1. 复习与导入：
2. 析构函数
3. 友元函数
4. 教学内容：

继承

1. 基类和派生类

1、概念

面向对象程序设计中，可以在已有类的基础上定义新的类，而不需要把已有的内容重新写一遍，叫做继承。已有类称为基类或父类，在基类的基础上建立的新类称为派生类或子类。

例：定义Student类和Student1类

#include<iostream>

class Student

{

private:

int num;

char name[20];

char sex;

public:

void display()

{

cout<<"num:"<<num<<endl;

cout<<"name:"<<name<<endl;

cout<<"sex:"<<sex<<endl;

}

};

class Student1

{

private:

int num;

char name[20];

char sex;

int age;

char addr[20];

public:

void display()

{

cout<<"num:"<<num<<endl;

cout<<"name:"<<name<<endl;

cout<<"sex:"<<sex<<endl;

cout<<"address:"<<addr<<endl;

}

};

例：利用继承方式重新定义类Student1

#include<iostream>

class Student

{

private:

int num;

char name[20];

char sex;

public:

void display()

{

cout<<"num:"<<num<<endl;

cout<<"name:"<<name<<endl;

cout<<"sex:"<<sex<<endl;

}

};

class Student1:public Student

{

private:

int age;

char addr[20];

public:

void display()

{

cout<<"age:"<<age<<endl;

cout<<"address:"<<addr<<endl;

}

};

2、定义派生类

定义派生类的一般形式：

class 派生类:[继承方式]基类名

{

派生类新增加的成员;

};

继承方式包括public,private和protected，继承方式是可选的，如果不写此项，则默认为private。

3、继承方式：

（1）公有继承public

基类的公用成员和保护成员在派生类中保持原有访问属性，其私有成员仍为私有

（2）私有继承private

基类的公用成员和保护成员在派生类中成了私有成员，其私有成员仍为基类私有

（3）受保护的继承protected

基类的公用成员和保护成员在派生类中成了保护成员，其私有成员仍为基类私有

（二）单继承

单继承就是每个派生类只有一个基类，派生类只从单个基类中继承属，注意基类中成员的访问权限和继承方式共同制约着它的访问属性。

（三）派生类的构造函数

单继承中不能继承基类的构造函数和析构函数，对于派生类要重新定义构造函数和析构函数。因为在派生类对象中也包含了基类数据成员的值，所以在定义一个派生类对象时，系统首先要通过派生类的构造函数调用基类的构造函数，对基类进行初始化，然后对派生类中新增加的成员初始化。

注意：派生类构造函数先调用基类构造函数，再执行派生类构造函数；析构函数的执行顺序是：先执行派生类的析构函数，再通过派生类的析构函数调用基类的析构函数。

（四）多继承

C++允许一个派生类同时继承多个基类，称为多重继承，简称多继承。

多继承派生类的一般格式：

class<派生类名>:<继承方式1><基类名1>,<继承方式2><基类名2>,<继承方式3><基类名3>……

{

新增成员列表

}

例如：class D:public A,private B,protected C

{

类D新增加的成员

}

多重继承派生类的构造函数形式与单继承时的构造函数形式基本相同，只是在初始表中包含多个基类构造函数。

例如：

派生类构造函数名（总参数表列）：基类1构造函数（参数表列），基类2构造函数（参数表列），基类3构造函数（参数表列）

｛

派生类中新增数据成员初始化语句

｝

派生类构造函数的执行顺序同样为：先调用基类的构造函数，再执行派生类构造函数的函数体。调用基类构造函数的顺序是按照声明派生类时基类出现的顺序。

（五）虚基类

在多继承中，若在多条继承路径上有公共基类，这个公共基类便会产生多个副本。换句话说，如果一个派生类有多个直接基类，而这些直接基类又有一个共同的基类，则在最终派生类中会保留该间接共同基类数据成员的多份同名成员。在一个类中保留间接共同基类的多份同名成员，这种现象是们不希望出现的。C++提供虚基类的方法，使得在继承间接共同基类时只保留一份成员。

虚基类是对派生类而言的，它本身的定义同基类一样，在定义派生类时声明该基类为虚基类即可，用关键字virtual加以说明。

例如，将类A声明为虚基类，方法如下：

Class A

{

……

}

Class B:virtual public A

{

……

}

Class C:virtual public A

{

……

}

注意：虚基类并不是在声明基类时声明的，而是在声明派生类时，指定继承方式时声明的。因为一个基类可以在生成一个派生类时作为虚基类，而在生成另一个派生类时不作为虚基类。

声明虚基类的一般形式为：

Class 派生类名:virtual 继承方式 基类名

经过这样的声明后，当基类通过多条派生路径被一个派生类继承时，该派生类只继承该基类一次。

虚基类的初始化也是利用构造函数完成的，如果在虚基类中定义了带参数的构造函数，而且没有定义默认构造函数，则在其所有派生类（包括直接派生或间接派生的派生类）中，通过构造函数的初始化表对虚基类进行初始化。

例如：

Class A

{

A(int i){}

};

Class B:virtual public A

{

B(int n):A(n){}

};

Class C:virtual public A

{

C(int n):A(n){}

};

Class D:public B,public C

{

D(int n):A(n),B(n),C(n){}

};

注意：在定义类D的构造函数时，与以往使用的方法有所不同。规定在最后的派生类中不仅要负责对其直接基类进行初始化，还要负责对虚基类初始化。