



类和对象

补：构造函数和析构函数



目录

- 对象的初始化方法
- 构造函数的基本概念
- 构造函数的使用
- 析构函数的基本概念
- 析构函数的使用



补.1 对象的初始化方法

- 对象的初始化方法

(1) 若全部成员都是公有,可按结构体的方式进行初始化

(若有私有成员,不能用此方法)

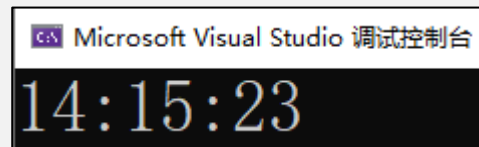
(2) 写一个赋初值的公有成员函数,在其它成员被调用之前进行调用

(3) 声明类时对数据成员进行初始化 (C++11标准支持)



(1) 若全部成员都是公有, 可按结构体的方式进行初始化

```
#include <iostream>
using namespace std;
class Time {
    public:
        int hour;
        int minute;
        int sec;
        int f2() { return 0; }; //成员函数不占空间
};
int main()
{
    Time t1={14, 15, 23};
    cout << t1.hour << ':' << t1.minute << ':' << t1.sec << endl;
}
```



(13,27): error C2440: “初始化”: 无法从“int”转换为“Time”

(13,27): message : 无构造函数可以接受源类型, 或构造函数重载决策不明确

以下两种形式均报错:

Time t1=(14, 15, 23);

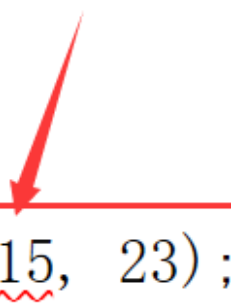
Time t1(14, 15, 23);

```
1  #include <iostream>
2  using namespace std;
3
4  class Time {
5  public:
6      int hour;
7      int minute;
8      int sec;
9      int f2() { return 0; }; //成员函数不占空间
10 };
11 int main()
12 {
13     Time t1 = (14, 15, 23);
14     cout << t1.hour << ':' << t1.minute << ':' << t1.sec << endl;
15 }
```

(13, 23): error C2440: “初始化”: 无法从“initializer list”转换为“Time”
(13, 23): message : 无构造函数可以接受源类型, 或构造函数重载决策不明确

```
1  #include <iostream>
2  using namespace std;
3
4  class Time {
5  public:
6      int hour;
7      int minute;
8      int sec;
9      int f2() { return 0; }; //成员函数不占空间
10 };
11 int main()
12 {
13     Time t1(14, 15, 23);
14     cout << t1.hour << ':' << t1.minute << ':' << t1.sec << endl;
15 }
```

以下两种形式均报错:
Time t1=(14, 15, 23);
Time t1(14, 15, 23);





error C2440: “初始化”: 无法从“initializer list”转换为“Time”
message : 无构造函数可以接受源类型, 或构造函数重载决策不明确

```
2   using namespace std;
3   class Time {
4   public:
5       int hour;
6       int minute;
7   private:
8       int sec;
9       int f2() { return 0; }; //成员函数不占空间
10  };
11  int main()
12  {
13      Time t1 = { 14, 15, 23 };
14      cout << t1.hour << ' :' << t1.minute << endl;
15  }
```

缺点: 若有私有数据成员, 不能用此方法

(2) 写一个赋初值的公有成员函数， 在其它成员被调用之前进行调用



```
class Time {  
    private:  
        int hour;  
        int minute;  
        int sec;  
    public:  
        void set(int h, int m, int s)  
        {  
            hour=h;  
            minute=m;  
            sec=s;  
        }  
};
```

```
int main()  
{  
    Time t;  
    t.set(14, 15, 23);  
    t.其它  
}
```




(3) 声明类时对数据成员进行初始化 (C++11标准支持)

```
class Time {  
    public:  
        int hour    = 0;  
        int minute = 0;  
        int sec     = 0;  
};
```

缺点：不同对象的值被统一初始化，无法个性化

➡ 需要引入合适的初始化方式!!! ➡ 构造函数!!!



目录

- 对象的初始化方法
- 构造函数的基本概念
- 构造函数的使用
- 析构函数的基本概念
- 析构函数的使用



补.2 构造函数的基本概念

- 构造函数的功能：
 - 完成对象的初始化工作, 对象建立时被**自动调用**
- 构造函数的形式：
 - 与类同名, 无返回类型**(非void, 也不是缺省int)**
 - 构造函数必须**公有**
 - 构造函数既可以**体内实现**, 也可以**体外实现**



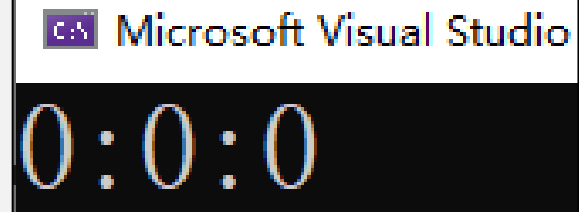
```
class Time {  
    private:  
        int hour, minute, sec; //相同类型可以写在一行上  
    public:
```

```
    Time() 无返回值(非void, 也不是缺省int)  
    {  
        hour    = 0;  
        minute  = 0;  
        sec     = 0;  
    }
```

体内实现

```
    void display()  
    {  
        cout << hour << ':' << minute << ':' << sec << endl;  
    }  
};
```

```
int main()  
{  
    Time t;  
    t.display();  
}
```



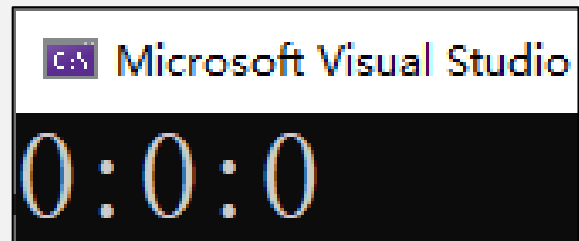


```
class Time {  
    private:  
        int hour, minute, sec; //相同类型可以写在一行上  
    public:  
        Time(); //函数声明  
        void display()  
        {    cout << hour << ':' << minute << ':' << sec << endl;  
        }  
};
```

```
Time::Time() 无返回值(非void, 也不是缺省int)  
{    hour=0;  
    minute=0;  
    sec=0;  
}
```

体外实现

```
int main()  
{  
    Time t;  
    t.display();  
}
```





目录

- 对象的初始化方法
- 构造函数的基本概念
- 构造函数的使用
- 析构函数的基本概念
- 析构函数的使用



补.3 构造函数的使用

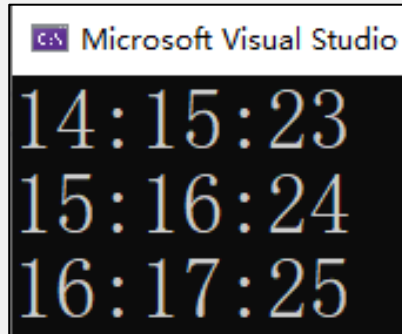
- 构造函数的使用
 - 若不指定构造函数，则系统缺省生成一个**无参空体**构造函数
 - 若用户定义了构造函数，则缺省构造函数**不再存在**
 - 允许定义带参数的构造函数，以解决无参构造函数初始化各对象的值相同的情况

```
class Time {
private:
    int hour, minute, sec;
public:
    Time(int h, int m, int s)
    {
        hour    = h;
        minute   = m;
        sec      = s;
    }
    void display()
    {
        cout << hour << ':' << minute << ':' << sec << endl;
    }
};
```

体内实现

```
int main()
{
    Time t1 (14, 15, 23);
    Time t2 {15, 16, 24};
    Time t3={16, 17, 25};
    t1.display();
    t2.display();
    t3.display();
}
```

三种形式均可





```
class Time {  
    private:  
        int hour, minute, sec;  
    public:  
        Time(int h, int m, int s);  
        void display()  
        {  
            cout << hour << ':' << minute << ':' << sec << endl;  
        }  
};
```

```
Time::Time(int h, int m, int s)  
{  
    hour    = h;  
    minute  = m;  
    sec     = s;  
}
```

体外实现

```
int main()  
{  
    Time t1 (14, 15, 23);  
    Time t2 {15, 16, 24};  
    Time t3={16, 17, 25};  
    t1.display();  
    t2.display();  
    t3.display();  
}
```

三种形式均可

Microsoft Visual Studio

```
14:15:23  
15:16:24  
16:17:25
```

```
(22, 19): error C2661: "Time::Time": 没有重载函数接受 2 个参数
(23, 10): error C2512: "Time": 没有合适的默认构造函数可用
(4, 7): message : 参见 "Time" 的声明
```

```
public:
```

```
    Time(int h, int m, int s);
```

```
    void display()
```

```
    {    cout << hour << ':' << minute << ':' << sec << endl;
    }
```

```
};
```

```
Time::Time(int h, int m, int s)
```

```
{    hour    = h;
```

```
    minute = m;
```

```
    sec    = s;
```

```
}
```

体外实现

//仅三参构造，则不符合会报错

```
int main()
```

```
{
```

```
    Time t1(14, 15);
```

```
    Time t2;
```

```
}
```



补.3 构造函数的使用

- 构造函数的使用（续）
 - 有参构造函数可以使用参数初始化表来对数据成员进行初始化
(仅适用于简单的赋值)

```
//无法通过参数初始化表实现的案例
Time::Time(int h, int m, int s)
{
    if (h>=0 && h<=23)
        hour = h;
    else
        hour = 0;
}
```

(构造函数初始值列表的引入原因等具体分析为后续课程内容，此处掌握基本概念即可)



```
class Time {  
    private:  
        int hour, minute, sec;  
    public:  
        Time(int h, int m, int s): hour(h), minute(m), sec(s)  
        {  
            函数体为空  
        }  
        void display()  
        {  
            cout << hour << ':' << minute << ':' << sec << endl;  
        }  
};  
int main()  
{  
    Time t1(14, 15, 23);  
    t1.display();  
}
```

成员名

参数名

h初始化hour
m初始化minute
s初始化sec

体内实现

Microsoft Visual Studio 调试控制台

14:15:23



```
class Time {  
    private:  
        int hour, minute, sec;  
    public:  
        Time(int h, int m, int s); //函数声明不允许带初始化表  
        void display()  
        {  
            cout << hour << ':' << minute << ':' << sec << endl;  
        }  
};
```

```
Time::Time(int h, int m, int s): hour(h), minute(m), sec(s)  
{  
    // 函数体为空      体外实现  
}
```

```
int main()  
{  
    Time t1(14, 15, 23);  
    t1.display();  
}
```

成员名

参数名

h初始化hour
m初始化minute
s初始化sec

Microsoft Visual Studio 调试控制台
14:15:23



补.3 构造函数的使用

- 构造函数的使用（续）
 - 构造函数**允许重载**
 - 构造函数允许带默认参数，但要注意可能与重载产生**二义性冲突**
 - 构造函数也可以**显式调用**，一般用于带参构造函数



- 构造函数允许重载

```
class Time {  
    ...  
    public:  
        Time();  
        Time(int h, int m, int s);  
};  
Time::Time()  
{  
    hour    = 0;  
    minute  = 0;  
    sec     = 0;  
}  
Time::Time(int h, int m, int s)  
{  
    hour    = h;  
    minute  = m;  
    sec     = s;  
}
```

```
int main()  
{  
    Time t1(14, 15, 23); //正确  
    Time t2;             //正确  
    ...  
}
```



- 构造函数允许带默认参数，但要注意可能与重载产生二义性冲突

```
class Time {  
    ...  
    public:  
        Time(int h, int m, int s=0);  
};
```

```
Time::Time(int h, int m, int s)  
{  
    hour    = h;  
    minute  = m;  
    sec     = s;  
}
```

```
int main()  
{  
    Time t1(14, 15, 23); //正确  
    Time t2(14, 15);     //正确  
}
```




- 构造函数允许带默认参数，但要注意可能与重载产生二义性冲突

```
class Time {  
    ...  
    public:  
        Time();  
        Time(int h, int m, int s=0);  
};  
  
Time::Time()  
{  
    hour    = 0;  
    minute  = 0;  
    sec     = 0;  
}  
  
Time::Time(int h, int m, int s)  
{  
    hour    = h;  
    minute  = m;  
    sec     = s;  
}
```

无参与带缺省参数的重载，不冲突
适应带0/2/3个参数的情况

```
int main()  
{  
    Time t1(14, 15, 23); //正确  
    Time t2(14, 15);     //正确  
    Time t3;             //正确  
}
```

- 构造函数允许带默认参数，但要注意可能与重载产生二义性冲突



```
class Time {  
    ...  
    public:  
        Time(int h=0, int m=0, int s=0);  
};
```

一个函数带三个缺省参数，
适应0/1/2/3个参数等4种情况

```
Time::Time(int h, int m, int s) {  
    {  
        hour    = h;  
        minute  = m;  
        sec     = s;  
    }  
}
```

```
int main()  
{  
    Time t1(14, 15, 23); //正确  
    Time t2(14, 15);     //正确  
    Time t3(14);         //正确  
    Time t4;             //正确  
}
```

- 构造函数允许带默认参数，但要注意可能与重载产生二义性冲突



```
class Time {  
    private:  
        int hour, minute, sec;  
    public:  
        Time()  
        {  
            hour = 0;  
            minute = 0;  
            sec = 0;  
        }  
        Time(int h=0, int m=0, int s=0) : hour(h), minute(m), sec(s)  
        {  
        }  
        void display()  
        {  
            cout << hour << ':' << minute << ':' << sec << endl;  
        }  
};
```

```
int main()  
{  
    Time t1(14, 15, 23); //3参正确  
    Time t2(14, 15);      //2参正确  
    Time t3(14);          //1参正确  
    Time t4;              //无参错误  
}
```

(25): error C2668: "Time::Time": 对重载函数的调用不明确
(14,5): message : 可能是 "Time::Time(int, int, int)"
(8,5): message : 或 "Time::Time(void)"
(25,12): message : 尝试匹配参数列表 "()" 时

无参与带缺省参数的重载，冲突!!!



- 构造函数也可以显式调用，一般用于带参构造函数

```
class Test {  
    private:  
        int a;  
    public:  
        Test(int x) {  
            a=x;  
        }  
};
```

```
Test fun()  
{  
    ...  
    return Test(10); //显式  
}  
  
int main()  
{  
    Test t1(10); //隐式  
    Test t2=Test(10); //显式  
    Test t3=Test{10}; //显式  
}
```



目录

- 对象的初始化方法
- 构造函数的基本概念
- 构造函数的使用
- 析构函数的基本概念
- 析构函数的使用



补.4 析构函数的基本概念

- 析构函数的功能：
 - 在对象被撤销时(生命期结束)时被**自动调用**
 - 完成一些善后工作(主要是内存清理)，但不是撤销对象本身
- 析构函数的形式：
 - ~类名();
 - 无返回值(**非void, 也不是int**)，无参，不允许重载
 - 析构函数必须**公有**
 - 析构函数既可以**体内实现**，也可以**体外实现**



- 构造函数与析构函数的形式:

```
class Time {  
    ...  
    public:  
        Time() //构造体内实现  
        { ...  
        }  
        ~Time() //析构体内实现  
        { ...  
        }  
};
```

```
class Time {  
    ...  
    public:  
        Time(); //构造声明  
        ~Time(); //析构声明  
};  
Time::Time() //构造体外实现  
{ ...  
}  
Time::~~Time() //析构体外实现  
{ ...  
}
```



目录

- 对象的初始化方法
- 构造函数的基本概念
- 构造函数的使用
- 析构函数的基本概念
- 析构函数的使用



补.5 析构函数的使用

- 析构函数的使用
 - 对象撤销时被自动调用，用户不能显式调用
 - 若不指定析构函数，则系统缺省生成一个无参空体析构函数
 - 若用户定义了析构函数，则缺省析构函数不再存在
 - 在数据成员没有动态内存申请需求的情况下，一般不需要定义析构函数(动态内存申请为后续课程内容，此处不再展开)



总结

- 对象的初始化方法
- 构造函数的基本概念（熟练掌握）
- 构造函数的使用（熟练掌握）
- 析构函数的基本概念（熟练掌握）
- 析构函数的使用