中山大學
SUN YAT-SEN UNIVERSITY

程序设计II实验

电子与信息工程学院（微电子学院）

庞志勇 高级实验师 助教：焦涵 、桂海田 博士研究生

PPT大纲

- 上次课实验总结
- 本次课的实验介绍

构造函数与析构函数

1. 构造函数的作用

构造函数主要用来在创建对象时完成对对象属性的一些初始化等操作, 当创建对象时, 对象会自动调用它的构造函数。一般来说, 构造函数有以下三个方面的作用:

- ✓给创建的对象建立一个标识符;
- ✓为对象数据成员开辟内存空间;
- ✓ 完成对象数据成员的初始化。

2. 默认构造函数

当用户没有显式的去定义构造函数时, 编译器会为类生成一个默认的构造函数, 称为"默认构造函数", 默认构造函数不能完成对象数据成员的初始化, 只能给对象创建一标识符, 并为对象中的数据成员开辟一定的内存空间。

3. 构造函数的特点

无论是用户自定义的构造函数还是默认构造函数都主要有以下特点:

①. 在对象被创建时自动执行;
②. 构造函数的函数名与类名相同;
③. 没有返回值类型、也没有返回值;
④. 构造函数不能被显式调用。

构造函数的作用是初始化，替代cin cout繁杂的输入输出操作赋值

```
#include <iostream>
#include <string>
using namespace std;

class Account
{
        string name; //姓名
        string numb; //账户号码
        unsigned int saving; //存款
 public:
       Account ( string name0, string numb0,
                    unsigned int saving0  )
     {
            name = name0;
            numb = numb0;
            saving = saving0;
      }
      void show()
     {
            cout << name << endl;
            cout << numb << endl;
            cout << saving << endl;
      }
            void Deposit(unsigned int amount) //存钱
            {
                    saving = saving + amount;
              }
              void GetDeposit(unsigned int amount) //取钱
              {
                    if (amount>saving)  //存款不够
                    {
                       saving = 0; //取出所有钱，剩余存款为0
                      cout << "you don't have enough money!" << endl;
                    }
                          else saving = saving - amount;
              }
       };
       int main()
       {
                    Account A("Karen", “123", 100);
                    A.show();
                    A.Deposit(1000);
                    A.show();
                    A.GetDeposit(3000);
                    A.show();
       }
```

多读多写

C++Primer第5版7.6节练习（类的静态成员）

练习7.56：什么是类的静态成员？它有何优点？静态成员与普通成员有何区别?

答：类的静态成员和类本身相关，不与类的各个对象保持关联。

它的优点是：对于所有对象共享的一个成员，不必要为每个对象都绑定该成员，因为它属于类，减少内存的占用。

静态成员属于类，而普通成员属于类的对象；静态成员可以作为默认实参，而普通成员不能作为默认实参。

多读多写

编写你自己的银行账户Account类。

```
#include<iostream>
#include<string>
using namespace std;

class Account
{
public:
	Account(string ow = "") :owner(ow), amount(0) {}
	Account(string ow, double am) :owner(ow),
amount(am) {}
	void clculate() { amount += amount * interestRate; }
	static double rate() { return interestRate; }
	static void reSetRate(double);
private:
		string owner;
		double amount;
		static double interestRate;
};
```

```
double Account::interestRate = 0.05;//静态成员不能在类
定义里边初始化，只能在class body外初始化。
void Account::reSetRate(double newRate)
{
		interestRate = newRate;
}

int main()
{
		Account a("zhangsan",10000);
		cout << Account::rate() << endl;
		cout << a.rate() << endl;
		Account::reSetRate(0.08);
		cout << Account::rate() << endl;
		cout << a.rate() << endl;
		return 0;
}
```

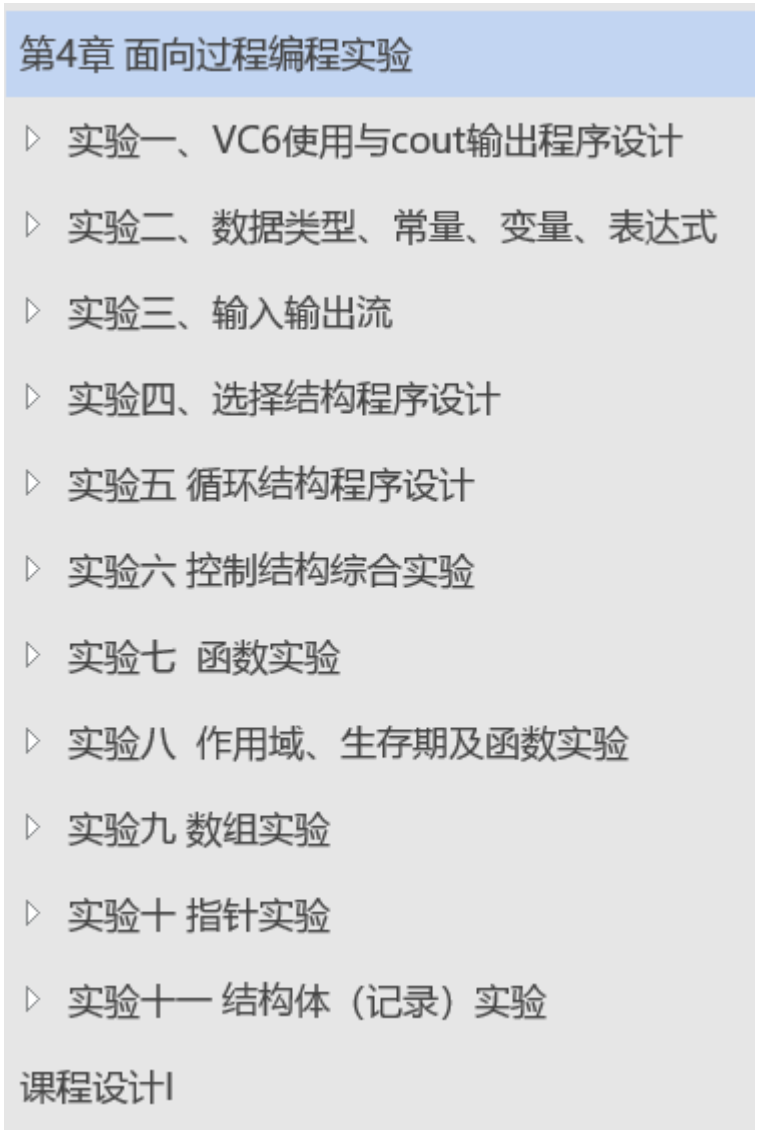
多读多写

C++ 银行账户（静态成员与友元函数）

```
#include <string>
using namespace std;//银行账户类
class Account {
public:
    Account(string accno,string name,float balance);
    ~Account();
   void Deposit(float amount); //存款
   void Withdraw(float amount) ;//取款
   float GetBalance(); //获取账户余额
    void show()://显示账户所有基本信息
    static int GetCount ) //获取账户数
    static float GetInterestRate();//获取利率
private:
     static int count;                  //银行账户数量
     static float InterestRate;         //利率
     string _accno, _accname;           //账户与户名
     float _balancer;                   //账户余额
}
```

要求如下：实现该银行账户类为账户类Account增加一个友元函数，实现账户结息，要求输出结息后的余额（结息余额=账户余额+账户余额*利率）。友元函数声明形式为friend void Update(Account& a);在main函数中，定义一个Account类型的指针数组，让每个指针指向动态分配的Account对象，并调用成员函数测试存款、取款、显示等函数，再调用友元函数测试进行结息。大家可以根据实际需求在类内添加其他方法，也可以修改成员函数的参数设置

要做自编教材哪些实验？



实验内容

- 实验内容根据理论课内容进行调整
- 每次课实验内容可以根据自己学习情况动态调整
- 课程设计可以自选其他内容

实验二 函数重载与运算符重载

一. 实验目的

1. 掌握运算符重载的定义及实现。
2. 掌握友元函数的定义方法。
3. 掌握用友元函数重载运算符的方法。
4. 掌握用类成员函数重载运算符的方法。

二. 实验原理

在实际程序设计中，有时候我们需要实现几个功能类似的函数，只是有些细节不同。例如希望在显示器上输出显示一个变量的值，这个变量有多种类型，可以是 int、float、char、bool 等。在C语言中，程序员往往需要分别设计出三个不同名的函数。但在C++中，这完全没有必要。C++ 允许多个函数拥有相同的名字，只要它们的参数列表不同就可以，这就是函数的重载。借助重载，一个函数名可以有多种用途。参数列表又叫参数签名，包括参数的类型、参数的个数和参数的顺序，只要有一个不同就叫做参数列表不同。

实验二 函数重载与运算符重载

二. 实验原理

函数重载（Function Overloading）是指在同一作用域内（同一个类、同一个命名空间等），有一组具有相同函数名，不同参数列表的函数，这组函数被称为重载函数。重载函数通常用来命名一组功能相似的函数，这样做减少了函数名的数量，避免了名字空间的污染，对于程序的可读性有很大的好处。函数重载是一种静态多态。函数重载的规则：

1）函数名称必须相同。

2）参数列表必须不同（个数不同、类型不同、参数排列顺序不同等）。

3）函数的返回类型可以相同也可以不相同。

4）仅仅返回类型不同不足以成为函数的重载。

5）使用函数重载时，对哪一函数进行调用，千万要注意不能引起歧义。例如：

```
void rotate（int&，int&）；
void rotate（int，int）；
```

产生值传递和引用传递表意含混不清的情况。

6）使用函数重载时，注意C++在函数调用时有时候会进行类型自动转换。

实验二 函数重载与运算符重载

二. 实验原理

- ✓ 运算符重载（Operator Overloading），也叫操作符重载，是C++的重要组成部分，它可以让程序更加的简单易懂。
- ✓ 运算符重载，就是对已有的运算符重新进行定义，赋予其另一种功能，以适应不同的数据类型。
- ✓ C++预定义中的运算符的操作对象只局限于基本的内置数据类型，但是对于我们自定义的类型（类）是没有办法操作的。但是大多时候我们需要对我们定义的类型进行类似的运算，这个时候就需要我们对这么运算符进行重新定义，赋予其新的功能，以满足自身的需求。
- ✓ 实际上，我们已经在不知不觉中使用了运算符重载。例如，+号可以对不同类型（int、float 等）的数据进行加法操作；<<既是位移运算符，又可以配合 cout 向控制台输出数据。C++ 本身已经对这些运算符进行了重载。
- ✓ C++ 也允许程序员自己重载运算符。
- ✓ 运算符重载的实质就是函数重载或函数多态。运算符重载是一种形式的C++多态。
- ✓ 运算符重载其实就是定义一个函数，在函数体内实现想要的功能，当用到该运算符时，编译器会自动调用这个函数。也就是说，运算符重载是通过函数实现的，它本质上是函数重载。
- ✓ 重载的运算符是带有特殊名称的函数，函数名是由关键字 operator 和其后要重载的运算符符号构成的。与其他函数一样，重载运算符有一个返回类型和一个参数列表。

实验二 函数重载与运算符重载

二. 实验原理

运算符重载的格式为:

<返回类型说明符> operator <运算符符号>(<参数表>)

{

<函数体>

}

operator是关键字，专门用于定义重载运算符的函数。我们可以将operator运算符名称这一部分看做函数名，运算符重载函数除了函数名有特定的格式，其它地方和普通函数并没有区别。当重载运算符不是后置“++”或“--”时，参数的个数比原运算符的操作数个数少一个，因为类的对象调用运算符重载成员函数时，自己的数据可以直接访问，不需要在参数表中传递，所以参数表中就不必列出该对象本身了。

实验二 函数重载与运算符重载

二. 实验原理

运算符重载的规则:

- ✓ 为了防止用户对标准类型进行运算符重载，C++规定重载后的运算符的操作对象必须至少有一个是用户定义的类型。
- ✓ 使用运算符不能违法运算符原来的句法规则。如不能将% 重载为一个操作数,

  例如: int index;

  %index; 这种是不被允许的。
- ✓ 不能修改运算符原先的优先级。
- ✓ 不能创建一个新的运算符，例如不能定义operator** （…）来表示求幂
- ✓ 大多数运算符可以通过成员函数和非成员函数进行重载但是下面这四种运算符只能通过成员函数进行重载:

  = 赋值运算符， （）函数调用运算符，[ ]下标运算符， ->通过指针访问类成员的运算符。
- ✓ 除了上述的规则，其实我们还应该注意在重载运算符的时候遵守一些明智的规则：例如：不要将+运算符重载为交换两个对象的值。
- ✓ 重载运算符的两种形式：成员函数形式和非成员函数（友元函数）形式。这两种形式都可访问类中的私有成员。

实验二 函数重载与运算符重载

二. 实验原理

https://www.runoob.com/cplusplus/cpp-overloading.html

运算符重载实例

下面提供了各种运算符重载的实例，帮助您更好地理解重载的概念。

序号	运算符和实例
1	一元运算符重载
2	二元运算符重载
3	关系运算符重载
4	输入/输出运算符重载
5	++ 和 -- 运算符重载
6	赋值运算符重载
7	函数调用运算符 () 重载
8	下标运算符 [] 重载
9	类成员访问运算符 -> 重载

三、 实验内容1

1. 用C++编写程序：设计Student类，构造函数初始化学生信息（学号，姓名，成绩），重载提取运算符>>和插入<<实现学生信息的输入输出功能。

提示：对“<<”和“>>”重载的函数形式如下：

istream& operator >> (istream&, 自定义类&);//输入运算符>>重载

ostream& operator << (ostream&, 自定义类&);//输出运算符<<重载

重载流输入、输出运算符

- ✓ C++对终端IO流的操作，封装了I/O类。然后，这些类重载了“<<”和“>>”等运算符，实现对数据流的输入、输出操作。C++编译系统都在类库中提供输入流istream和输出流ostream，cin和cout分别是istream类和ostream类的对象。在类库提供的头文件中已经对“<<”和“>>”进行了重载，能用来输出和输入C++标准类型的数据。
- ✓ 用户自己定义的类型的数据，是不能直接用“<<”和“>>”来输出和输入的，如果想用它们输出和输入自己声明的类型的函数，必须对他们重载。对“<<”和“>>”重载的函数形式如下：

istream& operator >> (istream&, 自定义类&);//输入运算符>>重载

ostream& operator << (ostream&, 自定义类&);//输出运算符<<重载

即重载输入“>>”运算符的第一个参数和函数返回类型都必须是 istream& 类型，第二个参数就是要进行输入操作的类。重载输出“<<”运算符的第一个参数和函数返回类型都必须是ostream& 类型，第二个参数是要进行输出操作的类。因此，只能将重载“>>”和“<<”的函数作为 友元 函数或普通的函数，而不能将它们定义为成员函数。因为，对于“>>”和“<<” 运算符的左操作数是 cout 和 cin.

https://blog.csdn.net/feng19870412/article/details/127523429 原文有几处好像输入输出搞反了?

重载流输入、输出运算符

假设有如下调用：

student stud; //student 是自己定义的一个类

cout << stud; //输出stud 对象

那么“<<”运算符的 左 操作数是 cout 对象，或者，“>>”运算符的左操作数是 cin对象。如果要把“<<”运算符定义为“成员函数”，也只能定义为 cout 对象的成员函数，那么，上面的调用，可以如下：

cout.operator << (stud); //是由cout 对象调用它自己的operator <<() 函数来操作

所以，要想重载 operator <<( ); 函数来操作 student 类，只能够把 operator <<(); 函数声明为student类的友元函数，或者定义为一个普通函数。不能把它定义为student类的一个成员函数。

对于一个问题还需要解释：根据重载的 operator <<(); 函数如下：

ostream& operator << (ostream&, 自定义类&);

我们看到，其返回类型是 ostream& ，是一个输出流对象的引用。在函数 operator <<( ); 操作完之后，为什么还要返回一个 ostream& 这样一个输出流对象的引用呢？看如下的一个输出语句就知道了：int a = 162899;cout << "a = " << a;上面执行 cout 输出的时候，其实是调用了2个重载的 operator <<(); 函数：

重载流输入、输出运算符

(1) 首先执行cout << "a = " 来输出一个字符串“a = ”那么，调用：

ostream .operator <<(String) //重载函数是ostream 类的成员函数

operator <<(ostream, String) //重载函数不是ostream 的成员函数，但是，要接收2 个参数，第一个参数必须是ostream 类型，因为，我们执行cout << "a = " 语句的时候，"<<"运算符的左操作数是cout 对象，是ostream 类型，所以，该函数不是ostream 的成员函数，那么，第一个参数就是ostream 类型。所以，这两个函数，都可以实现 cout 输出字符串的操作。所以，先输出流字符串“a = ”。

(2) 然后，再执行“ << a ”语句，它必须有如下的函数定义，才能够输出。ostream .operatro <<(int)

operator <<(ostream, int)

所以，在“<< a”语句的左边，怎么样都必须要有一个 ostream 类型的对象，才能够顺利的完成“<< a”这样的操作。所以，在执行“cout << “a = ””这样的语言之后，该函数必须返回一个 ostream 类对象，以支持后面继续输出“<< a”的操作。所以，我们规定，在重载“<<”和“>>”操作符的时候，为了能够支持连续第使用该操作符，例如：int a, double d, float f;cin >> a >> d >> f; //输出数据cout << a << d << f; //输出数据那么，我们在重载了“<<”操作符之后，返回 ostream 类对象，在重载了“>>”操作符之后，返回istream类对象

三、 实验内容1参考代码

```
#include<iostream>
#include <string>
using namespace std;

class Student{
public:
    Student(){}
    Student(string name,…){…};
    friend ostream & operator<<(ostream&,Student&);
    friend istream & operator>>(istream&,Student&);
private:
    string name;
    int age;
};

ostream & operator<<(ostream& stream, Student& stu)
{ // 重载流插入运算符
    stream << stu.name << " " << stu.age << endl;
    return stream;
}
```

```
istream & operator>>(istream& stream, Student& stu){
 // 重载流提取运算符
    stream >> stu.name;
    stream >> stu.age;
    return stream;
}
int main(){

    Student stu;
    cout<<"Please enter name and age of a
            student:"<<endl;
    cin >> stu;  // 将调用重载流提取运算符函数
    cout<<"The information of student is as
            follows:"<<endl;
    cout << stu; // 将调用重载插入运算符函数

    // 剩下保存到文件里的操作请自己实现。
    return 0;
}
```

三、 实验内容1参考代码

多读多写多思考优缺点

- ✓ https://blog.csdn.net/weixin_43480094/article/details/83720377

三、 实验内容2

2. 用C++编写程序：有两个整数矩阵a和b，均为2行3列。求两个矩阵之和。重载运算符“+”，使之能用于矩阵相加。如c=a+b。

$$\begin{bmatrix}1 & 4 & 2\\ 2 & 0 & 0\end{bmatrix}+\begin{bmatrix}0 & 0 & 5\\ 7 & 5 & 0\end{bmatrix}=\begin{bmatrix}1+0 & 4+0 & 2+5\\ 2+7 & 0+5 & 0+0\end{bmatrix}=\begin{bmatrix}1 & 4 & 7\\ 9 & 5 & 0\end{bmatrix}$$

```
#include <iostream>
using namespace std;
class Matrix
 {
public:
   Matrix();
   friend Matrix operator+(Matrix &,Matrix &);
   void input();    //矩阵输入函数
   void display();  //矩阵输出显示函数
   private:
   int mat[2][3];
 };
```

三、 实验内容2参考代码

多读多写
多思考优缺点
多在电脑上实践（尝试修改代码）

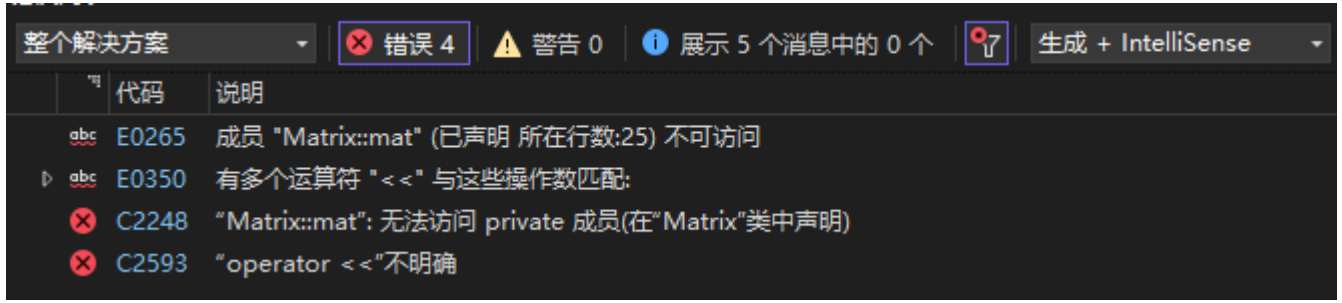
- ✓ https://blog.csdn.net/weixin_43470383/article/details/110562970

C++运算符重载时的函数参数一定是引用吗?

请同学们探索求证。

- ✓ https://www.zhihu.com/question/368377450
- ✓ 并不是，使用引用主要是为了减少对象拷贝时的开销，如果你完全不在意这个那就可以用普通的值传参就好了。
- ✓ 如果对流提取和插入运算符重载是必须用引用的。

探索求证:
上面链接代码 ostream& operator << (ostream& out, Matrix& m) // 定义重载运算符"<<"的友元函数
改为: ostream& operator << (ostream out, Matrix& m) // 定义重载运算符"<<"的友元函数
VS2022编译错误提示如下:



C++运算符重载时的函数参数一定是引用吗？

请同学们探索求证。

blog.csdn.net/youyou362/article/details/75023797

为什么这里的重载赋值运算符一定要用引用？ 这时实参传值给形参会创建一个临时变量，调用复制构造函数将实参复制给形参，实参的str与形参的str指向同一块内存。 所以，当赋值函数完成时，会清除函数的所占的内存， 形参就会被清除 ， 形参的str指向的内存也就会被清除。

学习的四重境界

知学、好学、会学、乐学

祝大家实验顺利！