计算机导论与程序设计 [CS006001-60]

段江涛

机电工程学院

2019 年 12 月

lecture-17 主要内容

建立自己的数据类型

1. 定义结构体数据类型: struct 结构体类型名 {...};
2. 定义结构体类型变量: struct 结构体类型名 变量名;
3. 引用结构体变量的数据成员: 结构体变量名. 成员名
4. 结构体指针变量和结构体数组

训练编程逻辑思维方式:

- 领会结构化 (模块化) 编程思想。
- 用结构体管理相互关联的数据, 使其成为结构化的整体数据, 便于统一处理。
- 大问题分解为小问题, 设计函数解决小问题, 各个子函数彼此之间相互独立 (便于调试, 不易出错), 又可通过参数和返回值传递数据。
- 主程序调用各个子函数, 解决 “大” 问题。

定义结构体数据类型: struct 结构体类型名 {...};

定义结构体数据类型

```
struct 结构体类型名
{
    数据类型 成员名1;
    数据类型 成员名2;
    ... // 定义其它成员
};
```

例: 定义结构体数据类型, 描述如下信息

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

```
struct Student // 定义结构体数据类型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
```

定义结构体数据类型: struct 结构体类型名 {...};

定义结构体数据类型

```
struct 结构体类型名
{
    数据类型 成员名1;
    数据类型 成员名2;
    ... // 定义其它成员
};
```

例: 定义结构体数据类型, 描述如下信息

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

```
struct Student // 定义结构体数据类型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
```

嵌套定义结构体: 结构体的成员, 也可以是另外一个结构体变量。

```
struct Date // 定义结构体类型 struct Date
{
    int month; // 月
    int day;   // 日
    int year;  // 年
};
struct Student // 定义结构体类型 struct Student
{
    int num;
    char name[20];
    char sex;
    int age;
    struct Date birthday; //成员birthday是struct Date类型变量
    char addr[30];
};
```

num	name	sex	age	birthday			addr
10010	Li Fang	M	18	month	day	year	Xian

定义结构体类型变量: struct 结构体类型名 变量名;

定义结构体类型后, 可以当作内置类型 (如, int) 一样进行变量定义和使用。

```
#include<stdio.h>
....
// 定义结构体数据类型struct Student
struct Student
{
    int num;
    char name[20];
    char sex;
    int age;
    float score;
    char addr[30];
};
```

```
void fun1()
{
    // 定义结构体类型变量
    struct Student stu1, stu2;
    ....
}

// 结构体类型变量可作为函数的形式参数
void fun2(struct Student stu)
{
    ....
}
```

定义结构体类型变量: struct 结构体类型名 变量名;

定义结构体类型后, 可以当作内置类型 (如, int) 一样进行变量定义和使用。

```
#include<stdio.h>
....
// 定义结构体数据类型struct Student
struct Student
{
    int num;
    char name[20];
    char sex;
    int age;
    float score;
    char addr[30];
};
```

```
void fun1()
{
    // 定义结构体类型变量
    struct Student stu1, stu2;
    ....
}

// 结构体类型变量可作为函数的形式参数
void fun2(struct Student stu)
{
    ....
}
```

结构体类型变量的初始化

```
#include<stdio.h>
....
// 定义结构体数据类型struct Student
struct Student
{
    int num;
    char name[20];
    char sex;
    int age;
    float score;
    char addr[30];
};
```

```
void fun()
{
    // 定义结构体变量时, 初始化各成员变量
    struct Student stu1={10101,"
     LiLin",'M',18,90,"Xidian"},stu2
     ;
    ....
}
```

引用结构体变量的数据成员: 结构体变量名. 成员名

定义结构体类型和结构体变量后, 通过“.” 成员运算符, 引用其数据成员。

```
#include<stdio.h>
....
// 定义结构体数据类型struct Student
struct Student
{
  int num;
  char name[20];
  char sex;
  int age;
  float score;
  char addr[30];
};
```

```
void fun()
{
  // 定义结构体变量时, 初始化各成员变量
  struct Student stu1={10101,"LiLin
    ",'M',18,90,"Xidian"},stu2;
  printf("%d,%s,%c,%d,%.2f,%s",stu1
    .num,stu1.name,stu1.sex,stu1.
    age,stu1.score,stu1.addr);
  stu2.num=10102;
  // 注: 字符串用数组表示, 不能直接赋值。
  strcpy(stu2.name,"Wanghong");
}
```

对结构体变量的成员可以像普通变量一样进行各种运算

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

```
struct Student stu1,stu2;
stu1.num=10010;
stu1.sex='F';
stu1.birthday.month=6;// 逐级引用
stu1.age=18;
stu2.age=stu1.age+1;
stu2.age++;
stu2=stu1; // stu1中的各数据成员赋值给stu2
scanf("%d",&stu2.num);
gets(stu2.name); // 接受键盘输入的字符串
stu2.sex=stu1.sex; // 同性别
```

“.” 成员运算符在所有运算符中优先级最高，

stu2.num 可以当作一个整体看待。

结构体指针变量

```
// 定义结构体
struct Student
{
  int num;
  char name[20];
  char sex;
  int age;
  char addr[30];
};
```

```
// stu2是结构体指针变量
struct Student stu1,*stu2;
stu1.num=10010;
stu1.sex='F';
// 引用结构体指针变量指向的数据成员
(*stu2).num=10011; // 一定要有括号
stu2->num=10011; // 或
scanf("%d",&(*stu2).num));
scanf("%d",&(stu2->num)); // 或
gets((*stu2).name);
gets(stu2->name); // 或
(*stu2).sex=stu1.sex;
stu2->sex=stu1.sex; // 或
```

结构体数组

```
// 定义结构体
struct Student
{
  int num;
  char name[20];
  char sex;
  int age;
  char addr[30];
};
```

```
// 结构体数组
struct Student stu[100];
scanf("%d",&stu[0].num);
gets(stu[0].name);
scanf("%c",&stu[0].sex);
stu[0].sex=getchar(); // 或
strcpy(stu[0].addr,"Xian");
```

例 1: 输入学生信息并输出统计数据

```
struct Student
{
    int num;
    char name[20];
    // 5门课成绩
    float score[5];
};
```

```
int main()
{
  struct Student stu;
  int i; float sum=0;
  scanf("%d%s",&stu.num,stu.name);
  for(i=0;i<5;i++)
     scanf("%f",&stu.score[i]);
  // 计算与输出
  printf("%d␣%s",stu.num,stu.name);
  for(i=0;i<5;i++)
  {
     printf("␣%.2f␣",stu.score[i]);
     sum=sum+stu.score[i];
  }
  printf("%.2f\n",sum/5.0);//平均成绩
  return 0;
}
```

例 2: 输入学生信息并输出统计数据 (函数实现)

```
struct Student
{
  int num;
  char name[20];
  // 5门课成绩
  float score[5];
};
// 地址传递
void input(struct Student *stu)
{
  int i;
  scanf("%d%s",&(stu->num),stu->name);
  for(i=0;i<5;i++)
    scanf("%f",&(stu->score[i]));
    //scanf("%f",&((*stu).score[i]));
}
```

```
void print(struct Student stu)
{
  int i; float sum=0;
  printf("%d␣%s",stu.num,stu.name);
  for(i=0;i<5;i++)
  {
     printf("␣%.2f␣",stu.score[i]);
     sum=sum+stu.score[i];
  }
  printf("%.2f\n",sum/5.0);
}
int main()
{
  struct Student stu;
  input(&stu); print(stu);
  return 0;
}
```

例 3: 输入学生信息并输出统计数据 (结构体数组), 输入

```
// 定义结构体
struct Student
{
  int num;
  char name[20];
  // 5门课成绩
  float score[5];
};
```

```
// 输入n个学生信息
//void input(struct Student stu[],int n)
void input(struct Student *stu, int n)
{
  int i,j;
  for(i=0;i<n;i++)
  {
     scanf("%d%s",&stu[i].num,stu[i].name);
     for(j=0;j<5;j++)
       scanf("%f",&stu[i].score[j]);
  }
}
```

例 3: 输入学生信息并输出统计数据 (结构体数组), 输出

```
//void print(struct Student stu[],int n)
void print(struct Student *stu,int n)
{
  int i,j; float sum;
  for(i=0;i<n;i++)
  {
    printf("%d␣%s",stu[i].num,stu[i].name);
    sum=0;//注意在此处置0
    for(j=0;j<5;j++)
    {
      printf("␣%.0f␣",stu[i].score[j]);
      sum=sum+stu[i].score[j];
    }
    printf("%.2f\n",sum/5.0);
  }
}
```

```
int main()
{
  struct Student stu[100];
  input(stu,2); // 输入2个学生
  print(stu,2); // 输出2个学生
  return 0;
}
```

例如, 输入两个学生的数据

1 zhang3 10 20 30 40 50

2 wang5 100 90 80 70 60

1 zhang3 10 20 30 40 50 30.00

2 wang5 100 90 80 70 60 80.00

例 4: 输入学生信息并输出统计数据 (结构体数组), 输入

```
// 含平均成绩的结构体
struct Student
{
  int num;
  char name[20];
  // 5门课成绩
  float score[5];
  // 平均成绩
  float aver;
};
```

```
// 输入并计算平均成绩, 返回平均成绩最高者(假定唯一)
struct Student input(struct Student *stu, int n)
{
  int i,j,maxIndex=0; float sum;
  for(i=0;i<n;i++)
  {
    sum=0;
    scanf("%d%s",&stu[i].num,stu[i].name);
    for(j=0;j<5;j++)
    {
      scanf("%f",&stu[i].score[j]);
      sum=sum+stu[i].score[j];
    }
    stu[i].aver=sum/5.0; //填充平均成绩
    //非唯一最大者, 这里添加条件。复杂条件可以设计独立函数
    if(stu[i].aver > stu[maxIndex].aver) maxIndex=i;
  }
  return stu[maxIndex];
}
```

例 4: 输入学生信息并输出统计数据 (结构体数组), 输出

```
void print(struct Student *stu,int n)
{
  int i,j;
  for(i=0;i<n;i++)
  {
    printf("%d␣%s␣",stu[i].num,
                    stu[i].name);
    for(j=0;j<5;j++)
      printf("%.2f␣",stu[i].score[j]);
    printf("%.2f\n",stu[i].aver);
  }
}
```

```
int main()
{
  struct Student stu[100],maxStu;
  maxStu=input(stu,3); // 输入3个学生
  print(stu,3); // 输出3个学生
  // 可复用print函数
  print(&maxStu,1); // 平均成绩最大者
  return 0;
}
```

例 5: 输入学生信息并输出统计数据 (结构体数组), 交换函数

```
// 交换两个结构体指针的内容, 地址传递
// 两个结构体的各数据成员互相交换
void swap(struct Student *stu1, struct Student *stu2)
{
    struct Student tmp;
    tmp = *stu1; *stu1 = *stu2; *stu2 = tmp;
}
```

例 5: 输入学生信息并输出统计数据 (结构体数组), 排序

```
// 定义排序函数(选择法，降序)：按平均值排序，如果平均值相同，按照前2门课比较单科成绩
void sort(struct Student stu[],int n)
{
  int i,j,k;
  for(i = 0; i < n-1; i++)
  {
    k = i; // 未经排序较大者
    for(j = i + 1; j < n; j++)
    {
       if(stu[j].aver > stu[k].aver) k=j;
       else if(stu[j].aver == stu[k].aver)
       {
         if(stu[j].score[0] > stu[k].score[0] ||
            stu[j].score[1] > stu[k].score[1])
            k = j;
       }
    }
    if(k != i) swap(&stu[i],&stu[k]); // 交换
  }
}
```

lecture-17 主要内容

建立自己的数据类型

1. 定义结构体数据类型: struct 结构体类型名 {...};
2. 定义结构体类型变量: struct 结构体类型名 变量名;
3. 引用结构体变量的数据成员: 结构体变量名. 成员名
4. 结构体指针变量和结构体数组

训练编程逻辑思维方式:

- 领会结构化 (模块化) 编程思想。
- 用结构体管理相互关联的数据, 使其成为结构化的整体数据, 便于统一处理。
- 大问题分解为小问题, 设计函数解决小问题, 各个子函数彼此之间相互独立 (便于调试, 不易出错), 又可通过参数和返回值传递数据。
- 主程序调用各个子函数, 解决 “大” 问题。

欢迎批评指正！