### 第一次实验

//类的定义：Renminbi.h

class Renminbi

{

public:

void display();

void set();

private:

int yuan;

int jiao;

int fen;

};

//主函数：main.cpp

#include<iostream>

#include"Renminbi.h"

using namespace std;

int main()

{

Renminbi r1;

r1.set();

r1.display();

return 0;

}

//成员函数：Renminbi.cpp

#include<iostream>

#include"Renminbi.h"

using namespace std;

void Renminbi::display()

{

cout<<yuan<<"元"<<jiao<<"角"<<fen<<"分"<<endl;

}

void Renminbi::set()

{

cout<<"依次输入元角分："<<endl;

cin>>yuan;

cin>>jiao;

cin>>fen;

}

//类的定义：rect.h

class rect

{

public:

void set();

void get\_r();

private:

int length;

int width;

};

//主函数：demo.cpp

#include<iostream>

#include"rect.h"

using namespace std;

int main()

{

rect r1;

r1.set();

r1.get\_r();

return 0;

}

//成员函数 rect.cpp

#include<iostream>

#include"rect.h"

using namespace std;

void rect::set()

{

cout<<"Please enter length&width"<<endl;

cin>>length;

cin>>width;

}

void rect::get\_r()

{

int area,circumference;

circumference=(length+width)\*2;

area=length\*width;

cout<<"circumference:"<<circumference<<endl;

cout<<"area:"<<area<<endl;

}

### 第二次实验

#include <iostream>

#define PI 3.14159265 //宏定义 PI=3.14159265

using namespace std;

class Cylinder

{

public:

Cylinder(); //声明一个无参的构造函数Cylinder

Cylinder(int r,int h) :radius(r),height(h){}; //定义一个有参数的构造函数，用参数的初始化表对数据成员初始化

Cylinder1(int rad,int hei) //定义有参数的构造函数

{

radius=rad;

height=hei;

cout<<"Constructor called."<<endl; //输出有关信息

}

~Cylinder() //定义析构函数

{

cout<<"Destructor called."<<endl; //输出指定的信息

}

void show\_data(); //声明成员函数

private:

int radius;

int height;

};

Cylinder::Cylinder() //在类外定义无参构造函数Cylinder

{

radius=10;

height=10;

}

void Cylinder::show\_data() //在类外定义成员函数volume

{

int volume; //定义变量体积

int side\_area; //定义变量侧面积

int bottom\_area; //定义变量底面积

side\_area=2\*PI\*radius\*height;

bottom\_area=PI\*radius\*radius;

volume=PI\*radius\*radius\*height;

cout<<"the side\_area:"<<side\_area<<endl; //输出侧面积

cout<<"the bottom\_area:"<<bottom\_area<<endl; //输出底面积

cout<<"the volume:"<<volume<<endl<<endl; //输出体积

}

int main()

{

Cylinder cylinder1; //建立对象cylinder1，不指定实参

cout<<"The data of cylinder1"<<endl; //输出相关的数据

cylinder1.show\_data();

Cylinder a[3]={ //定义对象数组并调用构造函数，提供实参

Cylinder(10,10),

Cylinder(20,20),

Cylinder(30,30)

};

cout<<"The data of a[0]"<<endl; //输出相关的数据

a[0].show\_data();

cout<<"The data of a[1]"<<endl;

a[1].show\_data();

cout<<"The data of a[2]"<<endl;

a[2].show\_data();

Cylinder cyl1(5,5); //建立对象cyl1

cout<<"The data of cyl1"<<endl; //输出相关数据

cyl1.show\_data();

return 0;

}

### 第三次实验

#include<iostream> //demo.cpp

#include "cylinder.h" //包含cylinder.h

#include "point.h" //包含point.h

#include "circle.h" //包含circle.h

#define PI 3.14159265 //宏定义PI

using namespace std;

int main()

{

Cylinder cylinder1(10,20,30,40);

cylinder1.display(); //输出数据

return 0;

}

#include<iostream> //point.cpp

#include "point.h" //包含 point.h

#define PI 3.14159265 //宏定义PI

using namespace std;

Point::Point() //无参构造函数初始化

{

x=0.0;

y=0.0;

}

Point::Point(float newx,float newy) //有参构造函数初始化

{

x=newx;

y=newy;

}

Point::show\_point() //输出x,y

{

cout<<"central\_point: ("<<x<<","<<y<<")"<<endl;

}

#include<iostream> //circle.cpp

#include "circle.h" //包含cricle.h

#define PI 3.14159265 //宏定义 PI

using namespace std;

Circle::Circle() //无参构造函数

{

r=0.0;

}

Circle::Circle(float newr) //有参构造函数

{

r=newr;

}

Circle::area() //面积函数

{

return r\*r\*PI;

}

Circle::display() //输出r,area

{

cout<<"circle\_rad:"<<r<<endl;

cout<<"circle\_area:"<<area()<<endl;

}

#include<iostream> //cylinder.cpp

#include "cylinder.h" //包含cylinder.h

#include "point.h" //包含point.h

#include "circle.h" //包含circle.h

#define PI 3.14159265 //宏定义PI

using namespace std;

Cylinder::Cylinder(float newh,float newr,float newx,float newy):Circle(newr),p(newx,newy) //派生类构造函数

{

h=newh;

}

Cylinder::volume() //派生类成员函数

{

return area()\*h;

}

Cylinder::display() //派生类成员函数

{

Circle::display(); //输出基类成员函数

p.show\_point(); //输出子对象

cout<<"cylinder\_height:"<<h<<endl; //输出派生类的数据

cout<<"cylinder\_volume:"<<volume()<<endl;

}

### 第四次实验

**Header Files:**

#ifndef \_POINT //point.h

#define \_POINT

class Point

{

public:

Point(); //声明构造函数

Point(float,float);

show\_point(); //声明成员函数

protected:

float x;

float y;

};

#endif

#ifndef \_CIRCLE //circle.h

#define \_CIRCLE

class Circle

{

public:

Circle(); //声明构造函数

Circle(float);

display(); //声明成员函数

area();

protected:

float r;

};

#endif

#ifndef \_CYLINDER //cylinder.h

#define \_CYLINDER

#include "circle.h"

#include "point.h"

class Cylinder:public Circle

{

public:

Cylinder(float,float,float,float); //声明构造函数

display(); //声明成员函数

volume();

protected:

Point p; //定义子对象

float h;

};

#endif

//currency.h

#pragma once

class Currency

{

public:

Currency(); //声明构造函数

Currency(float,float);

void set(float,float); //声明成员函数

virtual void print\_exchange()=0; //纯虚函数

protected:

float japan; //数据成员

float europe;

};

//Ja.h

#pragma once

#include "Currency.h"

class Eur:public Currency

{

public:

Eur(); //声明无参构造函数

Eur(float,float,float); //声明有参构造函数

void print\_exchange(); //声明成员函数

protected:

float total; //数据成员

};

//Ja.h

#pragma once

#include "Currency.h"

class Ja:public Currency

{

public:

Ja(); //声明无参构造函数

Ja(float,float,float); //声明有参构造函数

void print\_exchange(); //声明成员函数

protected:

float total; //数据成员

};

//currency.cpp

#include <iostream>

#include "currency.h"

using namespace std;

Currency::Currency() //无参构造函数

{

japan=0.0;

europe=0.0;

}

Currency::Currency(float newj,float newe) //有参构造函数

{

japan=newj;

europe=newe;

}

void Currency::set(float newj,float newe) //成员函数

{

japan=newj;

europe=newe;

}

//Eur.cpp

#include <iostream>

#include "Eur.h"

using namespace std;

Eur::Eur() //无参构造函数

{

total=0.0;

}

Eur::Eur(float newj,float newe,float newt):Currency(newj,newe) //有参构造函数

{

total=newt;

}

void Eur::print\_exchange() //成员函数

{

cout<<total<<"欧元可兑换美元："<<total\*1.1634<<endl;

}

//Ja.cpp

#include <iostream>

#include "Ja.h"

using namespace std;

Ja::Ja() //无参构造函数

{

total=0.0;

}

Ja::Ja(float newj,float newe,float newt):Currency(newj,newe) //有参构造函数

{

total=newt;

}

void Ja::print\_exchange() //成员函数

{

cout<<total<<"日元可兑换美元:"<<total/116.30<<endl;

}

//demo.cpp

#include <iostream>

#include "currency.h"

#include "Eur.h"

#include "Ja.h"

using namespace std;

int main()

{

Currency \*p; //定义指向基类的指针

Ja j(0,0,100); //建立Ja类对象j

Eur e(0,0,100); //建立Eur类对象e

p=&j; //使指针指向Ja类对象

p->print\_exchange();

p=&e; //使指针指向Eur类对象

p->print\_exchange();

return 0;

}