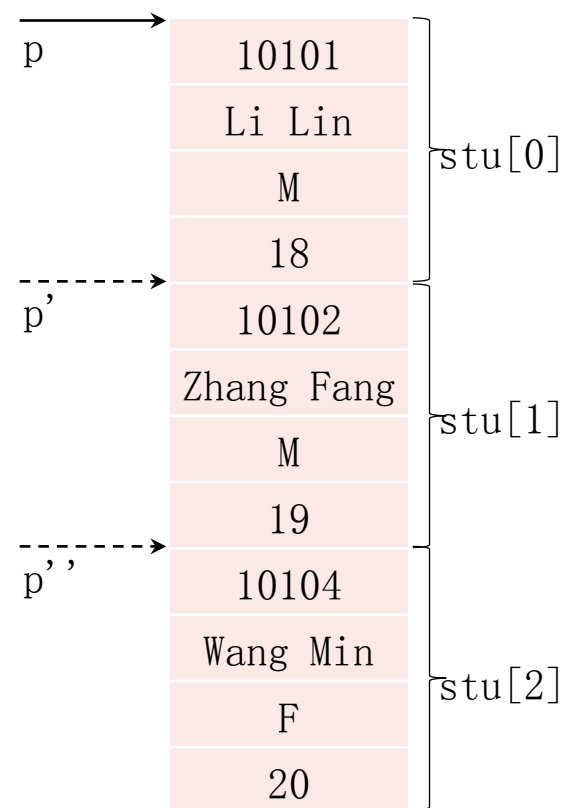
③ p->成员名

p->num

# 指向结构体数组的指针

【例9.6】有3个学生的信息，放在结构体数组中，要求输出全部学生的信息。

```
#include <stdio.h>
struct Student          //声明结构体类型struct Student
{
    int num;
    char name[20];
    char sex;
    int age;
};
struct Student stu[3]={10101,"Li Lin",'M',18},{10102,"Zhang Fang",'M',19},{10104,"Wang
Min",'F',20}};
//定义结构体数组并初始化
int main()
{
    struct Student *p;    //定义指向struct Student结构体变量的指针变量
    printf(" No. Name  sex age\n");
    for (p=stu;p<stu+3;p++)
        printf("%5d %-20s %2c %4d\n",p->num, p->name, p->sex, p->age);    //输出结果
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
No. Name          sex age
10101 Li Lin      M  18
10102 Zhang Fang  M  19
10104 Wang Min    F  20
请按任意键继续. . .
```

# 用结构体变量和结构体变量的指针作函数参数

---

将一个结构体变量的值传递给另一个函数，有3个方法：

(1) 用结构体变量的成员作参数。

例如，用`stu[1].num`或`stu[2].name`作函数实参，将实参值传给形参。用法和用普通变量作实参是一样的，属于“**值传递**”方式。应当注意实参与形参的类型保持一致。

(2) 用结构体变量作实参。

用结构体变量作实参时，采取的也是“**值传递**”的方式，将结构体变量所占的内存单元的内容全部按顺序传递给形参，形参也必须是同类型的结构体变量。在函数调用期间形参也要占用内存单元。这种传递方式在空间和时间上开销较大，如果结构体的规模很大时，开销是很可观的。此外，由于采用值传递方式，如果在执行被调用函数期间改变了形参（也是结构体变量）的值，该值不能返回主调函数，这往往造成使用上的不便。因此一般较少用这种方法。

(3) 用指向结构体变量（或数组元素）的指针作实参，将结构体变量（或数组元素）的**地址**传给形参。

---

# 用结构体变量和结构体变量的指针作函数参数

【例9.7】有n个结构体变量，内含学生学号、姓名和3门课程的成绩。要求输出平均成绩最高的学生的信息(包括学号、姓名、3门课程成绩和平均成绩)。

```
C:\WINDOWS\system32\cmd.exe
请输入各学生的信息： 学号、姓名、三门课成绩：
10101 Li 78 89 98
10103 Wang 98.5 87 69
10106 Sun 88 76.5 89

成绩最高的学生是：
学号:10101
姓名:Li
三门课成绩: 78.0, 89.0, 98.0
平均成绩: 88.33
请按任意键继续...
```

```
#include <stdio.h>
#define N 3          //学生数为3
struct Student      //建立结构体类型
{
    int num;         //学号
    char name[20];   //姓名
    float score[3];  //3门课成绩
    float aver;      //平均成绩
};

int main()
{
    void input(struct Student stu[]); //函数声明
    struct Student max(struct Student stu[]); //函数声明
    void print(struct Student stu); //函数声明
    struct Student stu[N], *p=stu; //定义结构体数组和指针
    input(p); //调用input函数
    print(max(p)); //调用print函数, 以max函数的返回值作为实参
    return 0;
}

void input(struct Student stu[]) //定义input函数
{
    int i;
    printf("请输入各学生的信息： 学号、姓名、三门课成绩:\n");
    for(i=0; i<N; i++)
    {
        scanf("%d %s %f %f %f", &stu[i].num, stu[i].name,
```

```
&stu[i].score[0], &stu[i].score[1], &stu[i].score[2]);
    //输入数据

    stu[i].aver=(stu[i].score[0]+stu[i].score[1]+stu[i].score[2])/3.0; //求平均成绩
    }

    struct Student max(struct Student stu[]) //定义max函数
    {
        int i, m=0; //用m存放成绩最高的学生在数组中的序号
        for(i=0; i<N; i++)
            if(stu[i].aver>stu[m].aver) m=i;
        //找出平均成绩最高的学生在数组中的序号
        return stu[m]; //返回包含该生信息的结构体元素
    }

    void print(struct Student stud) //定义print函数
    {
        printf("\n成绩最高的学生是:\n");
        printf("学号:%d\n姓名:%s\n三门课成绩:%5.1f, %5.1f, %5.1f\n平均成绩:%6.2f\n", stud.num, stud.name, stud.score[0], stud.score[1], stud.score[2], stud.aver);
    }
}
```

# 结构体小结

```
#include <stdio.h>
#define N 100    // 常数, 估计最大学生数
struct student  // 建立结构体类型
{
    int num;      // 学号
    char name[20]; // 姓名
    float score[3]; // 3门课成绩
    float aver;    // 平均成绩
};

void input(struct student stu[], int n); // 函数声明
int max(struct student stu[], struct student maxStu[], int n);
void print(struct student stu);
void prints(struct student stu[], int n);
void swap(struct student *stu1, struct student *stu2);
void sort(struct student stu[], int n);
```

```
int main()
{
    struct student stu[N], *p=stu;
    int n;    // 实际学生数
    int i, maxNum;
    struct student maxStu[N];

    scanf("%d", &n);
    input(p, n);

    printf("平均成绩最高者: \n");
    maxNum = max(stu, maxStu, n);
    for(i = 0; i < maxNum; i++)
        print(maxStu[i]);

    printf("\n排序前: \n");
    prints(p, n);

    printf("\n排序后: \n");
    sort(stu, n);
    prints(p, n);
    return 0;
}
```

```
// 在输入时，同时计算了平均值
void input(struct student stu[],int n)    // 定义input 函数
{
    int i;
    for(i=0;i<n;i++)
    {
        scanf("%d%s%f%f%f",&stu[i].num,stu[i].name,&stu[i].score[0],&stu[i].score[1],&stu[i].score[2]);
        stu[i].aver=(stu[i].score[0]+stu[i].score[1]+stu[i].score[2])/n;
    }
}
```

```
void print(struct student stud)    // 定义print函数
{
    printf("学号:%d\n姓名:%s\n三门课成绩:%5.1f,%5.1f,%5.1f\n平均成绩:%6.2f\n",stud.num,stud.name,stud.score[0],stud.score[1],stud.score[2],stud.aver);
}

void prints(struct student stu[],int n) // 定义prints函数
{
    int i;
    for(i = 0; i < n; i++)    print(stu[i]);
}
```

```
// 平均成绩最高者，考虑最高者不仅一个，返回平均成绩最高者学生数
int max(struct student stu[], struct student maxStu[], int n)    // 定义max 函数
{
    // 也可以先降序排序，然后取前几名平均成绩一样的最高者。
    // 这里，首先计算最大的平均成绩，然后再提取最大成绩一样的学生。
    int i, m = 0, maxNum = 0;
    for(i = 0; i < n; i++)
    {
        if (stu[i].aver > stu[m].aver) m = i;
    }
    for(i = 0; i < n; i++)
    {
        if (stu[i].aver == stu[m].aver)
        {
            maxStu[maxNum++] = stu[i];
        }
    }
    return maxNum;
}
```

// 交换两个学生顺序

```
void swap(struct student *stu1, struct student *stu2)
{
    struct student tmp;
    tmp = *stu1; *stu1 = *stu2; *stu2 = tmp;
}
```

// 定义排序函数(选择法, 降序): 按平均值排序, 如果平均值相同, 按照课程顺序依次比较单科成绩

```
void sort(struct student stu[], int n)
{
    int i, j, k;
    for(i = 0; i < n-1; i++) {
        k = i;
        for(j = i + 1; j < n; j++)
        {
            // 善用&& || 关系
            if(stu[j].aver > stu[k].aver ||
                (stu[j].aver == stu[k].aver && stu[j].score[0] > stu[k].score[0]) ||
                (stu[j].aver == stu[k].aver && stu[j].score[1] > stu[k].score[1]) ||
                (stu[j].aver == stu[k].aver && stu[j].score[2] > stu[k].score[2]))
            { k = j; }
        }
        if(k != i) swap(&stu[i], &stu[k]);
    }
}
```