第九章 用户自己建立数据类型

作者：石璞东

参考资料：《C程序设计（第四版）》谭浩强

前言

C语言允许用户根据需要自己建立数据类型，用它来定义变量。

9.1 定义和使用结构体变量

9.1.1 自己建立结构体类型

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。

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

声明一个结构体的一般形式为：

```
struct 结构体名{
    成员列表;
}
```

- 成员可以属于另一个结构体类型。

```
struct Date{
    int month;
    int year;
    int day;
};
struct Student{
    int num;
    char name[20];
    chat sex;
    int age;
    struct Date birthday;
    char addr[30];
};
```

9.1.2 定义结构体类型变量

- 先声明结构体类型，再定义该类型的变量

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

与`int a,b;`类似，定义完成后，`student1`和`student2`即为`struct Student`类型的变量。

- 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;
```

其一般形式为：

```
struct 结构体名{
  // 成员列表
}变量名列表;
```

- 不指定类型名而直接定义结构体类型变量

```
struct{
  //成员列表
}变量名列表;
```

9.1.3 结构体变量的初始化和引用

例9.1 把一个学生的学号、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{
        int num;
        char name[20];
        char sex;
        char addr[20];
    }student = {100001,"shipudong",'M',"China Xi'an"};
    printf("学号：%d\n姓名：%s\n性别：%c\n地址：
%s\n",student.num,student.name,student.sex,student.addr);
    return 0;
}
```

- `C99`标准允许对某一成员初始化，其他未被指定初始化的数值型成员被系统初始化为0，字符型成员被系统初始化为`\0`，指针型成员被系统初始化为`NULL`；

```
#include<stdio.h>
int main(){
    struct Student{
        int num;
        char name[20];
        char sex;
        char addr[20];
    }student = {100001,"shipudong",'M',"China Xi'an"};
    struct Student student1 = {.name = "hahaCoder"};
    printf("学号：%d\n姓名：%s\n性别：%c\n地址：
%s\n",student.num,student.name,student.sex,student.addr);
    printf("==========================\n");
    printf("学号：%d\n姓名：%s\n性别：%c\n地址：
%s\n",student1.num,student1.name,student1.sex,student1.addr);
    return 0;
```

```
}
```

- 可以通过`结构体变量名.成员名`引用结构体变量中成员的值，不能企图输出结构体变量名来达到输出结构体变量所有成员的值；
- 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。

```c
#include<stdio.h>
int main(){
    struct Date{
        int year;
        int month;
        int day;
    };
    struct Student{
        int num;
        char name[20];
        char sex;
        struct Date birthday;
        char addr[20];
    }student = {100001,"shipudong",'M',{2021,10,11},"China Xi'an"};
    struct Student student1 = {.name = "hahaCoder"};
    printf("学号：%d\n姓名：%s\n性别：%c\n出生日期：%d年%d月%d日\n地址：%s\n",student.num,student.name,student.sex,student.birthday.year,student.birthday.month,student.birthday.day,student.addr);
    printf("==========================\n");
    printf("学号：%d\n姓名：%s\n性别：%c\n出生日期：%d年%d月%d日\n地址：%s\n",student1.num,student1.name,student1.sex,student1.birthday.year,student1.birthday.month,student1.birthday.day,student1.addr);
    return 0;
}
```

- 对结构体变量的成员可以像普通变量一样进行各种运算
- 同类的结构体变量可以互相赋值；

```c
#include<stdio.h>
int main(){
    struct Date{
        int year;
        int month;
        int day;
    };
    struct Student{
        int num;
        char name[20];
        char sex;
        struct Date birthday;
        char addr[20];
```

```
    }student = {100001,"shipudong",'M',{2021,10,11},"China Xi'an"};
    struct Student student1 = {.name = "hahaCoder"};
    printf("学号：%d\n姓名：%s\n性别：%c\n出生日期：%d年%d月%d日\n地址：%s\n",student.num,student.name,student.sex,student.birthday.year,student.birthday.month,student.birthday.day,student.addr);
    printf("==========================\n");
    printf("学号：%d\n姓名：%s\n性别：%c\n出生日期：%d年%d月%d日\n地址：%s\n",student1.num,student1.name,student1.sex,student1.birthday.year,student1.birthday.month,student1.birthday.day,student1.addr);
    student1 = student;
    printf("==========================\n");
    printf("学号：%d\n姓名：%s\n性别：%c\n出生日期：%d年%d月%d日\n地址：%s\n",student.num,student.name,student.sex,student.birthday.year,student.birthday.month,student.birthday.day,student.addr);
    return 0;
}
```

- 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；

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{
        int num;
        char name[20];
        float score;
    }student1,student2;
    printf("请分别输入两个学生的学号、姓名和分数：\n");
    scanf("%d %s %f",&student1.num,student1.name,&student1.score);
    scanf("%d %s %f",&student2.num,student2.name,&student2.score);
    printf("成绩较高的学生为：\n");
    if(student1.score>student2.score){
        printf("%d %s %f\n",student1.num,student1.name,student1.score);
    }else if (student1.score<student2.score){
        printf("%d %s %f\n",student2.num,student2.name,student2.score);
    }else{
        printf("%d %s %.2f\n",student1.num,student1.name,student1.score);
        printf("%d %s %.2f\n",student2.num,student2.name,student2.score);
    }
    return 0;
}
```

9.2 使用结构体数组

9.2.1 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1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。

```c
#include<stdio.h>
#include<string.h>
struct Person{
    char name[20];
    int count;
}leader[3] = {"hahaCoder",0,"hahaAI",0,"hahaWebsite.",0};

int main(){
    char leader_name[20];
    for (int i = 1; i <= 8; i++) {
        printf("请输入第%d个候选人姓名：",i);
        scanf("%s",leader_name);
        for (int j = 0; j < 3; j++) {
            if(strcmp(leader_name, leader[j].name)==0){
                leader[j].count++;
            }
        }
    }

    for (int i = 0; i < 3; i++) {
        printf("姓名：%s --> 选票数：%d\n",leader[i].name,leader[i].count);
    }
    return 0;
}
```

定义结构体数组的一般形式为：

```
struct 结构体名{
  成员列表;
}数组名[数组长度];
```

9.2.2 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
#include<stdio.h>
struct Student{
    int num;
    char name[20];
    float score;
};
```

```
7
8  int main(){
9      struct Student student[5] = {
10         {0001,"hahaAI",98},
11         {0002,"hahaCoder",99},
12         {0003,"hahaWebsite",90},
13         {0004,"hahaOCR",89},
14         {0005,"hahaBook",100}
15     };
16     struct Student temp;
17     int i,j,k;
18     const int n = 5;
19     printf("得分由小到大的顺序为：\n");
20     for (i = 0; i < n-1; i++) {
21         k = i;
22         for (j = i+1; j < n; j++) {
23             if(student[j].score < student[k].score){
24                 k = j;
25             }
26         }
27         if(k!=i){
28             temp = student[k];
29             student[k] = student[i];
30             student[i] = temp;
31         }
32     }
33     for (int i = 0; i < n; i++) {
34         printf("学号：%d\t姓名：%s\t得分：
   %.2f\n",student[i].num,student[i].name,student[i].score);
35     }
36     return 0;
37 }
```

9.3 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。

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指向结构体变量，也可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。

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

int main(){
    struct Student{
        int num;
        char name[20];
        char sex;
        float score;
    }student,*p;
    p = &student;
    student.num = 100001;
    strcpy(student.name, "hahaCoder");
    student.sex = 'M';
    student.score = 98.29;
    printf("学号：%d\t姓名：%s\t性别：%c\t分数：%.2f\n",student.num,student.name,student.sex,student.score);
    printf("学号：%d\t姓名：%s\t性别：%c\t分数：%.2f\n",(*p).num,(*p).name,(*p).sex,(*p).score);
    printf("学号：%d\t姓名：%s\t性别：%c\t分数：%.2f\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种方法等价：

- `stu.成员名`：`stu.num`；
- `(*p).成员名`：`(*p).num`；
- `p->成员名`：`p->num`；

9.3.2 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。

```c
#include<stdio.h>
struct Student{
    int num;
    char name[20];
    float score;
};
struct Student student[5] = {
    {0001,"hahaAI",98},
    {0002,"hahaCoder",99},
    {0003,"hahaWebsite",90},
    {0004,"hahaOCR",89},
    {0005,"hahaBook",100}
};
```

```
14
15  int main(){
16      struct Student *p;
17      for (p = student; p < student+5; p++) {
18          printf("学号：%d\t姓名：%s\t分数：%.2f\n",p->num,p->name,p->score);
19      }
20      return 0;
21  }
```

9.3.3 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种方法：

- 用结构体变量的成员作参数；
- 用结构体变量作实参；
- 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；

例9.7 有`n`个结构体变量，内含学生学号、姓名和3门课的成绩，要求求出平均成绩最高的学生的信息。

```
1   #include<stdio.h>
2   #define NUM 3
3   struct Student{
4       int num;
5       char name[20];
6       float score[3];
7       float aver;
8   };
9
10  int main(){
11      void input(struct Student student[]);
12      struct Student max(struct Student student[]);
13      void print(struct Student student);
14      struct Student student[NUM],*p = student;
15      input(p);
16      print(max(p));
17      return 0;
18  }
19  void input(struct Student student[]){
20      for (int i = 0; i < NUM; i++) {
21          printf("请输入第%d个学生的学号、姓名、三门课成绩等信息：\n",i+1);
22          scanf("%d %s %f %f
    %f",&student[i].num,student[i].name,&student[i].score[0],&student[i].score
    [1],&student[i].score[2]);
23          student[i].aver =
    (student[i].score[0]+student[i].score[1]+student[i].score[2])/3.0;
24      }
25  }
```

```
struct Student max(struct Student student[]){
    int max_num = 0;
    for (int i = 0; i < NUM; i++) {
        if(student[i].aver > student[max_num].aver){
            max_num = i;
        }
    }
    return student[max_num];
}
void print(struct Student student){
    printf("成绩最高的学生为：\n");
    printf("学号：%d\t姓名：%s\t三门课成绩分别为：%.2f、%.2f、%.2f\t平均成绩：%.2f\n",student.num,student.name,student.score[0],student.score[1],student.score[2],student.aver);

}
```

9.4 用指针处理链表

9.4.1 什么是链表

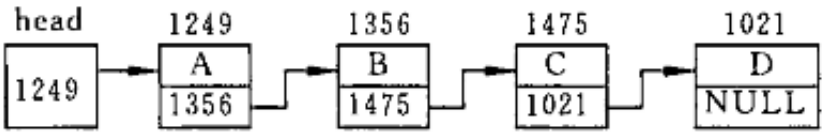

图 9.8

链表有一个**头指针**变量，图中以`head`表示，它存放一个地址，该地址指向一个元素，链表中每一个元素称为结点，每个结点都应该包括两个部分：

- 用户需要用的实际数据；
- 下一个结点的地址；

可以看出，`head`指向第1个元素，第1个元素又指向第2个元素……直到最后一个元素，该元素不再指向其他元素，它称为表尾，它的地址部分放一个`NULL`，表示空地址，链表到此结束。链表中各元素在内存中的地址可以是不连续的，要找某一元素，必须先找到上一个元素，根据它提供的下一元素地址才能找到下一个元素，如果不提供头指针，则整个链表无法访问。

9.4.2 建立简单的静态链表

参考试听课代码。

9.4.3 建立动态链表

参考试听课代码。

9.4.4 输出链表

参考试听课代码。

9.5 共用体类型

9.5.1 什么是共用体类型

定义共用体类型的一般形式为：

```
union 共用体名{
   成员列表
}变量列表;
```

共用体与结构体定义形式类似，但它们的含义是不同的，结构体变量所占内存长度是各成员占的内存长度之和，每个成员分别占有其自己的内存单元，而共用体变量所占的内存长度等于最长的成员的长度。

```
#include<stdio.h>
struct Student{
    int num;
    char name[20];
    float score[3];
    float aver;
};
union Person{
    int num;
    char name[20];
    float score[3];
    float aver;
};

int main(){
    printf("结构体Student的大小为：%d\n",sizeof(struct Student)); // 40
    printf("共用体union的大小为：%d\n",sizeof(union Person)); // 20
    return 0;
}
```

9.5.2 引用共用体变量的方式

```c
#include<stdio.h>
union Person{
    int num;
    char name[20];
    float aver;
} person;

int main(){
    person.num = 1000;
    printf("学号：%d\t姓名：%s\t平均分：%.2f\n",person.num,person.name,person.aver);
    return 0;
}
```

9.5.3 共用体类型数据的特点

- 同一个内存段可以用来存放几种不同类型的成员，但在每一瞬时只能存放其中一个成员，而不是同时存放几个；
- 可以对共用体变量初始化，但初始化列表中只能有一个常量；
- 共用体变量中起作用的成员是最后一次被赋值的成员，在对共用体变量中的一个成员赋值后，原有变量存储单元中的值就被取代；

共用体类型的数据的应用场景：有时需要对同一段空间安排不同的用途。

例9.8 有若干个人员的数据，其中有学生和教师，学生的数据中包括：姓名、号码、性别、职业、班级；教师的数据包括：姓名、号码、性别、职业、职务。

```c
#include<stdio.h>
struct{
    int num;
    char name[10];
    char sex;
    char job;
    union{
        int clas;
        char position[10];
    }category;
}person[2];

int main(){
    int i;
    for (i = 0; i < 2; i++) {
        printf("请输入第%d个用户的信息：\n",(i+1));
```

```
        scanf("%d %s %c %c",&person[i].num,person[i].name,&person[i].sex,&person[i].job);
        if(person[i].job=='s'){
            printf("请输入学生班级信息：\n");
            scanf("%d",&person[i].category.clas);
        }else if (person[i].job == 't'){
            printf("请输入老师职务信息：\n");
            scanf("%s",person[i].category.position);
        }else{
            printf("输入错误！\n");
        }
    }
    for (i = 0;i < 2; i++) {
        if(person[i].job == 's'){
            printf("学号：%d\t姓名：%s\t性别：%c\t职业：%c\t班级：%d\n",person[i].num,person[i].name,person[i].sex,person[i].job,person[i].category.clas);
        }else{
            printf("号码：%d\t姓名：%s\t性别：%c\t职业：%c\t职务：%s\n",person[i].num,person[i].name,person[i].sex,person[i].job,person[i].category.position);
        }
    }
    return 0;
}
```

9.6 使用枚举类型

所谓**枚举**就是把可能的值一一列举出来，变量的值只限于列举出来的值的范围内，声明枚举类型用`enum`开头，其一般形式为：`enum [枚举名]{枚举元素列表};`。

- `C`编译对枚举类型的枚举元素按常量处理，故不能对它们赋值；
- 每一个枚举元素都代表一个整数，`C`语言编译按定义时的顺序默认它们的值为0、1、2、3、4...；
- 枚举元素可以用来做比较；

例9.9 枚举类型案例

```
#include <stdio.h>

enum DAY
{
      MON, TUE, WED, THU, FRI, SAT, SUN
};

int main()
{
    enum DAY day;
```

```
    day = WED;
    printf("%d\n",day);
    return 0;
}
```

9.7 用`typedef`声明新类型

- 简单地用一个新的类型名代替原有的类型名

```
#include <stdio.h>

int main(){
//  int a = 10;
    typedef int shipudongInt;
    shipudongInt a = 10;
    printf("%d\n",a);
    return 0;
}
```

- 命名一个简单的类型名代替复杂的类型表示方法
  - 命名一个新的类型名代表结构体类型

```
typedef struct{
  int month;
  int day;
  int year;
}Date;
Date birthday;
```

  - 命名一个新的类型名代表数组类型

```
typedef int NUM[100];
NUM a;
```

  - 命名一个新的类型名代表指针类型

```
typedef char* string;
string p;
```

- 命名一个新的类型名代表指向函数的指针类型

```
typedef int (*pointer)();//pointer为指向函数的指针类型，该函数返回整型值；
pointer p1,p2;//p1、p2为pointer类型的指针变量
```