

08 - 结构体 II

C++ 程序设计进阶

SOJ 信息学竞赛教练组

2024 年 11 月 8 日

目录

1 复习回顾

2 构造函数

3 运算符函数

4 总结

- 结构体的定义

```
1 struct Student {  
2     string name;  
3     double s1, s2, sum;  
4     void output() {  
5         cout << name << " " << sum << endl;  
6     }  
7 };
```

- 结构体的定义

```
1 struct Student {  
2     string name;  
3     double s1, s2, sum;  
4     void output() {  
5         cout << name << " " << sum << endl;  
6     }  
7 };
```

- 结构体包含**成员变量**（属性）和**成员函数**（行为）

复习回顾

- 结构体的定义

```
1 struct Student {  
2     string name;  
3     double s1, s2, sum; 一般不写 sum = s1 + s2;  
4     void output() {  
5         cout << name << " " << sum << endl;  
6     }  
7 };
```

- 结构体包含**成员变量**（属性）和**成员函数**（行为）
- 成员变量在声明时一般不会进行初始化

复习回顾

- 结构体的定义

```
1 struct Student {  
2     string name;  
3     double s1, s2, sum; 一般不写 sum = s1 + s2;  
4     void output() {  
5         cout << name << " " << sum << endl;  
6     }  
7 };
```

- 结构体包含**成员变量**（属性）和**成员函数**（行为）
- 成员变量在声明时一般不会进行初始化
- 成员函数可以直接使用结构体内的成员变量

复习回顾

- 结构体的使用
 - 声明与初始化

```
1 Student a = {"Tom", 80, 82.5, 80 + 82.5};
```

复习回顾

- 结构体的使用

- 声明与初始化

- ```
1 Student a = {"Tom", 80, 82.5, 80 + 82.5};
```

- 整体赋值

- ```
1 Student b = a;
```


复习回顾

- 结构体的使用

- 声明与初始化

```
1 Student a = {"Tom", 80, 82.5, 80 + 82.5};
```

- 整体赋值

```
1 Student b = a;
```

- 访问成员列表

```
1 a.sum = a.s1 + a.s2;    // 访问成员变量  
2 a.output();             // 访问成员函数
```

```
1 Student.s1 = 1;         // 可行吗?  
2 Student.output();
```

复习回顾

- 结构体的使用

- 声明与初始化

```
1 Student a = {"Tom", 80, 82.5, 80 + 82.5};
```

- 整体赋值

```
1 Student b = a;
```

- 访问成员列表

```
1 a.sum = a.s1 + a.s2;    // 访问成员变量
2 a.output();             // 访问成员函数
```

```
1 Student.s1 = 1;      // 可行吗?
2 Student.output();
```

上节课使用 `{ }` 可以将结构体进行初始化，
那除此之外还有其他快捷初始化的方法吗？

目录

1 复习回顾

2 构造函数

3 运算符函数

4 总结

构造函数

- 构造函数常用在声明结构体变量时**初始化成员变量**
- 构造函数的定义
 - 函数名(参数列表)
 - 构造函数不能写返回值类型
 - 构造函数的函数名只能是结构体名

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student() {  
6         id = score = 0;  
7     }  
8 };
```

构造函数

- 构造函数常用在声明结构体变量时初始化成员变量
- 构造函数的定义
 - 函数名(参数列表)
 - 构造函数不能写返回值类型
 - 构造函数的函数名只能是结构体名

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student() {  
6         id = score = 0;  
7     }  
8 };
```

构造函数

- 构造函数常用在声明结构体变量时**初始化成员变量**
- 构造函数的定义
 - 函数名(参数列表)
 - 构造函数不能写返回值类型
 - 构造函数的函数名只能是结构体名
- 使用结构体声明变量时会自动调用构造函数进行初始化

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student() {  
6         id = score = 0;  
7     }  
8 };  
9  
10 int main() {  
11     Student a;  
12     cout << a.score << endl;  
13     // 输出 0  
14  
15     return 0;  
16 }
```

默认调用该构造函数

构造函数

```
1 #include <iostream>
2 using namespace std;
3
4 struct Student {
5     string name;
6     int id, score;
7     // 构造函数
8     Student() {
9         id = 1;
10        score = 60;
11    }
12 };
13
14 int main() {
15     Student a;
16     cout << a.score << endl;    // 输出 60
17
18     return 0;
19 }
```


构造函数

```
1 #include <iostream>
2 using namespace std;
3
4 struct Student {
5     string name;
6     int id, score;
7     // 构造函数
8     Student() {
9         id = 1;
10        score = 60;
11    }
12 };
13
14 int main() {
15     Student a;
16     cout << a.score << endl;    // 输出 60
17
18     return 0;
19 }
```

- 构造函数可以根据需要初始化成员变量的值

构造函数

- 构造函数可以传入参数，
用参数初始化成员变量

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student(string na, int i, int s) {  
6         name = na;  
7         id = i;  
8         score = s;  
9     }  
10 };
```

构造函数

- 构造函数可以传入参数，用参数初始化成员变量

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student(string na, int i, int s) {  
6         name = na;  
7         id = i;  
8         score = s;  
9     }  
10 };
```

构造函数

- 构造函数可以传入参数，用参数初始化成员变量
- 声明结构体变量：
结构体名 变量名(参数)

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student(string na, int i, int s) {  
6         name = na;  
7         id = i;  
8         score = s;  
9     }  
10 };  
11  
12 int main() {  
13     Student a("Tom", 1, 80);  
14     cout << a.score << endl;  
15     // 输出 80  
16  
17     return 0;  
18 }
```

构造函数

- 构造函数可以传入参数，用参数初始化成员变量
- 声明结构体变量：
结构体名 变量名(参数)
- 此时还可以使用
Student b;
声明结构体变量吗？

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student(string na, int i, int s) {  
6         name = na;  
7         id = i;  
8         score = s;  
9     }  
10 };  
11  
12 int main() {  
13     Student a("Tom", 1, 80);  
14     Student b;  
15  
16     return 0;  
17 }
```

构造函数

- 构造函数可以传入参数，用参数初始化成员变量
- 声明结构体变量：
结构体名 变量名(参数)
- 此时还可以使用
Student b;
声明结构体变量吗？

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student(string na, int i, int s) {  
6         name = na;  
7         id = i;  
8         score = s;  
9     }  
10 };  
11  
12 int main() {  
13     Student a("Tom", 1, 80);  
14     Student b;  
15  
16     return 0;  
17 }
```

无法调用该构造函数

构造函数

- 一个结构体可以有多个构造函数，但每个构造函数的参数列表必须不同

```
1 struct Student {
2     string name;
3     int id, score;
4     // 构造函数
5     Student() {
6         id = score = 0;
7     }
8     Student(string na, int i, int s) {
9         name = na;
10        id = i;
11        score = s;
12    }
13 };
14
15 int main() {
16     Student a("Tom", 1, 80);
17     Student b;
18
19     return 0;
20 }
```

构造函数

- 一个结构体可以有多个构造函数，但每个构造函数的参数列表必须不同

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student() {  
6         id = score = 0;  
7     }  
8     Student(string na, int i, int s) {  
9         name = na;  
10        id = i;  
11        score = s;  
12    }  
13 };  
14  
15 int main() {  
16     Student a("Tom", 1, 80);  
17     Student b;  
18  
19     return 0;  
20 }
```


构造函数

- 一个结构体可以有多个构造函数，但每个构造函数的参数列表必须不同
- 声明结构体变量的写法不同，会调用不同的构造函数

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student() {  
6         id = score = 0;  
7     }  
8     Student(string na, int i, int s) {  
9         name = na;  
10        id = i;  
11        score = s;  
12    }  
13 };  
14  
15 int main() {  
16     Student a("Tom", 1, 80);  
17     Student b;  
18  
19     return 0;  
20 }
```

自动调用该构造函数

构造函数

- 一个结构体可以有多个构造函数，但每个构造函数的参数列表必须不同
- 声明结构体变量的写法不同，会调用不同的构造函数

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     // 构造函数  
5     Student() {  
6         id = score = 0;  
7     }  
8     Student(string na, int i, int s) {  
9         name = na;  
10        id = i;  
11        score = s;  
12    }  
13 };  
14  
15 int main() {  
16     Student a("Tom", 1, 80);  
17     Student b;  
18  
19     return 0;  
20 }
```

自动调用该构造函数

例 8.1：计算学生的成绩

编程题

- 编写一个 Student 结构体，成员变量包含姓名 *name*、学号 *id*、分数 *score* (*name* 是字符串，*id*、*score* 是整数) 使用构造函数解决以下问题：
已知班级里有 n ($1 \leq n \leq 100$) 个学生，并且学生的平均分为 60 分。每个学生与平均分都有一定的差距 d ， d 为正数表示比平均分高多少分，负数表示比平均分低多少分。
最后输出 n 个学生的 *name*、*id*、*score*
- 样例输入
3
Tom 1 4
Lucy 2 -3
Ken 3 -1
- 样例输出
Tom 1 64
Lucy 2 57
Ken 3 59

例 8.1：计算学生的成绩

```
1 struct Student {
2     string name;
3     int id, score, d; // d 表示该学生与平均分的差距
4     Student() {
5         score = 60;    // 先假设每个学生的分数都是平均分
6     }
7     void input() {
8         cin >> name >> id >> d;
9         score += d;    // 通过 d 计算出该学生真实的 score
10    }
11    void output() {
12        cout << name << " " << id << " " << score << endl;
13    }
14 };
15
16 Student s[110];
17 int main() {
18     int n;
19     cin >> n;
20     for (int i = 0; i < n; i++) s[i].input();
21     for (int i = 0; i < n; i++) s[i].output();
22     return 0;
23 }
```

目录

1 复习回顾

2 构造函数

3 运算符函数

4 总结

- 已知两个 string 类型的变量可以通过 < 进行比较

```
1 string a = "abc";  
2 string b = "ABC";  
3 if (a < b) cout << "字符串 a 比较小";
```

- 已知两个 string 类型的变量可以通过 < 进行比较

```
1 string a = "abc";  
2 string b = "ABC";  
3 if (a < b) cout << "字符串 a 比较小";
```

- 当比较两个结构体变量的大小关系时，可以直接使用 < 吗？

```
1 Student a = {"Tom", 1, 85};    // Tom 同学学号为 1, 分数为 85  
2 Student b = {"Lucy", 2, 80};   // Lucy 同学学号为 2, 分数为 80  
3 if (a < b) cout << "学生 a 比较小";
```

- 已知两个 string 类型的变量可以通过 < 进行比较

```
1 string a = "abc";  
2 string b = "ABC";  
3 if (a < b) cout << "字符串 a 比较小";
```

- 当比较两个结构体变量的大小关系时，可以直接使用 < 吗？

```
1 Student a = {"Tom", 1, 85};    // Tom 同学学号为 1, 分数为 85  
2 Student b = {"Lucy", 2, 80};   // Lucy 同学学号为 2, 分数为 80  
3 if (a < b) cout << "学生 a 比较小";
```


- 运算符函数使得结构体类型变量支持符号运算

运算符函数

- 运算符函数使得结构体类型变量支持符号运算
- 常被定义（或重载）的运算符函数有：
 - 算术运算符：+ - * / %
 - 关系运算符：< > <= >= == !=

运算符函数

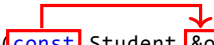
- 运算符函数的函数头与普通函数相似，不同的是：
 - 函数名由 **operator 运算符** 组成
 - 参数列表往往需要 **const** 和 **&**

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     bool operator < (const Student &o) const {  
5         // 形参 o 有成员变量 o.name, o.id, o.score  
6         return 运算结果  
7     }  
8 };
```

运算符函数

- 运算符函数的函数头与普通函数相似，不同的是：
 - 函数名由 **operator 运算符** 组成
 - 参数列表往往需要 **const** 和 **&**

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     bool operator < (const Student &o) const {  
5         // 形参 o 有成员变量 o.name, o.id, o.score  
6         return 运算结果  
7     }  
8 };
```



保证形参不被修改

运算符函数

- 运算符函数的函数头与普通函数相似，不同的是：
 - 函数名由 **operator 运算符** 组成
 - 参数列表往往需要 **const** 和 **&**

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     bool operator < (const Student &o) const {  
5         // 形参 o 有成员变量 o.name, o.id, o.score  
6         return 运算结果  
7     }  
8 };
```

保证自己的成员变量不被修改

保证形参不被修改

< 运算符函数

- 运算符函数的功能是设计比较规则

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     bool operator < (const Student &o) const {  
5         // 设计比较规则:  
6         // 如果 自己 < 形参, 返回 true  
7         // 否则 返回 false  
8     }  
9 };
```

< 运算符函数

- 运算符函数的功能是设计比较规则

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     bool operator < (const Student &o) const {  
5         // 设计比较规则:  
6         if (score < o.score) return true;  
7         else return false;  
8     }  
9 };
```

例 8.2：比较两个学生的大小关系

编程题

- 编写一个 Student 结构体，成员变量包含姓名 *name*、学号 *id*、分数 *score*。
输入两个学生的信息，使用 < 比较两个结构体变量，输出其中分数较小的学生的名字
- 样例输入
Tom 1 89
Lucy 2 75
- 样例输出
Lucy

例 8.2：比较两个学生的大小关系

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     void input() {  
5         cin >> name >> id >> score  
6     }  
7     bool operator < (const Student &o) const {  
8         // 如果第一个学生的分数比较小：  
9         if (score < o.score) return true;  
10        else return false;  
11    }  
12 };
```

例 8.2：比较两个学生的大小关系

```
1 struct Student {
2     string name;
3     int id, score;
4     void input() {
5         cin >> name >> id >> score
6     }
7     bool operator < (const Student &o) const {
8         // 如果第一个学生的分数比较小,
9         // 由于 score < o.score 结果为 true, 直接返回
10        // 否则 score < o.score 结果为 false, 也直接返回
11        return score < o.score;
12    }
13 };
```

例 8.2：比较两个学生的大小关系

```
1 struct Student {
2     string name;
3     int id, score;
4     void input() {
5         cin >> name >> id >> score
6     }
7     bool operator < (const Student &o) const {
8         return score < o.score;
9     }
10 };
11
12 int main() {
13     Student a, b;
14     a.input();
15     b.input();
16     // 学生 a 调用了 < 运算符函数
17     // 如果 a 的分数比 b 小，就返回 true，否则返回 false
18     if (a < b) cout << a.name << endl;
19     else cout << b.name << endl;
20 }
```

例 8.2：比较两个学生的大小关系

```
1 struct Student {
2     string name;
3     int id, score;
4     void input() {
5         cin >> name >> id >> score
6     }
7     bool operator < (const Student &o) const {
8         return score < o.score;
9     }
10 };
11
12 int main() {
13     Student a, b;
14     a.input();
15     b.input();
16     // 学生 a 调用了 < 运算符函数
17     // 如果 a 的分数比 b 小，就返回 true，否则返回 false
18     if (a < b) cout << a.name << endl;
19     else cout << b.name << endl;
20 }
```

- 重载的运算符只有 <，所以不能使用 >、<= 等进行比较大小

> 运算符函数

- 当想要使用 > 比较两个结构体变量时，必须重载 > 运算符函数

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     void input() {  
5         cin >> name >> id >> score  
6     }  
7     bool operator > (const Student &o) const {  
8         return id > o.id;  
9     }  
10 };
```

> 运算符函数

- 当想要使用 > 比较两个结构体变量时，必须重载 > 运算符函数

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     void input() {  
5         cin >> name >> id >> score  
6     }  
7     bool operator > (const Student &o) const {  
8         return id > o.id;  
9     }  
10 };
```

- 该运算符函数比较的是什么？

其他运算符函数

- 同理，当需要判断两个结构体变量的 == 或 >= 等关系时，也可以重载对应的运算符函数

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     void input() {  
5         cin >> name >> id >> score  
6     }  
7     // 重载 == 运算符，比较分数是否相等  
8  
9  
10 };
```

其他运算符函数

- 同理，当需要判断两个结构体变量的 == 或 >= 等关系时，也可以重载对应的运算符函数

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     void input() {  
5         cin >> name >> id >> score  
6     }  
7     bool operator == (const Student &o) const {  
8         return score == o.score;  
9     }  
10 };
```


其他运算符函数

- 同理，当需要判断两个结构体变量的 == 或 >= 等关系时，也可以重载对应的运算符函数

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     void input() {  
5         cin >> name >> id >> score  
6     }  
7     bool operator >= (const Student &o) const {  
8         return score >= o.score;  
9     }  
10 };
```

多个比较规则的运算符函数

当两个学生结构体变量需要先比较总分谁大，
如果总分相同再比较姓名字典序谁大时，该怎么办？

多个比较规则的运算符函数

- 可以重载多个运算符函数来进行比较吗?
 - 重载多个不同的运算符函数是可以的

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     bool operator > (const Student &o) const {  
5         return score > o.score;  
6     }  
7     bool operator == (const Student &o) const {  
8         return score == o.score;  
9     }  
10    bool operator > (const Student &o) const {  
11        return name > o.name;  
12    }  
13 };
```

多个比较规则的运算符函数

- 可以重载多个运算符函数来进行比较吗?
 - 重载多个不同的运算符函数是可以的
 - 但不可以多次重载相同的运算符函数

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     bool operator > (const Student &o) const {  
5         return score > o.score;  
6     }  
7     bool operator == (const Student &o) const {  
8         return score == o.score;  
9     }  
10    bool operator > (const Student &o) const {  
11        return name > o.name;  
12    }  
13 };
```

多个比较规则的运算符函数

- 可以重载多个运算符函数来进行比较吗?
 - 重载多个不同的运算符函数是可以的
 - 但不可以多次重载相同的运算符函数
- 理论上可行，但有没有更简单的写法？

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     bool operator > (const Student &o) const {  
5         return score > o.score;  
6     }  
7     bool operator == (const Student &o) const {  
8         return score == o.score;  
9     }  
10    bool cmp2(const Student &o) const {  
11        return name > o.name;  
12    }  
13 };
```

多个比较规则的运算符函数

- 可以将多个比较规则合并到一个运算符函数里：

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     bool operator > (const Student &o) const {  
5         // 能使用 score 的前提是 score 不同  
6         if (score != o.score) return score > o.score;  
7         // score 相同时才使用 name 做比较  
8         else return name > o.name;  
9     }  
10 };
```

多个比较规则的运算符函数

- 可以将多个比较规则合并到一个运算符函数里：

```
1 struct Student {  
2     string name;  
3     int id, score;  
4     bool operator > (const Student &o) const {  
5         // 能使用 score 的前提是 score 不同  
6         if (score != o.score) return score > o.score;  
7         // score 相同时才使用 name 做比较  
8         // 不需要写 else:  
9         // 因为一旦 if 的条件成立, 函数就 return 结束了  
10        return name > o.name;  
11    }  
12 };
```

例 8.3：成绩最优异的学生

编程题

- 编写一个 Student 结构体，成员变量包含姓名 *name*、学号 *id*、分数 *score*。定义一个 $>$ 运算符函数，用于比较学生分数的关系，找到 n ($1 \leq n \leq 100$) 个学生里分数最高的学生。如果有多个同学的分数并列最高，则输出学号最大的学生的名字
- 样例输入
3
Tom 3 90
Lucy 1 90
Ken 2 85
- 样例输出
Tom

例 8.3：成绩最优异的学生

```
1 struct Student {
2     string name;
3     int id, score;
4     void input() {
5         cin >> name >> id >> score;
6     }
7     bool operator > (const Student &o) const {
8         if (score != o.score) return score > o.score;
9         return id > o.id;
10    }
11 };
12 Student s[110];
13 int main() {
14     int n;
15     cin >> n;
16     for (int i = 0; i < n; i++) s[i].input();
17     Student mx;
18     for (int i = 0; i < n; i++) {
19         if (i == 0; s[i] > mx) mx = s[i];
20     }
21     cout << mx.name << endl;
22     return 0;
23 }
```

例 8.4：时间运算

编程题

- 给出 n 个时间，每个时间用小时和分钟表示。希望你计算所有时间的和并输出。

- 样例输入

4

1 15

0 56

5 12

3 8

- 样例输出

10 31

例 8.4：时间运算

```
1 struct Time {
2     int hour, minute;
3     void input() {
4         cin >> hour >> minute;
5     }
6     void output() {
7         cout << hour << " " << minute << endl;
8     }
9     Time operator + (const Time &o) {
10        Time res;    // res 记录当前时间和 o 时间相加的结果
11        res.hour = hour + o.hour;
12        res.minute = minute + o.minute;
13        if (res.minute >= 60) {
14            res.hour++;
15            res.minute -= 60;
16        }
17        return res;
18    }
19 };
```

例 8.4：时间运算

```
1 int main() {  
2     int n;  
3     cin >> n;  
4     Time sum = {0, 0}; // 初始化: sum 的小时数和分钟数都为 0  
5     for (int i = 1; i <= n; i++) {  
6         Time tmp;  
7         tmp.input();  
8         sum = sum + tmp;  
9     }  
10    sum.output();  
11  
12    return 0;  
13 }
```

目录

1 复习回顾

2 构造函数

3 运算符函数

4 总结

总结

- 构造函数
 - 有、无参构造函数
- 运算符函数
 - $<$ 、 $>$ 、 $+$ 运算符函数
- 运算符函数应用
 - 多个比较规则
 - 结构体数组找最值