实验七 继承与派生（二）

1. 实验目的

掌握利用单继承和多重继承的方式定义派生类的方法；深刻理解在各种继承方式下构造函数和析构函数的执行顺序；理解和掌握公有继承、私有继承和保护继承对基类成员的访问机制；理解虚基类的概念以及引入虚基类的目的和作用。

1. 实验内容
2. 有以下程序，请完成下面工作:
3. 阅读程序，写出运行时输出的结果。

② 然后上机运行，验证结果是否正确。

1. 分析程序执行过程，尤其是调用构造函数和析构函数的过程。

#include<iostream>

using namespace std;

class A

{public:

A( ){cout<<″constructing A ″<<endl;}

~A( ){cout<<″destructing A ″<<endl;}

};

class B : public A

{public:

B( ){cout<<″constructing B ″<<endl;}

~B( ){cout<<″destructing B ″<<endl;}

};

class C : public B

{public:

C( ){cout<<″constructing C ″<<endl;}

~C( ){cout<<″destructing C ″<<endl;}

};

int main( )

{C c1;

return 0;

}

1. 设计一个圆类Circle和一个桌子类Table，再设计一个圆桌类Roundtable，它是从前两个类派生出来的。要求输出圆桌的高度、面积、颜色等数据，其中，类Circle包含数据成员radius和圆面积的成员函数，类Table包含数据成员height和返回高度的成员函数，类Roundtable再添加数据成员color和相应的成员函数。

#include <iostream>

#include <string>

using namespace std;

#define pi 3.1415926

class Table

{

public:

Table();

virtual ~Table();

public:

Table(int h);

void display();

public:

int height;

};

Table::Table(int h)

{

height=h;

}

void Table::display()

{

cout<<"heighe="<<height<<endl;

}

class Circle

{

public:

Circle();

virtual ~Circle();

public:

Circle(int r);

void math();

public:

int radius;

double area;

};

Circle::Circle(int r)

{

radius=r;

}

void Circle::math()

{

area=pi\*radius\*radius;

}

class Rundtable : public Circle,public Table

{

public:

Rundtable();

virtual ~Rundtable();

public:

Rundtable(int r,int h,string l):Circle(r),Table(h)

{

color=l;

};

void display1();

public:

string color;

};

void Rundtable::display1()

{

cout<<"area="<<area<<endl;

cout<<"color="<<color<<endl;

}

void main()

{

Rundtable r(2,4,"green");

r.math();

r.display();

r.display1();

}

1. 设计一个基类Person，包含name和age两个数据成员；由它派生出学生类Student和教师类Teacher，其中学生类添加学号，教师类添加职称信息；每个类均有构造函数和析构函数；编程实现，并用一些数据进行测试。

#include <iostream>

#include <string>

using namespace std;

class Person

{

public:

Person();

virtual ~Person();

public:

Person(string n,int a);

public:

string name;

int age;

};

Person::~Person()

{

cout<<"Person destructing"<<endl;

}

Person::Person(string n,int a)

{

name=n;

age=a;

cout<<"Person constructing"<<endl;

}

class Student : public Person

{

public:

Student();

virtual ~Student();

public:

Student(string n,int a,int u):Person(n,a)

{

num=u;

cout<<"Student constructing"<<endl;

}

void display1();

public:

int num;

};

Student::~Student()

{

cout<<"Student destructing"<<endl;

}

void Student::display1()

{

cout<<name<<" "<<age<<" "<<num<<endl;

}

class Teacher : public Person

{

public:

Teacher();

virtual ~Teacher();

public:

Teacher(string n,int a,string z):Person(n,a)

{

zhicheng=z;

cout<<"Teacher constructing"<<endl;

}

void display();

public:

string zhicheng;

};

Teacher::~Teacher()

{

cout<<"Teacher destructing"<<endl;

}

void Teacher::display()

{

cout<<name<<" "<<age<<" "<<zhicheng<<endl;

}

void main()

{

Teacher t("a",40,"teacher");

Student s("b",18,123);

t.display();

s.display1();

}

3. 实验小结