

# 第2讲 类和对象（中）

参考教材的第14章

黄永峰

62792710

yfhuang@tsinghua.edu.cn

2023.2.28

# 封装性源于中国文化



昵图网 [www.nipic.com](http://www.nipic.com)

联合汇赢

By:melody9988 No.20120621092300899000

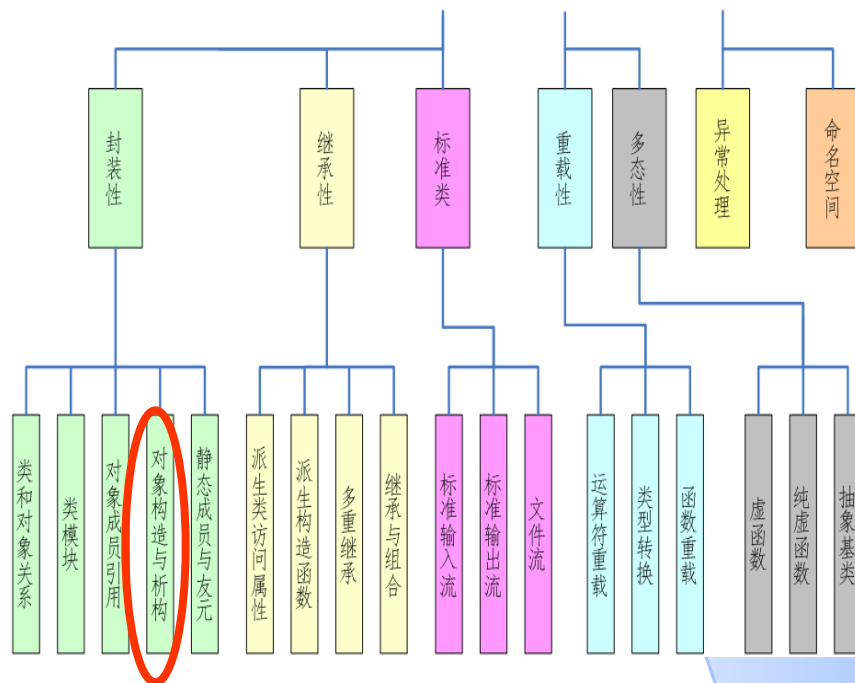
# 本讲主要内容

C++的OOP程序 = 对象 + 消息 + 工具

## 2.1 构造函数

## 2.2 析构函数

## 2.3 对象的拷贝与赋值



思考：如何从构造函数角度来理解类/对象的封装性？

# 问题的引入

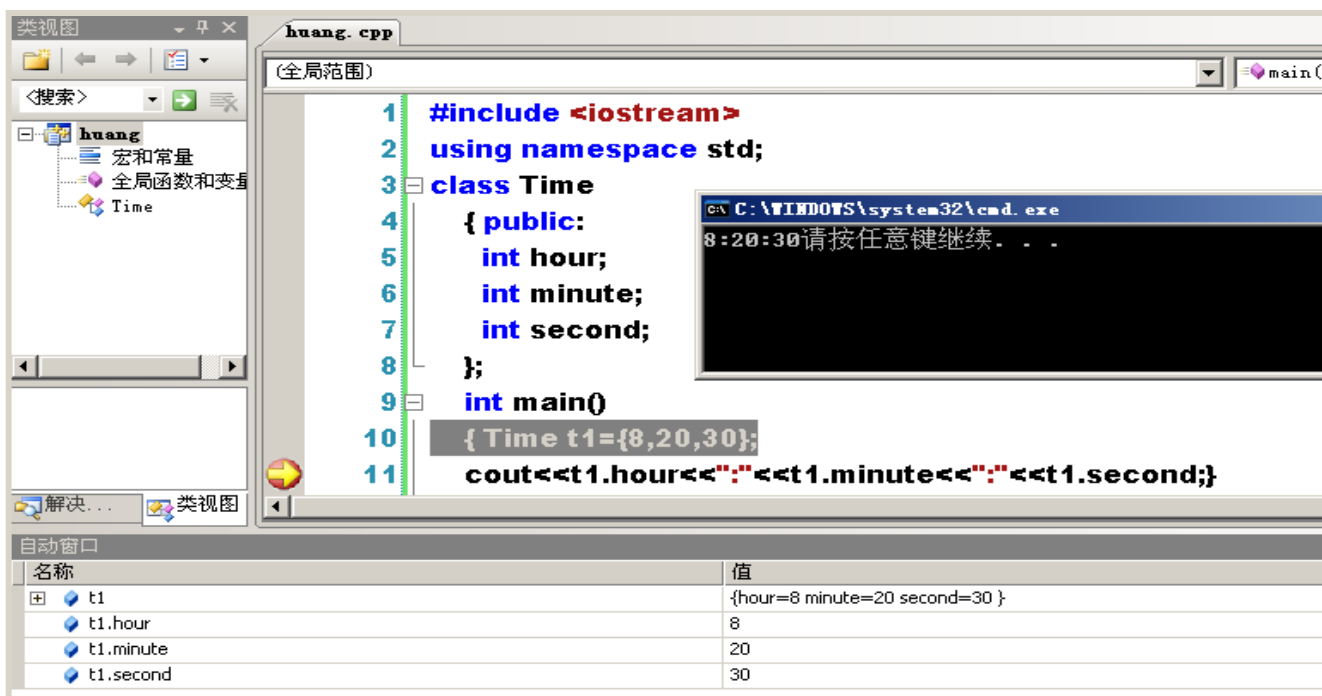
- 变量的初始化：在定义变量时赋给其一个初始值。int a=88;
- 那么，在用某类定义一个对象时，如何对该对象进行初始化呢？

```
struct student  
{  
    long int num;  
    char  name[20];  
    char  sex;  
    char  addr[20];  
} a = {89031, "Li Lin", 'M', "123 Beijing Road"};
```



## 2.1 构造函数

- ❑ 对象初始化：对象初始化是在对象定义时，给其数据成员初始值；
- ❑ (1) 类数据成员是否可在声明类时初始化？为什么？
- ❑ (2) 是否可以在定义对象时采用赋值语句对数据成员进行初始化？（如果类中所有成员都是public，则可以。）
- ❑ (3) 类中的数据成员都是private，是否可以？如何初始化？



```

1  #include <iostream>
2  using namespace std;
3  class Time
4  { public:
5      int hour;
6      int minute;
7      int second;
8  };
9  int main()
10 { Time t1={8,20,30};
11   cout<<t1.hour<<":"<<t1.minute<<":"<<t1.second;<<endl;

```

名称	值
t1	{hour=8 minute=20 second=30 }
t1.hour	8
t1.minute	20
t1.second	30

❑ 定义成员函数来初始化！

❑ 是否可以定义一个在对象构建时自动调用的成员函数来初始化？

# 成员函数对对象初始化

```

1. class Date
2. {
3.     int d, m, y;
4. public:
5.     void init(int dd, int mm, int yy);
6.     ...
7. };
8. void Date::init(int dd, int mm, int yy)
9. {
10.    d = dd;
11.    m = mm;
12.    y = yy;
13. }
14. void func()
15. {
16.    Date A, B;
17.    A.init(1, 1, 1900);    //函数调用
18.    B.init(30, 12, 2010); //函数调用
19. }

```

C++编译器在类代码中隐藏了下列灰色代码

```

class Date
{
    int d, m, y;
public:
    void init(Date* this, int dd, int mm, int yy);
    ...
};
void Date::init(Date* this, int dd, int mm, int yy)
{
    this->d = dd; //成员变量访问被替换成通过this指针参数访问
    this->m = mm;
    this->y = yy;
}
void func()
{
    Date A, B;
    A.init(&A, 1, 1, 1900); //函数调用最终还是同一个版本
    B.init(&B, 30, 12, 2010); //函数调用最终还是同一个版本
}

```

C++由此实现了同一类的对象共享一组成员函数代码

# 构造函数对对象初始化

- 构造函数(constructor): 特殊成员函数, 专门用来处理对象的初始化;
- 特点: (1) 不要由用户调用, 建立对象时系统自动执行;  
(2) 构造函数名字必须与类名同名, 不能由用户任意命名;  
(3) 没有函数类型, 不返回任何值。

## □构造函数类型

- 默认构造函数:空构造函数; 不带参数或默认参数构造函数;
- 用户自定义构造函数
  - 不带参数构造函数
  - 带参数构造函数:
    - (1) 函数体传递参数;
    - (2) 通过参数初始化表来传递;
    - (3) 指定默认参数。



# 默认（缺省）构造函数

- (1) 如果没有为类提供任何构造函数，编译器会自动生成一个默认无参数的构造函数，且函数体为空，不进行任何工作；
- (2) 一个类中构造函数没有参数，或者构造函数的所有参数都有默认值，也可称其为缺省构造函数。
- (3) 一个类中，只能有一个缺省构造函数。

```
class Date
{
    int d, m, y;
public:
    Date();        //缺省构造函数
    ~Date();       //缺省析构函数
    ...
};

void func()
{
    Date A;
}
```

尽管我们没有定义，但实际编译器已经为我们提供了缺省的构造函数和析构函数。

隐藏调用A.Date();

隐藏调用A.~Date();

构造函数可以防止程序员在使用对象之前忘记初始化，又可以将这些过程隐藏起来，使程序更加优美。



# 不带参数的构造函数

```
class Date {
private:
    int d, m, y;
public:
    Date(); // 函数声明中显式地加入无参数构造函数
    ...
};
// 函数定义中给出自定义的无参数构造函数
void Date::Date()
{
    // 将每一个成员初始化为0
    d = 0; m = 0; y = 0;
}

void func()
{
    Date A;
    ...
    Date * pD =
        new Date;
    ...
}
```

在一般的对象定义和动态分配时，编译器都会自动调用无参数构造函数。

# 构造函数的特点

1. 构造函数**必须声明为public成员**；
2. 构造函数的**调用时间**为对象进入作用域、对象的生命周期开始时刻；
3. 构造函数**没有返回值**，在声明和定义时不需要注明void返回类型；
4. 构造函数只能由编译器根据对象产生以隐藏方式调用，**不能由程序代码直接调用**；
5. 构造函数体可以包含任何内容，包括cin、cout，但**不提倡在构造函数中加入与初始化无关的操作**；
6. 如果用户没有定义构造函数，编译器会自动提供**缺省的无参数构造函数**，其函数体为空（什么都不做）。

■ 问题：为什么某型号工业产品一模一样？



# 带参数的构造函数

- 不带参数构造函数在函数体中对对象数据成员初始化，使得同类所有对象的初始值相同；但：**如何使得同类不同对象的初始值不同呢？**
  - 带参数构造函数：构建不同对象时，从外部将不同值传递给构造函数，使得不同对象的初始化不同
  - **一般格式：构造函数名(类型1 形参1，类型2 形参2，...)**
- 
- **用户不能调用构造函数，那如何给出实参呢？**
  - **实参是在构建对象时给出，一般格式为：**
  - **类名 对象名(实参1，实参2，...);**
  - **例如：Date t1(8,8,8);**



# 构造函数的重载

- 一个类可定义多个构造函数，以便对类对象提供不同的初始化方法。但对象构建时只能执行其一
- 类所有构造函数名字必须相同，而参数个数或参数类型必须有所不同，即所谓的**构造函数重载**

<搜索>

test2

- 宏和常量
- 全局函数和变量
- Box

---

- Box(int h, int w, int len)
- Box(void)
- volume(void)
- height
- length
- width

```

4  {public:
5      Box();                      //声明一个无参的构造函数
6      Box(int h,int w,int len):height(h),width(w),length(len){ }
7      //声明一个有参的构造函数，用参数的初始化表对数据成员初始化
8      int volume();
9  private:
10     int height;
11     int width;
12     int length;};
13
14  Box::Box()
15  {height=10;
16   width=10;
17   length=10;}
18
19  int Box::volume()
20  {return(height*width*length);}
21
22  int main()
23  {
24      Box box1;                  //建立对象box1,不指定实参
25      cout<<"The volume of box1 is"<<box1.volume()<<endl;
26      Box box2(15,30,25);        //建立对象box2,指定3个实参
27      cout<<"The volume of box2 is"<<box2.volume()<<endl;
28      return 0; }

```

C:\WINDOWS\system32\cmd.exe

The volume of box1 is1000

The volume of box2 is11250

请按任意键继续. . .

# 构造函数的重载

```

class Date
{
    int d, m, y;
public:
    Date(int dd, int mm, int yy); //初始化d,m,y
    Date(int dd, int mm);         //初始化d,m,用当天的y
    Date(int dd);                 //初始化d,用当天的m,y
    Date();                      //无参数构造函数,用当天日期
    Date(const char * strDate)   //用字符串日期初始化
    ...
};
  
```

```

main()
{
    Date today(4);
    Date July4(4,7);
    Date DDay(6,6,1944);
    Date now;
    Date guy("1983-11-5");
    ...
}
  
```

如何让编译器调用  
有参构造函数？

编译器根据实参的类  
型和个数决定调用哪  
一个的构造函数

# 默认参数的构造函数

```

1.  class Date
2.  {
3.      int d, m, y;
4.  public:
5.      Date(int dd, int mm, int yy); //初始化d,m,y
6.      Date(int dd, int mm);         //初始化d,m,用当天的y
7.      Date(int dd);                 //初始化d,用当天的m,y
8.      Date();                       //无参数构造函数
9.      ...
10. };
11.
12. int main()
13. {
14.     Date today(4);
15.     Date July4(4,7);
16.     Date DDay(6,6,1944);
17.     Date now;
18.
19.     ...
20. }

```

利用C++对缺省参数的支持，可以将多个版本的构造函数合成一个：

```
Date(int dd=0, int mm=0, int yy=0);
```

注意：默认构造函数与默认参数的构造函数的区别与联系



# 默认参数的构造函数

- ① 在声明构造函数时指定默认值，而不能只在定义构造函数时指定默认值；
- ② 如果构造函数的全部参数都指定了默认值，则在定义对象时可以给一个或几个实参，也可以不给出实参；
- ③ 在一个类中定义了全部是默认参数的构造函数后，**不能再定义重载构造函数，为什么？**；
- ④ 好处：即使在调用构造函数时没有提供实参值，不仅不会出错，而且还确保按照默认的参数值对对象进行初始化。

# 用参数初始化表来简化构造函数

```
class Date
{
    int d, m, y;
public:
    Date(int dd, int mm, int yy);
    ...
};

Date::Date(int dd, int mm, int yy)
    : d(dd), m(mm), y(yy)
{
}
```

关于参数初始化表：

- 初始化表用于初始化成员变量；
- 要初始化的成员次序和个数不限；
- 括号内的初始值可以是常量、函数形参或其它合法表达式；
- 一般用于简单类型成员变量的初始化。

格式为：成员变量1(初始值1)， 成员变量2(初始值2)，...

```

1  #include <iostream>
2  using namespace std;
3  class A
4  {
5  public:
6      A(int i,int j)
7      {a1=i;
8       a2=j;}
9  void print()
10 {cout<<a1<<" , "<<a2<<endl;}
11 private:
12     int a1,a2; };
13
14 class B
15 {public:
16     B(int i, int j, int k): a(i,j), b(k){}
17 void print();
18 private:
19     A a;
20     int b;};
21
22 void B:: print()
23 { a.print();
24  cout<<b<<endl;}
25 void main()
26 { B b (6,7,8);
27  b.print();}

```



```

C:\WINDOWS\system32\cmd.exe
6 , 7
8
请按任意键继续. . .

```

□ 用参数初始化表可解决某个类中对象成员的初始化问题 17

# 分析题中构造函数及重载问题？

类视图

huang20110530

- 宏和常量
- 全局函数和变量
- Box

volume(void)

height

length

width

◆ 此时调用哪个构造函数？

```

1  #include <iostream>
2  using namespace std;
3  class Box
4  {public:
5      //Box(int h,int w ,int len):height(h),width(w),length(len){}
6      int volume();
7  private:
8      int height;
9      int width;
10     int length;
11 }
12
13 int Box::volume()
14 {return(height*width*length);
15 }
16
17 int main()
18 {
19     Box box1;
20     cout<<"The volume of box2 is "<<box1.volume()<<endl;
21     // Box box2(15,30,25);
22     // cout<<"The volume of box2 is "<<box2.volume()<<endl;
23     return 0;
24 }

```

C:\WINDOWS\system32\cmd.exe

The volume of box2 is 1168231104  
请按任意键继续. . .

```

1 #include <iostream>
2 using namespace std;
3 class Box
4 {public:
5     Box(int h,int w,int len):height(h),width(w),length(len){}
6     int volume();
7 private:
8     int height;
9     int width;
10    int length;
11 };
12
13 int Box::volume()
14 {return(height*width*length);
15 }
16
17 int main()
18 {
19     Box box1;
20     cout<<"The volume of box2 is "<<box1.volume()<<endl;
21     Box box2(15,30,25);
22     cout<<"The volume of box2 is "<<box2.volume()<<endl;
23     return 0;
24 }
    
```



- 为什么会错？怎样避免这样错误？
- 未定义构造函数时，系统才产生默认构造函数

输出

显示输出来源(S): 生成

```

1>----- 已启动全部重新生成: 项目: huang20110530, 配置: Debug Win32 -----
1>正在删除项目“huang20110530”(配置“Debug|Win32”)的中间文件和输出文件
1>正在编译...
1>huang.cpp
1>c:\documents and settings\administrator\my documents\visual studio 2008\projects\huang20110530\huang20110530\huang.cpp(19) : error C2512: “Box”: 没有合适的默认构造函数可用
1>生成日志保存在“file:///c:/Documents and Settings/Administrator/My Documents/Visual Studio 2008/Projects/huang20110530/huang20110530/Debug/BuildLog.htm”
1>huang20110530 - 1 个错误, 0 个警告
===== 全部重新生成: 成功 0 个, 失败 1 个, 跳过 0 个 =====
    
```

```

0  #include <iostream>
7  using namespace std;
8  class Box
9  {public:
10 Box(int h=10,int w=10,int len=10):height(h),width(w),length(len);
11 int volume( );
12 private:
13 int height;
14 int width;
15 int length;};
16
17 int Box::volume( )
18 {return(height*width*length);}
19
20 int main( )
21 {
22 Box box1;           //没有给实参
23 cout<<"The volume of box1 is"<<box1.volume()<<endl;
24 Box box2(15);       //只给定一个实参
25 cout<<"The volume of box2 is"<<box2.volume()<<endl;
26 Box box3(15,30);    //只给定2个实参
27 cout<<"The volume of box3 is "<<box3.volume()<<endl;
28 Box box4(15,30,20); //给定3个实参
29 cout<<"The volume of box4 is "<<box4.volume()<<endl;
30 return 0;}

```

在定义函数的时候可以不用给出实参

图 1

名称	值	类型
box1	{ height=10 width=10 length=10	Box
height	10	int
length	10	int
width	10	int
box2	{ height=15 width=10 length=10	Box
height	15	int
length	10	int
width	10	int

自动窗口 局部变量 线程 模块 监视 1



## 2.2 析构函数

□ 析构函数(destructor): 特殊成员函数, 作用与构造函数相反, 名字是类名前加一个“~”符号。当对象生命期结束时, 程序就会自动执行析构函数。

```
class Date
{
    int d, m, y;
public:
    Date();        //缺省构造函数
    ~Date();       //缺省析构函数
    ...
};
```

尽管我们没有定义, 但其实编译器已经为我们提供了缺省的构造函数和析构函数。

```
int main()
{
    Date A;
}
```

隐藏调用 `A.Date()` ;

隐藏调用 `A.~Date()` ;

用于释放资源, 或其他操作

# 构造函数和析构函数的执行顺序

```

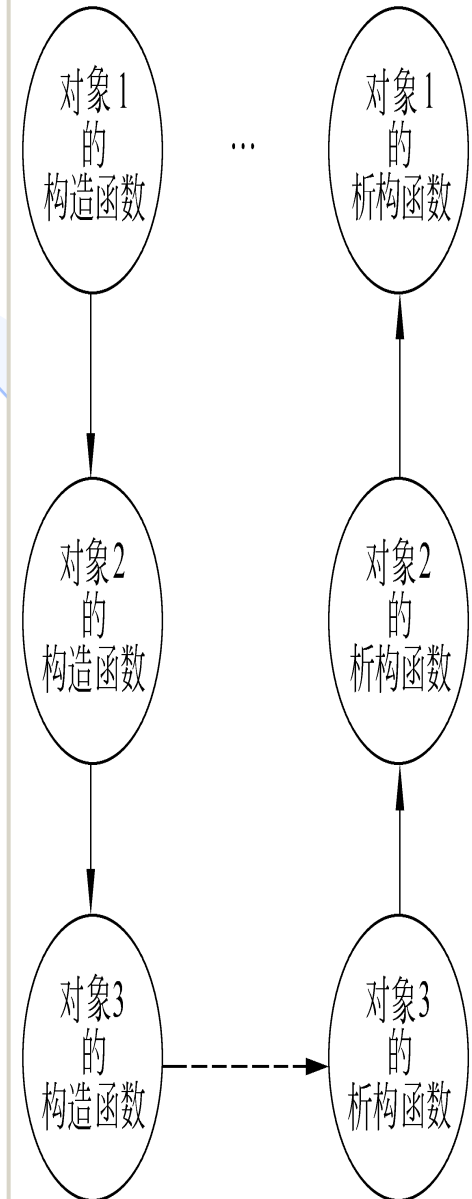
1 #include<string>
2 #include<iostream>
3 using namespace std;
4 class Student //声明Student类
5 {public:
6 Student(int n, char *nam, char s) //定义构造函数
7 { num=n;
8   name=nam;
9   sex=s;
10  cout<<"Constructor called."<<endl; } //输出有关信息
11
12 ~Student() //定义析构函数
13 {cout<<"Destructor called."<<endl;} //输出有关信息
14
15 void display() //定义成员函数
16 {cout<<"num: "<<num<<endl;
17  cout<<"name:"<<name<<endl;
18  cout<<"sex:"<<sex<<endl<<endl; }
19 private:
20   int num;
21   char *name;
22   char sex;};
23
24 int main()
25 {Student stud1(10010,"Wang_li",'f'); //建立对象stud1
26  stud1.display(); //输出学生1的数据
27  Student stud2(10011,"Zhang_fun",'m'); //定义对象stud2
28  stud2.display(); //输出学生2的数据
29  return 0;}
  
```

```

C:\WINDOWS\system32\cmd.exe
Constructor called.
num: 10010
name:Wang_li
sex:f

Constructor called.
num: 10011
name:Zhang_fun
sex:m

Destructor called.
Destructor called.
请按任意键继续. . .
  
```



# 析构函数的特点

1. 析构函数**必须声明为public成员**；
2. 析构函数的**调用时间**为对象离开作用域或被删除等对象的生命周期结束时刻；
3. 析构函数**没有返回值和参数**，在声明和定义时不需要注明void返回类型，**一个类只有一个析构函数**；
4. 析构函数只能由编译器根据对象回收以隐藏方式调用，**不能由程序代码直接调用**；
5. 析构函数体可以包含任何内容，包括cin、cout，**建议析构函数进行对象回收时的内存清理等结束工作**；
6. 如果用户没有定义析构函数，编译器会自动提供**缺省的析构函数**，其函数体为空（什么都不做）。

问题：析构函数能否重载？为啥？

## 2.3 对象的赋值和复制（拷贝）

1.对象赋值一般形式为：对象名1 = 对象名2

```

1  #include <iostream>
2  using namespace std;
3  class Box
4  {public:
5      Box(int=10,int=10,int=10);
6      int volume();
7  private:
8      int height;
9      int width;
10     int length; };
11
12 Box::Box(int h,int w,int len)
13 {height=h;
14  width=w;
15  length=len; }
16
17 int Box::volume()
18 {return(height*width*length); }
19
20 int main()
21 {Box box1(15,30,25),box2;
22  Box *pt=new Box;
23  cout<<"The volume of box1 is "<<box1.volume()<<endl;
24  box2=box1;
25  cout<<"The volume of box2 is "<<box2.volume()<<endl;
26  *pt=box1;
27  cout<<"The volume of *pt is "<<(*pt).volume()<<endl;
28  delete pt;
29  return 0; }

```

自动窗口		
名称	值	类型
*pt	{height=15 width=30 length=25 }	Box
height	15	int
width	30	int
length	25	int
box1	{height=15 width=30 length=25 }	Box
height	15	int
width	30	int
length	25	int

自动窗口		
名称	值	类型
pt	0x003861a8 {height=-17891602 width=-17891602 length=-17891602 }	Box *
height	-17891602	int
width	-17891602	int
length	-17891602	int

```

C:\WINDOWS\system32\cmd.exe
The volume of box1 is 11250
The volume of box2 is 11250
The volume of *pt is 11250
请按任意键继续. . .

```

# 对象的拷贝及其构造函数

- 对象拷贝:用已有对象快速地拷贝出多个相同对象。

格式1:类名 对象2(对象1); 如: Box box2(box1);

格式2:类名 对象名2 = 对象名1;

- 在建立对象2时调用一个特殊的构造函数—拷贝构造函数 (copy constructor).系统缺省的拷贝构造函数的举例:

```
Box::Box(const Box& b)
{height=b.height;
 width=b.width;
 length=b.length;}
```

- 拷贝构造函数只有一个参数, 参数是本类的对象

问题:(1)对象的赋值和拷贝有区别吗?

(2)普通构造函数和拷贝构造函数是如何重载?

```

1 #include <iostream>
2 using namespace std;
3 class Box
4 {public:
5     Box(int=10,int=10,int=10);
6     int volume();
7 private:
8     int height;
9     int width;
10    int length;};
11
12 Box::Box(int h,int w,int len)
13 {height=h;
14  width=w;
15  length=len;}
16
17 int Box::volume()
18 {return(height*width*length); } //返回体积
19
20 int main()
21 {Box box1(15,30,25); //定义box1
22  cout<<"The volume of box1 is"<<box1.volume()<<endl;
23  Box box2=box1,box3=box2; //按box1来复制box2,box3
24  cout<<"The volume of box2 is"<<box2.volume()<<endl;
25  cout<<"The volume of box3 is "<<box3.volume()<<endl;
26 }

```

//声明有默认参数的构造函数

监视 1

名称	值	类型
height	15	int
length	25	int
width	30	int
box2	{ height=15 width=30 length=25 }	Box
height	15	int
length	25	int
width	30	int
box3	{ height=15 width=30 length=25 }	Box
height	15	int
length	25	int
width	30	int

自动窗口 局部变量 线程 模块 监视 1



# 对象的隐式复制

- 在对象定义时是显示复制。在对象作为函数参数或返回值时，会发生隐式复制，且会调用拷贝构造函数

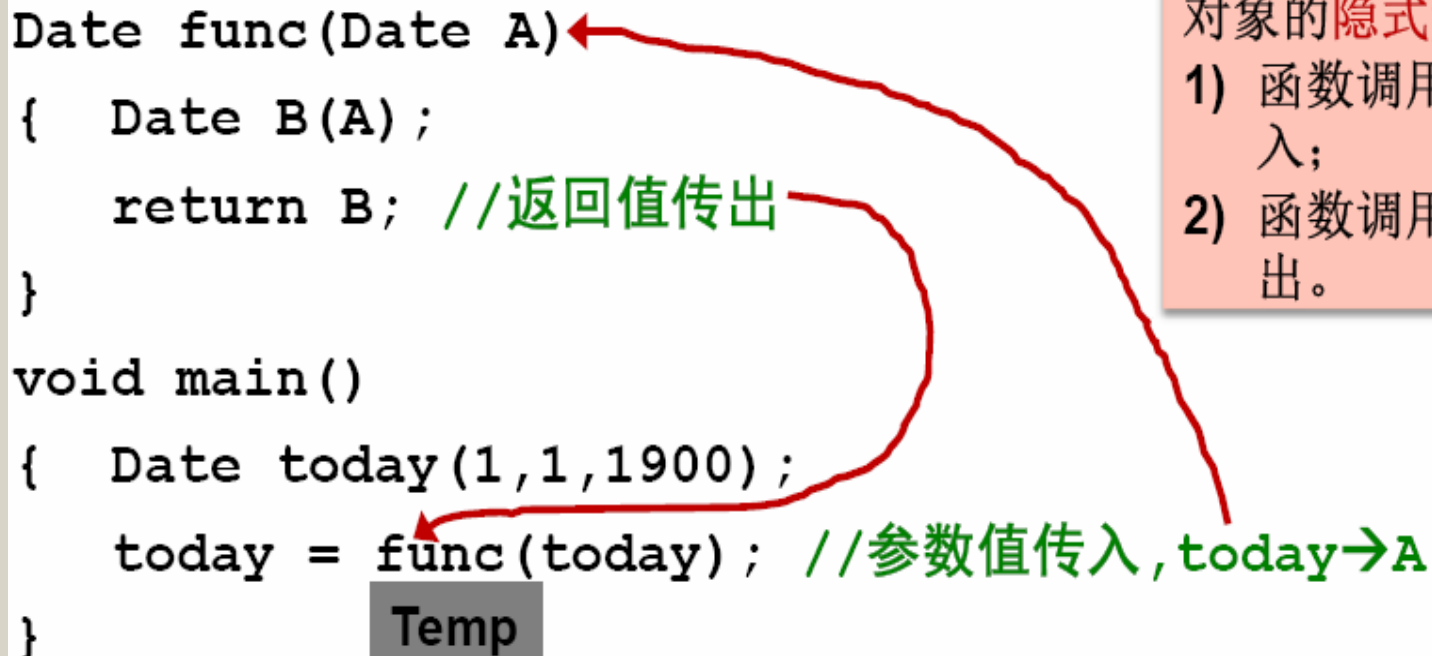
## 隐式复制

```

Date func(Date A)
{
    Date B(A);
    return B; //返回值传出
}

void main()
{
    Date today(1,1,1900);
    today = func(today); //参数值传入, today→A
}
    
```

Temp



对象的隐式复制发生在：

- 1) 函数调用时参数值传入；
- 2) 函数调用时返回值传出。

□ `today=func(today);` → `today=temp;` //是拷贝还是赋值？

```

1. class Date
2. {
3.     int d, m, y;
4. public:
5.     Date(int dd=0, int mm=0, int yy=0);
6.     Date(Date &D); //拷贝构造函数
7.     ~Date();
8.     ...
9. };
10. Date::Date(int dd, int mm, int yy)
11. : d(dd), m(mm), y(yy)
12. { cout << "Constructor called! Address=0x" <<
13.     hex << setw(8) << setfill('0') << this << endl;
14. }
15. Date::Date(Date &D)
16. { d = D.d; m = D.m; y = D.y;
17.     cout << "Copy constructor called! Address=0x" <<
18.         hex << setw(8) << setfill('0') << this << endl;
19. }
20. Date::~~Date()
21. { cout << "Destructor called! Address=0x" <<
22.     hex << setw(8) << setfill('0') << this << endl;
23. }

```

```

Date func(Date A)
{
    return Date(A);
}

void main()
{
    Date today;
    today = func(today);
}

```

Date(A)  
临时对象  
被保留到  
这句话之  
后再释放

```

Constructor called! Address=0x0018FF14
Copy constructor called! Address=0x0018FE08
Copy constructor called! Address=0x0018FE40
Destructor called! Address=0x0018FE08
Destructor called! Address=0x0018FE40
Destructor called! Address=0x0018FF14

```

```

today
A ← today
Date(A) ← A
~A
today = Date(A)
~Date(A)
~today

```

# 小结：对象的生命周期

- 类的对象和简单变量一样，都有诞生和结束的时刻，从诞生到结束的这段时间就是它的生存期。在生存期内，对象将保持其内存和状态，直到被改写为止。

```
class Date
```

```
{
```

```
    ...
```

```
};
```

```
Date globalDay;
```

```
void main()
```

```
{
```

```
    Date today;
```

```
    Date * pDate = new Date;
```

```
    ...
```

```
}
```

静态内存分配的对象，生存周期从定义位置到其作用域结束为止

globalDay

动态内存分配的对象，生存周期从new运算符开始，到同一指针被执行delete运算符为止

today

## 第2次实验练习

□要求：必做2道题，选做题2道；提交源程序和运行结果，按交时间第4周末之前

- 1、见教材第14章课后练习第7题。建立一个对象数组，内置5个学生的数据（学号，成绩）。用指针指向数组首元素，输出第1、3、5个学生的数据。
- 2、五子棋程序设计中，在第一次作业的申明2个类ChessBoard（棋盘）类和playerU类的基础上，急需在各个类中添加相应的初始化函数。建议：在ChessBoard（棋盘）类中能初始化棋盘的大小等。在playerU类中，能初始化玩家的姓名等。每个初始化函数能初始化的参数可以在此之上自由发挥。另外，为每个类设计相应的析构函数。

## 第2次实验练习选做题

1、在“公司人事管理系统”前次基础上，增加无参构造和有参构造函数，实现对4个员工对象的初始化（初始化值自己确定）。要求：（1）程序设计中要体现构造函数的重载；（2）4个对象初始值不同。（3）使用析构函数，在程序退出时显示“欢迎使用，再见”；（4）分析这些对象释放时析构函数执行顺序。

□ 参考和阅读教程第14章代码，并在此基础上改进和优化。

2. 完成如下类申明中构造函数、析构函数和成员函数set（）、print（）的定义。并构造2个不同的对象。

```
class Strings {
public:
    Strings(char *s);
    ~Strings();
    void Print();
    void Set(char *s);
private:
    int length;
    char *str;};
```