***单继承和多继承***

**1.阅读代码，并按要求练习。**

class A

{

public:

A(int num):data1(num){ }

~A()

{

cout<<" Destory A"<<endl;

}

void f() **const**

{

cout<<" Excute A::f() ";

cout<<" Data1="<<data1<<endl;

}

void g()

{

cout<<" Excute A::g() "<<endl;

}

private:

int data1;

};

class B:public A

{

public:

B(int num1,int num2):A(num1),data2(num2){}

~B() { cout<<" Destory B"<<endl; }

void f( ) const

{

cout<<" Excute B::f() ";

cout<<" Data1="<< data1;

cout<<" Data2="<<data2<<endl;

}

void f(int n) const

{

cout<<" Excute B::f(int) ";

cout<<" n="<<n;

cout<<" Data1="<< data1;

cout<<" Data2="<<data2<<endl;

}

void h()

{

cout<<" Excute B::h() "<<endl;

}

private:

int data2;

};

1）完成B类的构造函数，使得参数num1对应data1，num2对应data2；

2）尝试在main 函数中使用这两个类. 编译程序看是否有编译错误？

指出错误的原因。

3）将基类中的private改为protected,再编译。

理解protected访问权限，在public继承方式下的可访问性。

4）修改main函数，

例如：

void main()

{

B b(1,2);

b.f();

b.g();

b.f(3);

b.h();

}

看看哪些语句合法？为什么？执行的是基类的实现，还是派生类的实现？

5）将继承A类的继承方式改为private，编译能通过吗？

再执行4）中的main函数，看看哪些语句变得不合法了？为什么？

6）将继承A类的继承方式改回public，并实现B类自定义的拷贝构造和赋值函数。

7） 分别创建A和B类的两个对象a和b,分别执行a.f(),b.f(), a.g(),b.g(), a.f(1),b.f(1),a.h(),b.h()

哪些可以通过编译，执行结果如何？

8）增加代码A \* p=new B(1,2);,理解向上类型转换的安全性。

9）在8）的基础上，执行p->f(),输出是什么？与 B\* p=new B(1,2); p->f();的结果一样吗？

10）在8）的基础上，执行p->f(1),能通过编译吗？为什么？

11）在8）的基础上，执行p->g()和p->h()，能行吗？为什么？

12）在8）的基础上，执行delete p; ,输出是什么？B类的析构函数执行了吗？

**2.改成单继承**

class A

{

public:

A(int num):data(num) {}

void AFuncs() {cout<<"This is A \'s public function!"<<endl;}

protected:

int data;

};

class B

{

public:

B(int num):value(num) {}

void BFuncs() {cout<<"This is B \'s public function!"<<endl;}

protected:

int value;

};

class C:public A,private B

{

public:

C(int num1,int num2,int y);

void MyFuncs()

{

BFuncs();

cout<<"This function call B::BFuncs() !"<<endl;

}

private:

int yyy;

};

1)完成C类的构造函数。

2)在main 函数中可以向C类对象发送哪些消息？

3)在不改变C类的功能的条件下，利用类的组合重新定义并实现C类，使其变成单继承。

4)实现新的C类的构造，析构，拷贝、赋值函数。

**3.某同学设计开发一个游戏，游戏中有墙(Wall)和门(Door),他给出了如下的类定义：**

class Wall

{

public:

Wall():color(0) { cout<<"构造一面墙"<<endl;}

void Paint(int newColor)

{

color = newColor;

cout<<"用新颜色粉刷墙"<<endl;

}

int GetColor() const {return color;}

private:

int color;

};

class Door

{

public:

Door():openOrClose(false) { cout<<"构造一扇门"<<endl;}

void Open()

{

if (!IsOpened( ))

{

openOrClose = true;

cout<<"门被打开了"<<endl;

}

else

{

cout<<"门开着呢！"<<endl;

}

}

void Close()

{

if ( IsOpened( ) )

{

openOrClose = false;

cout<<"门被关上了"<<endl;

}

Else

{

cout<<"门关着呢！"<<endl;

}

}

bool IsOpened() const { return openOrClose;}

private:

bool openOrClose;

};

**1)请你用多重继承的方式，实现带有一扇门的墙(WallWithDoor)类。功能变更为：**

**a.当用红色粉刷墙时，关闭门。**

**b.当用绿色粉刷墙时，打开门。**

**c.当用其它颜色刷墙时，门的状态不变。**

**2)用单继承的方法，实现同样功能**

**3)用水平关联的方式，实现同样功能**