第四章学习的是派生类与继承

继承是什么意思呢？就是说本来定义了一个类，比如Student：

class Student

{public:

void display() //对共有成员函数display的定义

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

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

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

}

private: //Student 中的私有成员数据

int num;

string name;

char sex;

};

如果不凑巧，类外一个部门除了上面类中的name，number，sex外，还要用到age、address等

怎么办呢？当然可以重新定义个类，把所有的信息都放进去。但显然，人类这种懒惰的人不想这么做。于是，就诞生了派生类。

假设已经声明了一个基类Student（上面写的），再此基础上通过单继承建立一个派生类Student1：

class Student1:public Student //声明基类是Student

{public:

void display\_1() //新增加的成员函数display\_1

{cout<<"age"<<age<<endl;

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

private:

int age; //新增加的数据成员

string addr;

}；

第一行：class Student1:public Student

看出声明派生类一般形式：

class 派生类名：[继承方式] 基类名

{

派生类新增成员

}；

继承方式包括：public（公用的），private（私有的）和protected（受保护的），如不写，则默认为private。

我理解的类的生成过程经历了三个步骤：

1. 吸收基类成员 全部吸收(构造、析构除外）
2. 改造基类成员
3. 添加派生类新成员

**吸收基类成员:**在C++的继承机制中，派生类吸收基类中除构造函数和析构函数之外的全部成员。

**改造基类成员:**通过在派生类中定义同名成员（包括成员函数和数据成员）来屏蔽（隐藏）在派生类中不起作用的部分基类成员。

**添加派生类新成员:**仅仅继承基类的成员是不够的，需要在派生类中添加新成员，以保证派生类自身特殊属性和行为的实现。

**重名成员:**

1.派生类定义了与基类同名的成员，在派生类中访问同名成员时屏蔽了基类的同名成员

2.在派生类中使用基类的同名成员，显式地使用类名限定符：

类名 :: 成员

**访问静态成员**

基类定义的静态成员，将被所有派生类共享（基类和派生类共享基类中的静态成员）

根据静态成员自身的访问特性和派生类的继承方式，在类层次体系中具有不同的访问性质

派生类中访问静态成员，用以下形式显式说明：

类名 :: 成员 或通过对象访问 对象名 . 成员

这一章的程序最值得注意就是基类成员在派生类中的访问属性，防止非法访问。

Copyright ©2021-2099 Chen Yilu. All rights reserved