第9章 用户自己建立数据类型

- 为什么要建立用户类型？

  C语言内置数据类型无法满足实际需求。

  - 标量类型
    - 数值型、字符型、指针类型(派生类型)
    - 存储单一数值
  - 复合类型：数组类型(派生类型)
    - 可存储多个数值，但数组类型中的元素均相同
- 用户可建立哪些数据类型
  - 结构体(与数组对比)
    - 客观世界同类对象共有的属性，对象的值由所有属性的值共同体现
    - 有多个元素(成员)构成，占用连续的存储空间，元素可属不同类型
  - 共用体
    - 不同成员共享同一存储空间
  - 枚举
    - 所有可能的值是可逐一列举的

第9章 用户自己建立数据类型

9.1 定义和使用结构体变量

9.1.1 自己建立结构体类型

9.1.2 定义结构体类型变量

9.1.3 结构体变量的初始化和引用

9.1.1 自己建立结构体类型

- 用户自己建立的、由不同类型数据组成的复合型的数据结构，称为结构体(structure)。
  - ◆例如，学生信息包括：学号、姓名、性别、年龄、专业、住址等项，每个学生均包含这些信息，从而组成一个复合数据；通讯录中的联系人包括：姓名、电话、单位、与本人的关系等；
- 声明结构体类型的一般形式

```
struct identifier_opt {
    members_list;
};
```

可省略

声明结束，不可省略

其中，成员声明方式：

```
T1 m11, m12, ...;
T2 m21, m22, ...;
```

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

9.1.2 定义结构体类型变量

- 结构体是一种用户自定义的数据类型，当然可以不止一种，可以设计出很多种结构体类型，例如：
  - **struct Teacher、struct Date、……**
  - 不同的类型可以各自包含不同的成员
- 结构体类型变量的定义
  - 先声明结构体类型，再定义该类型的变量
    - **结构体类型名 变量名1,变量名2,……;**
    - **struct Student s1, s2;**
  - 声明结构体类型的同时定义变量
    - **struct Student{ … } s1, s2;**
  - 不指定结构体类型名而直接定义变量
    - **struct { … } s1, s2;**

结构体类型名

结构体类型变量

匿名结构体

9.1.2 定义结构体类型变量

➢ 注意事项：

(1) 结构体是一种用户自定义的数据类型，当然可以不止一种，可以设计出很多种结构体类型，例如：

struct Teacher、struct Date、……

◆各自包含不同的成员

(2) 结结构体变量的存储结构

◆构体变量和数组类似，其成员占用连续的存储空间，如**struct Student**，但编译器会进行对齐处理

num	name	sex	age	score	addr

(3) 结构体成员可以是另一结构体类型的变量

num	name	sex	age	birthdate			addr
				year	month	day	

结构体数据对齐，是指结构体内的各个数据对齐。在结构体中的第一个成员的首地址等于整个结构体的变量的首地址，而后的成员的地址随着它声明的顺序和实际占用的字节数递增。为了总的结构体大小对齐，会在结构体中插入一些没有实际意思的字符来填充（**padding**）结构体。

在结构体中，成员数据对齐满足以下规则：

a、结构体中的第一个成员的首地址也即是结构体变量的首地址。

b、结构体中的每一个成员的首地址相对于结构体的首地址的偏移量（**offset**）是该成员数据类型大小的整数倍。

c、结构体的总大小是对齐模数（对齐模数等于**#pragma pack(n)**所指定的**n**与结构体中最大数据类型的成员大小的最小值）的整数倍。

9.1.3 结构体变量的初始化和引用

- 引用结构体变量
  - ◆引用结构体变量和引用其他类型变量完全相同，单独引用结构体变量没有太大意义。
- 引用结构体成员
  - ◆通过成员运算符(.)引用结构体成员
  - ◆**s1.num, s1.name, s1.birthdate.year, …**
- 结构体变量初始化
  - ◆在定义结构体变量的同时初始化
    - ●类似于数组的初始化方式，使用集合中的元素依次初始化成员
    - ●**struct Student s1 = {101, “Lisi”};**
  - ◆通过引用结构体成员进行初始化
    - ●**struct Student s1;**
    - ●**s1.num = 101; strcpy(s1.name, “Lisi”);**

9.1.3 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、年龄、成绩、住址)放在一个结构体变量中，然后输出这个学生的信息。

- 分析：
  - ◆建立一个结构体类型，存储有关学生的各项信息
  - ◆用它定义结构体变量
  - ◆对该结构体变量初始化
  - ◆输出该结构体变量各成员的值

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


```
#include <stdio.h>
int main()
{
   struct Student s = {101, "James", 'M', 18,
   88.59, "D11 XY Road"};
   printf("NO.:%d name:%s sex:%c
   address:%s\n", a.num, a.name, a.sex,
   a.addr);
  return 0;
}
```

```
struct Student s;
scanf("%d %s %c %d %f %s", &s.num,
s.name, &s.sex, &s.age, &s.score, s.addr
```

```
s.num = 101; strcpy(s.name, "James");
s.sex = 'M'; s.age = 18; s.score = 88.59;
strcpy(s.addr, "D11 XY Road");
```

例**9.2** 依上例结构，输入两个学生的信息，输出成绩较高学生的信息

- 分析：

(1)定义两个结构相同的结构体变量**s1**和**s2**；

(2)分别输入两个学生的所有信息；

(3)比较两个学生的成绩，如果学生**s1**的成绩高于学生**s2**，就输出学生**s1**的全部信息，否则输出学生**s2**的全部信息。

```
struct Stu s1, s2, s;
scanf("%d %s %c %d %f", &s1.num,
   s1.name, &s1.sex, &s1.age, &s1.score);
scanf("%d %s %c %d %f", &s2.num,
   s2.name, &s2.sex, &s2.age, &s2.score);
s = s1.score>s2.score?s1:s2;
printf("%d %s %c %d %.2f\n", s.num,
     s.name, s.sex, s.age, s.score);
```

9.2 使用结构体数组

9.2.1定义结构体数组

9.2.2 结构体数组的应用举例

9.2.1定义结构体数组

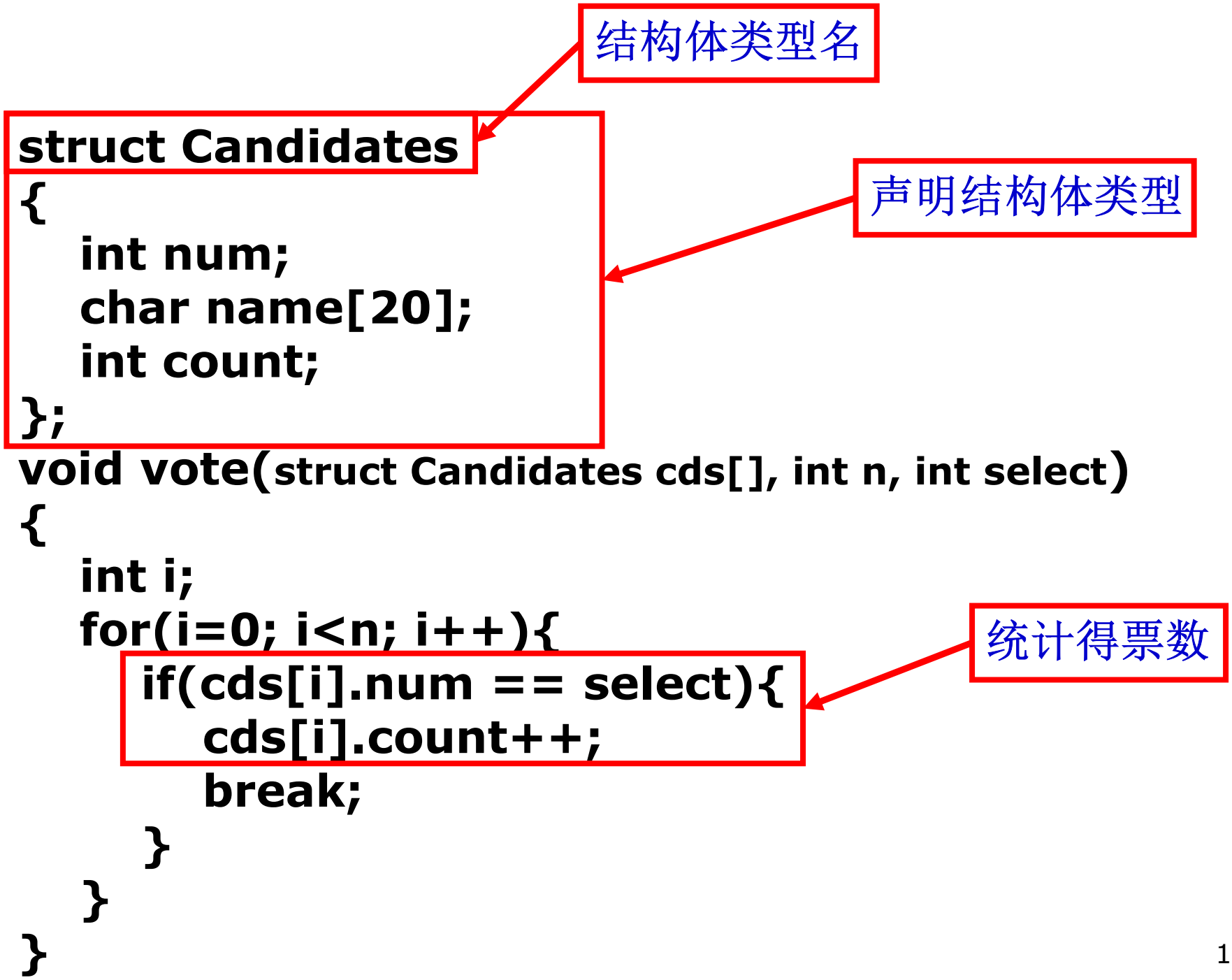
例**9.3** 新学年伊始，班委换届，现班中有**5**名同学竞选班长，每位同学**(**包含候选人**)**仅可给其中一位候选人**(**编号**1~5)**投票，要求编写一个统计选票的程序，先后输入候选人的编号，最后输出各人得票结果。

分析：

(1) 候选人信息包括编号、姓名、最终得票数，因此可以定义为结构体类型。**5**个候选人可以用该类型的数组来表示。

(2) 投票人输入编号后，对应编号的候选人票数增加**1**，所有投票人依次投票。

(3) 输出最终投票结果。


结构体类型名
声明结构体类型
统计得票数
struct Candidates
{
int num;
char name[20];
int count;
};
void vote(struct Candidates cds[], int n, int select)
{
int i;
for(i=0; i<n; i++){
if(cds[i].num == select){
cds[i].count++;
break;
}
}
}

```
#include <stdio.h>
int main()
{
  struct Candidates cds[5]={{1, "Zhao", 0},
  {2, "Qian", 0}, {3, "Sun", 0}, {4, "Li", 0}, {5,
  "Zhou", 0}};
  int i, select;
  for(i=0; i<30; i++){
      scanf("%d", &select);
      vote(cds, 5, select);
  }
  for(i=0; i<5; i++){
      printf("%4d%10s%2d\n", cds[i].num,
  cds[i].name, cds[i].count);
  }
  return 0;
}
```

定义结构体数组

依次投票并统计

输出投票结果

9.2.2 结构体数组的应用举例

例**9.4** 在上例的基础上，对投票结果按照从得票从高到低排序，并输出。

- 分析：在上例中仅需要增加一个对候选人得票排序的函数即可，排序方法可以使用冒泡排序、选择排序或插入排序，比较的是候选人之间的得票数，交换的是候选人的位置。

```
void sort(struct Candidates cds[], int n){
    int i, j;
    for(i=1; i<n; i++){
        struct Candidates key = cds[i];
        j = i-1;
        while(j>=0 && cds[j].count<key.count){
            cds[j+1] = cds[j]; j--;
        }
        cds[j+1] = key;
    }
}
```

待插入元素

插入元素

移动元素，找到插入位置

```
#include <stdio.h>
int main()
{
  struct Candidates cds[5]={{1, "Zhao", 0},
   {2, "Qian", 0}, {3, "Sun", 0}, {4, "Li", 0}, {5,
   "Zhou", 0}};
   int i, select;
   for(i=0; i<30; i++){
       scanf("%d", &select);
       vote(cds, 5, select);
   }
   sort(cds, 5);  //调用排序函数
   for(i=0; i<5; i++){
       printf("%4d%10s%2d\n", cds[i].num,
   cds[i].name, cds[i].count);
   }
   return 0;
}
```

定义结构体数组

依次投票并统计

输出投票结果

9.3 结构体指针

9.3.1 指向结构体变量的指针

9.3.2 指向结构体数组的指针

9.3.3 用结构体变量和结构体变量的指针作函数参数

9.3.1 指向结构体变量的指针

- 指向结构体对象的指针变量既可以指向结构体变量，也可以用来指向结构体数组中的元素。
- 指针变量的基类型必须与结构体变量的类型相同。

例如：

```
struct Student s;
struct Student *p = &s;
```

结构体变量通过运算符(.)访问成员

指向结构体变量的指针通过运算符(->)访问成员

如以下两语句等价：

```
s.score = 98.25
p->score = 98.25
```

9.3.2 指向结构体数组的指针

将9.4中输出候选人得票信息时使用指向结构体数组的指针来实现

```
#include <stdio.h>
int main()
{
    ……
   struct Candidates *p;
   for(p=cds; p<cds+5; p++){
        printf("%4d%10s%2d\n", p->num,
   p->name, p->count);
   }
   return 0;
}
```

9.3.3 用结构体变量和结构体变量的指针作函数参数

(1) 结构体变量成员做参数。

例如

int maxVotes = max(cds[1].count, cds[2].count);

(2) 结构体变量做参数(不建议)

例如 **void print(struct Candidates c);**

函数调用时，实参将值复制到形参**c**，将逐一复制对应成员的值，若结构体类型成员较多，则时间和空间开销较大。

(3) 指向结构体变量或数组的指针做参数

例如 **void print(struct Candidates *p);**

函数调用时，传递的是地址，和结构体成员多少没有关系，为避免函数中修改指针所指向对象的值，可使用**const**约束，如：**void print(const struct Candidates *p);**

对例**9.4**进行修改，所有功能均用函数来实现

```
void input(struct Candidates *p, int n);
void vote(struct Candidates cds[], int n, int select);
void output(struct Candidates *p, int n);
void print(const struct Candidates *p);
void sort(struct Candidates *p, int n);
struct Candidates getMaxCand(struct
   Candidates *p, int n);
```