程序设计基础

第九章: 结构

刘新国

浙江大学计算机学院
浙江大学 CAD&CG 国家重点实验室

December 8, 2021

内容提要

结构类型的定义

- 结构的定义
- 结构变量的定义和初始化
- 结构类型的函数返回值和参数
- 结构和数组
- 结构和指针

内容提要

结构类型的定义
- 结构的定义
- 结构变量的定义和初始化
- 结构类型的函数返回值和参数
- 结构和数组
- 结构和指针

什么是结构

结构：一种自定义的、复合数据类型

什么是结构

结构：一种自定义的、复合数据类型

- 一个结构体包含多个结构成员

什么是结构

结构：一种自定义的、复合数据类型

- 一个结构体包含多个结构成员
- 每一个结构成员可以单独使用

什么是结构

结构：一种自定义的、复合数据类型

- 一个结构体包含多个结构成员
- 每一个结构成员可以单独使用
- 用于表示复杂的对象或数据类型

什么是结构

结构：一种自定义的、复合数据类型

- 一个结构体包含多个结构成员
- 每一个结构成员可以单独使用
- 用于表示复杂的对象或数据类型

例如：如何表示学生的信息

什么是结构

结构：一种自定义的、复合数据类型

- 一个结构体包含多个结构成员
- 每一个结构成员可以单独使用
- 用于表示复杂的对象或数据类型

例如：如何表示学生的信息

- 学号

什么是结构

结构：一种自定义的、复合数据类型

- 一个结构体包含多个结构成员
- 每一个结构成员可以单独使用
- 用于表示复杂的对象或数据类型

例如：如何表示学生的信息

- 学号
- 姓名

什么是结构

结构：一种自定义的、复合数据类型

- 一个结构体包含多个结构成员
- 每一个结构成员可以单独使用
- 用于表示复杂的对象或数据类型

例如：如何表示学生的信息

- 学号
- 姓名
- 计算机成绩
- 英语成绩
- 数学成绩
- 平均成绩

什么是结构

结构：一种自定义的、复合数据类型

- 一个结构体包含多个结构成员
- 每一个结构成员可以单独使用
- 用于表示复杂的对象或数据类型

例如：如何表示学生的信息

- 学号
- 姓名
- 计算机成绩
- 英语成绩
- 数学成绩
- 平均成绩

什么是结构

结构：一种自定义的、复合数据类型

- 一个结构体包含多个结构成员
- 每一个结构成员可以单独使用
- 用于表示复杂的对象或数据类型

例如：如何表示学生的信息

- 学号
- 姓名
- 计算机成绩
- 英语成绩
- 数学成绩
- 平均成绩

```
struct student
{
    int   num;        // 学号
    char  name[20];   // 姓名
    int   computer;   // 计算机
    int   english;    // 英语
    int   math;       // 数学
    float average;    // 平均
};
```

内容提要

结构类型的定义
- 结构的定义
- 结构变量的定义和初始化
- 结构类型的函数返回值和参数
- 结构和数组
- 结构和指针

结构的定义

```
struct 结构类型名称
{
    结构成员表
};
```

- 使用 c 语言关键字 struct 定义结构

结构的定义

```
struct  结构类型名称
{
    结构成员表
};
```

- 使用 c 语言关键字 struct 定义结构
- **结构类型名称** 符合 c 语言标识符语法

结构的定义

```
struct 结构类型名称
{
    结构成员表
};
```

- 使用 c 语言关键字 struct 定义结构
- 结构类型名称 符合 c 语言标识符语法
- 结构成员表用一对大括号扩起来，末尾以分号结束

结构的定义

```
struct 结构类型名称
{
    结构成员表
};
```

- 使用 c 语言关键字 struct 定义结构
- **结构类型名称** 符合 c 语言标识符语法
- **结构成员表**用**一对大括号**扩起来，末尾以**分号**结束

结构成员表的格式

```
成员1类型名  成员1变量名;
成员2类型名  成员2变量名;
成员3类型名  成员3变量名;
.......      .......
```

结构的定义

结构成员表中，相同类型的成员可以共用类型名

```
struct student
{
    int    num;
    char   name[20];
    int    computer;
    int    english;
    int    math;
    float  average;
};
```

结构的定义

结构成员表中，相同类型的成员可以共用类型名

```
struct student
{
    int    num;
    char   name[20];
    int    computer;
    int    english;
    int    math;
    float  average;
};
```

```
struct student
{
    int    num;
    char   name[20];
    int    computer, english, math;
    float  average;
};
```

结构的定义

结构成员表中，相同类型的成员可以共用类型名

和普通的变量定义一样

- 可以独立分开
- 也可以合在一起

```
struct student
{
    int    num;
    char   name[20];
    int    computer, english, math;
    float  average;
};
```

```
struct student
{
    int    num;
    char   name[20];
    int    computer;
    int    english;
    int    math;
    float  average;
};
```

结构的定义 – 若干例子

```
// 平面坐标点
struct point
{
    double x;
    double y;
};
```

结构的定义 – 若干例子

```
// 平面坐标点
struct point
{
    double x;
    double y;
};
```

或者

```
struct point
{
    double x, y;
};
```

结构的定义 – 若干例子

```
// 平面坐标点
struct point
{
    double x;
    double y;
};
```

或者

```
struct point
{
    double x, y;
};
```

```
// 图像信息数据
struct image
{
    int width, height;
    int format;  //格式
    void *pixels;//像素
};
```

结构的定义 – 若干例子

```
// 平面坐标点
struct point
{
    double x;
    double y;
};
```

或者

```
struct point
{
    double x, y;
};
```

```
// 图像信息数据
struct image
{
    int width, height;
    int format;  //格式
    void *pixels;//像素
};
```

```
// 一个产品信息数据
struct image
{
    int id;    //编号
    int type; //类型
    char name[20];//名称
    float price;  //价格
};
```

结构的定义 – 若干例子

```
// 复数
struct complex
{
    double real;
    double imag;
};
```

结构的定义 – 若干例子

```
// 复数
struct complex
{
    double real;
    double imag;
};

// 地址
struct address
{
    char city[20];
    char street[40];
    int  zip;
};
```

结构的定义 – 可以嵌套

结构可以嵌套 – 一个结构可以包含另一个结构

```
struct friend
{
    char name[20];
    char phone[13];
    int  age;
    struct address addr;
    char memo[20];
};
```

- 结构成员 addr 也是一个结构。

结构的定义 – 可以嵌套

结构可以嵌套 – 一个结构可以包含另一个结构

```
struct friend
{
    char name[20];
    char phone[13];
    int  age;
    struct address addr;
    char memo[20];
};
```

- 结构成员 addr 也是一个结构。
- 成员结构必须先定义，否则会编译错误

结构的定义 – 可以嵌套

结构可以嵌套 – 一个结构可以包含另一个结构

```
struct friend
{
    char name[20];
    char phone[13];
    int  age;
    struct address addr;
    char memo[20];
};
```

- 结构成员 addr 也是一个结构。
- 成员结构必须先定义，否则会编译错误
- 结构不能自己嵌套自己

内容提要

结构类型的定义
- 结构的定义
- 结构变量的定义和初始化
- 结构类型的函数返回值和参数
- 结构和数组
- 结构和指针

结构变量定义

1. 常规定义

```
struct friend f1, f2, jack;
```

结构变量定义

1. 常规定义

```
struct friend f1, f2, jack;
```

- 定义了三个变量：f1, f2, jack

结构变量定义

1. 常规定义

```
struct friend f1, f2, jack;
```

- 定义了三个变量：f1, f2, jack
- 他们具有相同的结构：struct friend

结构变量定义

1. 常规定义

```
struct friend f1, f2, jack;
```

- 定义了三个变量：f1, f2, jack
- 他们具有相同的结构：struct friend

```
struct friend
{
    char name[20];
    char phone[13];
    int  age;
    struct address addr;
    char memo[20];
};
```

结构变量定义

1. 常规定义

```
struct friend f1, f2, jack;
```

- 定义了三个变量：f1, f2, jack
- 他们具有相同的结构：struct friend

```
struct friend
{
    char name[20];
    char phone[13];
    int  age;
    struct address addr;
    char memo[20];
};
```

- 注意：要使用关键词 struct

结构变量定义

结构变量的存储

C 语言规定，结构类型变量的存储布局按照其类型定义中成员的先后顺序排列

结构变量定义

结构变量的存储

C 语言规定，结构类型变量的存储布局按照其类型定义中成员的先后顺序排列

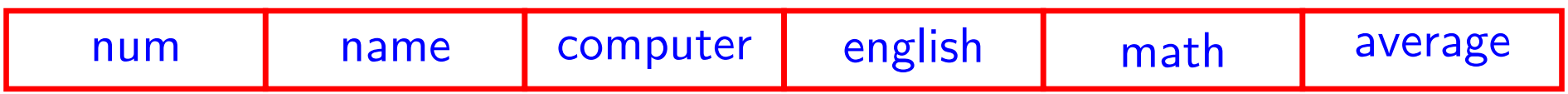
```
struct student
{
    int   num;
    char  name[20];
    int   computer;
    int   english;
    int   math;
    float average;
};
```

结构变量定义

结构变量的存储

C 语言规定，结构类型变量的存储布局按照其类型定义中成员的先后顺序排列

```
struct student
{
    int    num;
    char   name[20];
    int    computer;
    int    english;
    int    math;
    float  average;
};
```



结构变量定义

2. 带初始化的定义

```
struct friend f = { "大头", "13912345678", 20,
    {"杭州", "紫金港三墩人民公园", 310058},
    "学霸" };
```

结构变量定义

2. 带初始化的定义

```
struct friend f = { "大头", "13912345678", 20,
     {"杭州", "紫金港三墩人民公园", 310058},
     "学霸" };

struct friend
{
     char name[20];
     char phone[13];
     int  age;
     struct address addr;
     char memo[20];
};

// 地址
struct address
{
     char city[20];
     char street[40];
     int  zip;
};
```

结构变量定义

2. 带初始化的定义

```
struct friend f = { "大头", "13912345678", 20,
     {"杭州", "紫金港三墩人民公园", 310058},
     "学霸" };

struct friend
{
     char name[20];
     char phone[13];
     int  age;
     struct address addr;
     char memo[20];
};

// 地址
struct address
{
     char city[20];
     char street[40];
     int  zip;
};
```

结构变量定义

2. 带初始化的定义

```
struct friend f = { "大头", "13912345678", 20,
     {"杭州", "紫金港三墩人民公园", 310058},
     "学霸" };

struct friend
{
     char name[20];
     char phone[13];
     int  age;
     struct address addr;
     char memo[20];
};

// 地址
struct address
{
     char city[20];
     char street[40];
     int  zip;
};
```

结构变量定义

2. 带初始化的定义

```
struct friend f = { "大头", "13912345678", 20,
     {"杭州", "紫金港三墩人民公园", 310058},
     "学霸" };

struct friend
{
     char name[20];
     char phone[13];
     int  age;
     struct address addr;
     char memo[20];
};

// 地址
struct address
{
     char city[20];
     char street[40];
     int  zip;
};
```

结构变量定义

2. 带初始化的定义

```
struct friend f = { "大头", "13912345678", 20,
      {"杭州", "紫金港三墩人民公园", 310058},
      "学霸" };

struct friend
{
      char name[20];
      char phone[13];
      int  age;
      struct address addr;
      char memo[20];
};

// 地址
struct address
{
      char city[20];
      char street[40];
      int  zip;
};
```

结构变量定义

2. 带初始化的定义

```
struct friend f = { "大头", "13912345678", 20,
      {"杭州", "紫金港三墩人民公园", 310058},
      "学霸" };
```

```
struct friend
{
      char name[20];
      char phone[13];
      int  age;
      struct address addr;
      char memo[20];
};
```

```
// 地址
struct address
{
      char city[20];
      char street[40];
      int  zip;
};
```

结构变量定义

2. 带初始化的定义

```
struct friend f = { "大头", "13912345678", 20,
      {"杭州", "紫金港三墩人民公园", 310058},
      "学霸" };

struct friend
{
      char name[20];
      char phone[13];
      int  age;
      struct address addr;
      char memo[20];
};

// 地址
struct address
{
      char city[20];
      char street[40];
      int  zip;
};
```

结构变量定义

2. 带初始化的定义

结构变量定义

2. 带初始化的定义

- 按照定义的顺序进行对应初始化;

结构变量定义

2. 带初始化的定义

- 按照定义的顺序进行对应初始化;
- 可以部分初始化;

结构变量定义

2. 带初始化的定义

- 按照定义的顺序进行对应初始化;
- 可以部分初始化;
  - 没有匹配初值的成员，值为 0

结构变量定义

2. 带初始化的定义

- 按照定义的顺序进行对应初始化;
- 可以部分初始化;
  - 没有匹配初值的成员，值为 0

没有初始化的结构变量

结构变量定义

2. 带初始化的定义

- 按照定义的顺序进行对应初始化;
- 可以部分初始化;
  - 没有匹配初值的成员，值为 0

没有初始化的结构变量

- 如果是局部变量，他的成员值为垃圾值

结构变量定义

2. 带初始化的定义

- 按照定义的顺序进行对应初始化;
- 可以部分初始化;
  - 没有匹配初值的成员，值为 0

没有初始化的结构变量

- 如果是局部变量，他的成员值为垃圾值
- 否则（全局变量或静态变量，他的成员值为全零

结构变量定义

3. 混合定义 – 同时定义结构和变量

```
struct student
{
    int   num;                              /*学号*/
    char  name[10];                         /*姓名*/
    int   computer, english, math;          /*成绩*/
    double average;                         /*平均成绩*/
} s1, s2;
```

结构变量定义

3. 混合定义 – 同时定义结构和变量

```
struct student
{
    int   num;                          /*学号*/
    char  name[10];                     /*姓名*/
    int   computer, english, math;      /*成绩*/
    double average;                     /*平均成绩*/
} s1, s2;
```

- 定义了结构类型：struct student

结构变量定义

3. 混合定义 – 同时定义结构和变量

```
struct student
{
    int   num;                              /*学号*/
    char  name[10];                         /*姓名*/
    int   computer, english, math;          /*成绩*/
    double average;                         /*平均成绩*/
} s1, s2;
```

- 定义了结构类型：struct student
- 定义了结构变量：s1, s2

结构变量定义

4. 无名定义 – 不给类型取名称

```
struct
{
    int   num;                              /*学 号*/
    char  name[10];                         /*姓 名*/
    int   computer, english, math;          /*成 绩*/
    double average;                         /*平 均 成 绩*/
} s1, s2;
```

结构变量定义

4. 无名定义 – 不给类型取名称

```
struct
{
    int   num;                          /*学 号*/
    char  name[10];                     /*姓 名*/
    int   computer, english, math;      /*成 绩*/
    double average;                     /*平 均 成 绩*/
} s1, s2;
```

- 定义了结构类型，但是没有名称

结构变量定义

4. 无名定义 – 不给类型取名称

```
struct
{
    int   num;                          /*学 号*/
    char  name[10];                     /*姓 名*/
    int   computer, english, math;      /*成 绩*/
    double average;                     /*平 均 成 绩*/
} s1, s2;
```

- 定义了结构类型，但是没有名称
- 定义了结构变量：s1, s2

结构变量定义

4. 无名定义 – 不给类型取名称

```
struct
{
    int   num;                          /*学 号*/
    char  name[10];                     /*姓 名*/
    int   computer, english, math;      /*成 绩*/
    double average;                     /*平 均 成 绩*/
} s1, s2;
```

- 定义了结构类型，但是没有名称
- 定义了结构变量：s1, s2
- 因为没有名称，这种结构类型无法再次使用

结构变量定义

4. 无名定义 – 不给类型取名称

```
struct
{
    int   num;                          /*学 号*/
    char  name[10];                     /*姓 名*/
    int   computer, english, math;      /*成 绩*/
    double average;                     /*平 均 成 绩*/
} s1, s2;
```

- 定义了结构类型，但是没有名称
- 定义了结构变量：s1, s2
- 因为没有名称，这种结构类型无法再次使用
  - 例如无法再用这种结构定义新的变量

结构变量定义

4. 无名定义 – 不给类型取名称

```
struct
{
    int   num;                          /*学 号*/
    char  name[10];                     /*姓 名*/
    int   computer, english, math;      /*成 绩*/
    double average;                     /*平 均 成 绩*/
} s1, s2;
```

- 定义了结构类型，但是没有名称
- 定义了结构变量：s1, s2
- 因为没有名称，这种结构类型无法再次使用
  - 例如无法再用这种结构定义新的变量
  - 但是已经定义的变量 s1、s2 可以使用，因为有名称

结构变量定义

一个综合的例子

```
struct
{
    int   num;                              /*学号*/
    char  name[10];                         /*姓名*/
    int   computer, english, math;          /*成绩*/
    double average;                         /*平均成绩*/
} s1, s2 = { 123, "小头", 98, 80, 95 };
```

结构变量定义

一个综合的例子

```
struct
{
    int   num;                          /*学 号*/
    char  name[10];                     /*姓 名*/
    int   computer, english, math;      /*成 绩*/
    double average;                     /*平 均 成 绩*/
} s1, s2 = { 123, "小 头", 98, 80, 95 };
```

- 定义了无名结构，定义了结构变量：s1, s2

结构变量定义

一个综合的例子

```
struct
{
    int   num;                              /*学 号*/
    char  name[10];                         /*姓 名*/
    int   computer, english, math;          /*成 绩*/
    double average;                         /*平 均 成 绩*/
} s1, s2 = { 123, "小 头", 98, 80, 95 };
```

- 定义了无名结构，定义了结构变量：s1, s2
- 对变量 s2 进行初始化

结构的成员变量引用和使用

通过点运算符指定成员变量，使用如同普通变量

结构变量名.结构成员名

```
struct student stu;
stu.num = 101;
strcpy(stu.name, "zhang");
stu.math = 90;
stu.english = 80;
stu.computer = 95;
stu.average = (stu.math + stu.english +
               stu.computer)/3.0;
```

结构的成员变量引用和使用

通过点运算符指定成员变量，使用如同普通变量

结构变量名.结构成员名

```
struct student stu;
stu.num = 101;
strcpy(stu.name, "zhang");
stu.math = 90;
stu.english = 80;
stu.computer = 95;
stu.average = (stu.math + stu.english +
               stu.computer)/3.0;
```

结构的成员变量引用和使用

通过点运算符指定成员变量，使用如同普通变量

结构变量名.结构成员名

```
struct student stu;
stu.num = 101;
strcpy(stu.name, "zhang");
stu.math = 90;
stu.english = 80;
stu.computer = 95;
stu.average = (stu.math + stu.english +
               stu.computer)/3.0;

struct friend f; //包含结构嵌套
strcpy(f.addr.city, "Beijing");
```

结构变量的整体赋值

```
struct friend f, g;
......
f = g; //将 g 完整地赋值给 f
```

- 结构赋值时，实际上执行的是内存拷贝

结构变量的整体赋值

```
struct friend f, g;
......
f = g; //将 g 完整地赋值给 f
```

- 结构赋值时，实际上执行的是内存拷贝

用结构变量初始化同类型结构变量

```
struct friend f = {12345, "张飞"}, g = f;
```

- 将变量 g 初始化为 f（不适合全局变量定义）

结构变量的整体赋值

```
struct friend f, g;
......
f = g; //将 g 完整地赋值给 f
```

- 结构赋值时，实际上执行的是内存拷贝

用结构变量初始化同类型结构变量

```
struct friend f = {12345, "张飞"}, g = f;
```

- 将变量 g 初始化为 f（不适合全局变量定义）

例程 9-1 输出平均成绩最高的学生

假设学生的信息包括：学号，姓名，三门课程成绩和平均成绩。输入 n 个学生的成绩，计算平均成绩，并输出平均分最高的学生信息。

定义学生结构

```
struct student
{
    int   num;                              /*学 号*/
    char  name[10];                         /*姓 名*/
    int   computer, english, math;          /*成 绩*/
    double average;                         /*平 均 成 绩*/
};
```

例程 9-1 输出平均成绩最高的学生

假设学生的信息包括：学号，姓名，三门课程成绩和平均成绩。输入 n 个学生的成绩，计算平均成绩，并输出平均分最高的学生信息。

准备读入学生信息

```
int main(void)
{
    struct student max, s;
    int k, n;

    printf("输入学生人数: ");
    scanf("%d", &n);
    printf("输入学生的学号、姓名和三门功课成
        绩: \n");
```

例程 9-1 输出平均成绩最高的学生

假设学生的信息包括：学号，姓名，三门课程成绩和平均成绩。输入 n 个学生的成绩，计算平均成绩，并输出平均分最高的学生信息。

读入学生信息，计算平均分，记录最高分学生

```
for( k = 0; k<n; k++ )
{
    printf("学生%d: ", k);
    scanf("%d%s%d%d%d", &s.num, s.name, &s
        .math, &s.english, &s.computer);
    s.average = (s.math+s.english+s.
        computer)/3.0; //计算平均分
    if( k==0 || s.average > max.average )
        max = s; // 记录高分学生
}
```

例程 9-1 输出平均成绩最高的学生

假设学生的信息包括：学号，姓名，三门课程成绩和平均成绩。输入 n 个学生的成绩，计算平均成绩，并输出平均分最高的学生信息。

输出结果：最高分学生的信息

```
printf("平均成绩最好的学生是：\n");
printf("    学号：%d\n", max.num);
printf("    姓名：%s\n", max.name);
printf("    数学：%d\n", max.math);
printf("    英语：%d\n", max.english);
printf("  计算机：%d\n", max.computer);
printf("平均成绩：%.2f\n", max.average);
```

内容提要

结构类型的定义
- 结构的定义
- 结构变量的定义和初始化
- 结构类型的函数返回值和参数
- 结构和数组
- 结构和指针

结构可以用作函数返回值和函数参数

```
// 复数结构
struct complex {
    double real, imag;
};

// 复数相加
struct complex add(struct complex a,
                   struct complex b)
{
    struct complex sum;
    sum.real = a.real + b.real;
    sum.imag = a.imag + b.imag;
    return sum; // 返回结果是一个结构(复数)
}
```

结构可以用作函数返回值和函数参数

```
// 复数结构
struct complex {
    double real, imag;
};

// 复数相加
struct complex add(struct complex a,
                   struct complex b)
{
    struct complex sum;
    sum.real = a.real + b.real;
    sum.imag = a.imag + b.imag;
    return sum; // 返回结果是一个结构(复数)
}
```

用结构作为参数，还可以简化函数接口

内容提要

结构类型的定义
- 结构的定义
- 结构变量的定义和初始化
- 结构类型的函数返回值和参数
- 结构和数组
- 结构和指针

结构类型的数组

- 结构成员可以是数组，例如 struct student 的 name 成员就是字符数组

结构类型的数组

- 结构成员可以是数组，例如 struct student 的 name 成员就是字符数组
- 当用结构作为数组的元素类型时，就是结构数组

结构类型的数组

- 结构成员可以是数组，例如 struct student 的 name 成员就是字符数组
- 当用结构作为数组的元素类型时，就是结构数组

例如：

```
struct student students[50];
```

结构类型的数组

- 结构成员可以是数组，例如 struct student 的 name 成员就是字符数组
- 当用结构作为数组的元素类型时，就是结构数组

例如：

```
struct student students[50];
```

- 结构数组定义的一般语法是：

```
struct 结构名称 结构数组名称[数组长度];
```

结构类型的数组

- 结构成员可以是数组，例如 struct student 的 name 成员就是字符数组
- 当用结构作为数组的元素类型时，就是结构数组

例如：

```
struct student students[50];
```

- 结构数组定义的一般语法是：

```
struct 结构名称 结构数组名称[数组长度];
```

- 遵循数组的语法规则

结构类型的数组

- 结构成员可以是数组，例如 struct student 的 name 成员就是字符数组
- 当用结构作为数组的元素类型时，就是结构数组

例如：

```
struct student students[50];
```

- 结构数组定义的一般语法是：

```
struct 结构名称 结构数组名称[数组长度];
```

- 遵循数组的语法规则
  - 定义、初始化

结构类型的数组

- 结构成员可以是数组，例如 struct student 的 name 成员就是字符数组
- 当用结构作为数组的元素类型时，就是结构数组

例如：

```
struct student students[50];
```

- 结构数组定义的一般语法是：

```
struct 结构名称 结构数组名称[数组长度];
```

- 遵循数组的语法规则
  - 定义、初始化
- 强调：不要遗漏关键字 struct

结构数组初始化

在定义结构数组变量的时候，可以同时初始化数组元素

结构数组初始化

在定义结构数组变量的时候，可以同时初始化数组元素

- 遵循数组的初始化规则

结构数组初始化

在定义结构数组变量的时候，可以同时初始化数组元素

- 遵循数组的初始化规则　　格式：元素初值列表

结构数组初始化

在定义结构数组变量的时候，可以同时初始化数组元素

- 遵循数组的初始化规则　　格式：元素初值列表
- 遵循结构的初始化规则

结构数组初始化

在定义结构数组变量的时候，可以同时初始化数组元素

- 遵循数组的初始化规则　　格式：元素初值列表
- 遵循结构的初始化规则　　格式：成员初值列表

结构数组初始化

在定义结构数组变量的时候，可以同时初始化数组元素

- 遵循数组的初始化规则　　格式：元素初值列表
- 遵循结构的初始化规则　　格式：成员初值列表
- 类似于二维数组的初始化规则

```
struct student students[50] = {
    {12345, "公孙赞", 78},          // 部分给出
    {12345, "袁绍", 95, 88, 89},// 按顺序对应
    }; // 其余结构元素没有初始化
```

结构数组初始化

在定义结构数组变量的时候，可以同时初始化数组元素

- 遵循数组的初始化规则　　格式：元素初值列表
- 遵循结构的初始化规则　　格式：成员初值列表
- 类似于二维数组的初始化规则

```
struct student students[50] = {
    {12345, "公孙赞", 78},          // 部分给出
    {12345, "袁绍", 95, 88, 89},// 按顺序对应
    }; // 其余结构元素没有初始化

int mat[50][5] = {
    {1, 2, 3, 4, 5},
    {6, 7, 8, 9}, // 部分给出
    };
```

结构数组初始化

在定义结构数组变量的时候，可以同时初始化数组元素

- 遵循数组的初始化规则　　格式：元素初值列表
- 遵循结构的初始化规则　　格式：成员初值列表
- 类似于二维数组的初始化规则

```
struct student students[50] = {
    {12345, "公孙赞", 78},          // 部分给出
    {12345, "袁绍", 95, 88, 89},// 按顺序对应
    }; // 其余结构元素没有初始化

int mat[50][5] = {
    {1, 2, 3, 4, 5},
    {6, 7, 8, 9}, // 部分给出
    };
```

- 有两层大括号

结构数组初始化

在定义结构数组变量的时候，可以同时初始化数组元素

- 遵循数组的初始化规则　　格式：元素初值列表
- 遵循结构的初始化规则　　格式：成员初值列表
- 类似于二维数组的初始化规则

```
struct student students[50] = {
    {12345, "公孙赞", 78},          // 部分给出
    {12345, "袁绍", 95, 88, 89},// 按顺序对应
    }; // 其余结构元素没有初始化

int mat[50][5] = {
    {1, 2, 3, 4, 5},
    {6, 7, 8, 9}, // 部分给出
    };
```

- 有两层大括号　　还可以定义二维的和更高维的结构数组

例程 9-2 学生成绩排序

输入 n 个学生的信息和成绩，计算平均成绩，并按照平均成绩从高到低输出学生的信息。

定义学生结构

```
struct student
{
    int   num;                          /*学号*/
    char  name[10];                     /*姓名*/
    int   computer, english, math;      /*成绩*/
    double average;                     /*平均成绩*/
};
```

例程 9-2 学生成绩排序

输入 n 个学生的信息和成绩，计算平均成绩，并按照平均成绩从高到低输出学生的信息。

准备读入学生信息

```
int main(void)
{
    struct student students[50], s;
    int k, n;

    printf("输入学生人数: ");
    scanf("%d", &n);
    printf("输入学生的学号、姓名和三门功课成
        绩: \n");
```

例程 9-2 学生成绩排序

输入 n 个学生的信息和成绩，计算平均成绩，并按照平均成绩从高到低输出学生的信息。

读入学生信息，计算平均分

```
for( k = 0; k<n; k++ )
{
    scanf("%d%s%d%d%d", &s.num, s.name, &s
        .math, &s.english, &s.computer);
    s.average = (s.math+s.english+s.
        computer)/3.0; //计算平均分
    students[k] = s;
}
```

例程 9-2 学生成绩排序

输入 n 个学生的信息和成绩，计算平均成绩，并按照平均成绩从高到低输出学生的信息。

排序：从高到低，选择法

```
for(k = 0; k<n-1; k++ )
{
    int j, index = k; //最大的
    for( j=k+1; j<n; j++ )
        if( students[j].average>students[
            index].average )
             index = j;//记下新的最大值下标
    s = students[k];  //交换 k 和 index
    students[k] = students[index];
    students[index] = s;
}
```

例程 9-2 学生成绩排序

输入 n 个学生的信息和成绩，计算平均成绩，并按照平均成绩从高到低输出学生的信息。

按顺序输出学生信息和平均分

```
// 输出排序后的信息
printf("按照平均分从高到低\n");
printf("学号     姓名  平均\n");
for( k = 0; k<n; k++ ){
    printf("%4d%7s%6.1f\n", students[k].
        num, students[k].name, students[k].
        average);
}
```

内容提要

结构类型的定义
- 结构的定义
- 结构变量的定义和初始化
- 结构类型的函数返回值和参数
- 结构和数组
- 结构和指针

结构指针 - 指向结构变量的指针

结构指针 - 指向结构变量的指针

- 遵循指针的所有语法规则

结构指针 - 指向结构变量的指针

- 遵循指针的所有语法规则
- 使用指针的定义和使用规则

```
struct student f, *p;
```

结构指针

结构指针的使用：箭头运算符 ->

```
struct student f, *p;
p = &f; // 让指针 p 指向结构变量 f
p->num = 12345;                    // 设置学号
strcpy(p->name, "曹操");           // 设置姓名
p->math = 80;                      // 数学成绩
scanf("%d", &p->computer);// 读入计算机成绩
scanf("%d", &p->english); // 读入英语成绩
p->average = (p->math + p->computer + p->
    english)/3.0; // 计算平均成绩
```

结构指针的使用：箭头运算符 ->

```
struct student f, *p;
p = &f; // 让指针 p 指向结构变量 f
p->num = 12345;                    // 设置学号
strcpy(p->name, "曹操");           // 设置姓名
p->math = 80;                      // 数学成绩
scanf("%d", &p->computer);// 读入计算机成绩
scanf("%d", &p->english); // 读入英语成绩
p->average = (p->math + p->computer + p->
    english)/3.0; // 计算平均成绩
```

- 箭头运算符 -> 的优先级非常高

结构指针的使用：箭头运算符 ->

```
struct student f, *p;
p = &f; // 让指针 p 指向结构变量 f
p->num = 12345;                    // 设置学号
strcpy(p->name, "曹操");           // 设置姓名
p->math = 80;                      // 数学成绩
scanf("%d", &p->computer);// 读入计算机成绩
scanf("%d", &p->english); // 读入英语成绩
p->average = (p->math + p->computer + p->
    english)/3.0; // 计算平均成绩
```

- 箭头运算符 -> 的优先级非常高
  - 同属最高优先级：()，[]，.，->

结构指针

结构指针的使用：箭头运算符 ->

```
struct student f, *p;
p = &f; // 让指针 p 指向结构变量 f
p->num = 12345;                    // 设置学号
strcpy(p->name, "曹操");           // 设置姓名
p->math = 80;                      // 数学成绩
scanf("%d", &p->computer);// 读入计算机成绩
scanf("%d", &p->english); // 读入英语成绩
p->average = (p->math + p->computer + p->
    english)/3.0; // 计算平均成绩
```

- 箭头运算符 -> 的优先级非常高
  - 同属最高优先级：()，[]，.，->
  - 高于所有其他运算符

结构指针

点运算符与箭头运算符：优先级相同

`p->num` 等价于 `(*p).num`
`p->math` 等价于 `(*p).math`

箭头运算符`->`使用起来更方便

结构指针

点运算符与箭头运算符：优先级相同

`p->num` 等价于 `(*p).num`
`p->math` 等价于 `(*p).math`

箭头运算符`->`使用起来更方便

取地址运算符 &，点运算符、箭头运算符

`&p->computer` 等价于 `&(*p).computer`

结构指针

点运算符与箭头运算符：优先级相同

```
p->num   等价于  (*p).num
p->math  等价于  (*p).math
```

箭头运算符->使用起来更方便

取地址运算符 &，点运算符、箭头运算符

```
&p->computer  等价于  &(*p).computer
```

- & 运算符的优先级低于点运算符. 和箭头运算符->

结构指针

点运算符与箭头运算符：优先级相同

```
p->num   等价于  (*p).num
p->math  等价于  (*p).math
```

箭头运算符->使用起来更方便

取地址运算符 &，点运算符、箭头运算符

```
&p->computer  等价于  &(*p).computer
```

- & 运算符的优先级低于点运算符. 和箭头运算符->
- & 运算符作用在成员变量 computer 上

例程 9-3 修改学生成绩

输入 n 个学生的信息和成绩，再根据输入的学号、课程、成绩进行修改。

定义学生结构

```
struct student
{
    int   num;                          /*学号*/
    char  name[10];                     /*姓名*/
    int   computer, english, math;      /*成绩*/
    double average;                     /*平均成绩*/
};
```

例程 9-3 修改学生成绩

输入 n 个学生的信息和成绩，再根据输入的学号、课程、成绩进行修改。

准备读入学生信息

```
int main(void)
{
    struct student students[50], s;
    int k, n;

    printf("输入学生人数: ");
    scanf("%d", &n);
    printf("输入学生的学号、姓名和三门功课成
        绩: \n");
```

例程 9-3 修改学生成绩

输入 n 个学生的信息和成绩，再根据输入的学号、课程、成绩进行修改。

读入学生信息，计算平均分

```
for( k = 0; k<n; k++ )
{
    scanf("%d%s%d%d%d", &s.num, s.name, &s
        .math, &s.english, &s.computer);
    s.average = (s.math+s.english+s.
        computer)/3.0; //计算平均分
    students[k] = s;
}
```

例程 9-3 修改学生成绩

输入 n 个学生的信息和成绩，再根据输入的学号、课程、成绩进行修改。

修改成绩

```
while( 1 ) { // 修改
    int num, course, score;
    printf("输入需修改的学号、课程、成绩: ");
    scanf("%d%d%d", &num, &course, &score);
    if( update_score(students, n, num, course,
        score)<0 )
        break;
}
```

调用函数进行修改操作

```
int update_score(struct student *s, int n, int
    num, int course, int score);
```

例程 9-3 修改学生成绩

输入 n 个学生的信息和成绩，再根据输入的学号、课程、成绩进行修改。

修改成绩函数：update_score

```
int update_score(struct student *p, int n, int
    num, int course, int score)
{
    int k;
    for( k=0; k<n; k++, p++ )
        if( p->num==num ) {
            switch( course ) {
                case 1: p->math = score;
                        update_average(p);
                        return k;
```

例程 9-3 修改学生成绩

输入 n 个学生的信息和成绩，再根据输入的学号、课程、成绩进行修改。

修改成绩函数：update_score

```
                case 2: p->english = score;
                        update_average(p);
                        return k;
                case 3: p->computer = score;
                        update_average(p);
                        return k;
            }
            return -1; // 修改失败
        }
    return -1; // 学号num无对应学生
}
```

例程 9-3 修改学生成绩

输入 n 个学生的信息和成绩，再根据输入的学号、课程、成绩进行修改。

重新计算平均成绩：update_average

```
void update_average(struct student *p)
{
    p->average = (p->math + p->english + p->
        computer)/3.0;
}
```

总结

- 什么是结构
- 结构的定义、使用、变量初始化
- 结构的存储形式和规则
- 结构数组、结构指针
- 结构返回值和结构参数
- 结构数组的排序
- 点运算符.
- 箭头运算符->

今天到此为止