**实验五**

**类与对象**

**【实验目的与要求】**

1、掌握声明类的方法，类和类的成员的概念以及定义对象的方法。

2、初步掌握用类和对象编制基于对象的程序。

3、学习检查和调试基于对象的程序。

**【实验内容】**

1、有以下程序：

#include<iostream>

using namespace std；

class Time

// 定义 Time 类

{

public：

// 数据成员为公用的

int hour;

int minute;

int sec ;

};

int main()

{

Time tl;

//定义 t1 为 Time 类对象

cin>>t1.hour;

//输入设定的时间

cin>>t1.minute;

cin>>t1.sec;

cout<<t1.hourl<<”：”<<t1.minute<<”：”<<t1.sec<<endl ;

return 0;

}

改写程序，要求：

(1)将数据成员改为私有的；

(2)将输入和输出的功能改为由成员函数实现；

(3)在类体内定义成员函数。

然后编译和运行程序。请分析什么成员应指定为公用的?什么成员应指定为私有的? 什么函

数最好放在类中定义? 什么函数最好在类外定义?

2、分别给出如下的 3 个文件：

(1)含类定义的头文件 student.h，

//student.h

(这是头文件，在此文件中进行类的声明)

class Student

//类声明

{

public:

//公用成员函数原型声明

void display();

private:

int num;14

char name[20];

char sex;

};

(2)包含成员函数定义的源文件 student.cpp

//student.cpp

在此文件中进行函数的定义

#include <iostream>

#include”student.h”

//不要漏写此行，否则编译通不过

void Student::display()

//在类外定义 display 类函数

{

cout<<”num：”<<num<<endl;

cout<<”name：”<<name<<endl;

cout<<”sex：”<<sex<<endl;

}

(3)包含主函数的源文件 main.cpp。

为了组成一个完整的源程序，应当有包括主函数的源文件：

//main.cpp

主函数模块

#include <iostream>

//将类声明头文件包含进来

#include “student.h”

int main()

{

Student stud;

//定义对象

Student stud1(007,”tcg”,’m’);

stud.display();

//执行 stud 对象的 display 函数

return 0;

}

请完善该程序，在类中增加一个对数据成员赋初值的成员函数 set\_value。上机调试并运行。

3、需要求 3 个长方柱的体积，请编一个基于对象的程序。数据成员包括 length(长)、width(宽)、

height(高)。要求用成员函数实现以下功能：

(1)由键盘分别输入 3 个长方柱的长、宽、高；

(2)计算长方柱的体积；

(3)输出 3 个长方柱的体积。

请编程序，上机调试并运行。

4、编写程序：定义抽象基类 Shape，由它派生出五个派生类：Circle（圆形）、Square（正

方形）、Rectangle（ 长方形）、Trapezoid （梯形）和 Triangle （三角形），用虚函数分别计算

各种图形的面积，并求出它们的和。要求用基类指针数组。使它的每一个元素指向一个派生类的

对象。

注：主函数中定义如下对象

Circle circle(10.2);

Square square(3);

Rectangle rectangle(3,4);

Trapezoid trapezoid(2.0,4.5,3);

Triangle triangle(4,5,6);5、设计一个函数，返回两个向量的交集，函数头如下：

template<typename T>

vector<T> intersection(const vector<T>& v1, const vector<T>& v2)

两个向量的交际为它们的共同包含的元素，编写一个测试程序，提示用户输入两个向量，每

个包含 5 个字符串，打印输出它们的交集

**三、算法分析、程序与结果**

1. #include<iostream>

#include<string>

using namespace std;

class Time // 定义 Time 类

{

private: // 数据成员为公用的

int hour;

int minute;

int sec;

public:

void SetTime(int H, int M, int S)

{

hour = H; minute = M; sec = S;

}

void Showtime()

{

cout << hour << ":" << minute << ":" << sec << endl;

}

};

int main()

{

Time tl; //定义 t1 为 Time 类对象

double x, y, z;

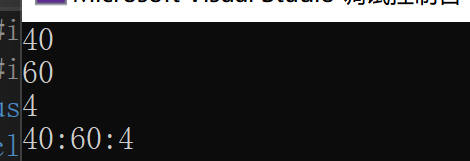
cin >> x >> y >> z;

tl.SetTime( x,y,z);

tl.Showtime();

return 0;

}



2#include <iostream>

using namespace std;

#include"student.h" //不要漏写此行，否则编译通不过

void Student::display() //在类外定义 display 类函数

{

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

cout << "name：" << name << endl;

cout << "sex：" << sex << endl;

}

void Student::set\_value()

{

cin >> num;

cin >> name;

cin >> sex;

}

#include <iostream> //将类声明头文件包含进来

#include "student.h"

int main()

{

Student stud; //定义对象

stud.set\_value();

stud.display(); //执行 stud 对象的 display 函数

return 0;

}class Student //类声明

{

public: //公用成员函数原型声明

void display();

void set\_value();

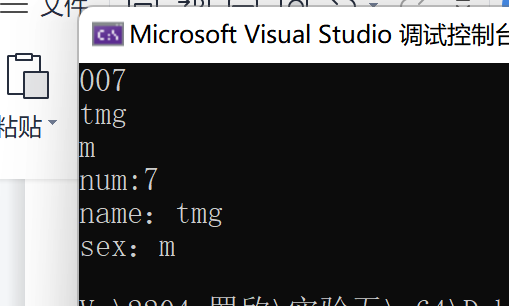
private:

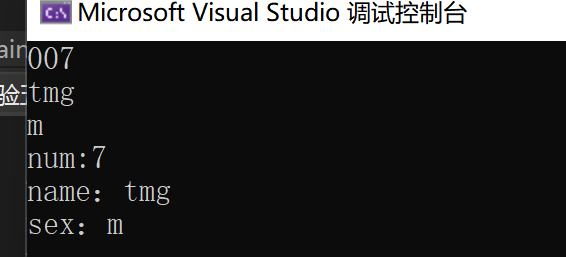
int num;

char name[20];

char sex;

};



.

3.

#include <iostream>

using namespace std;

class rectangle

{

public:

void Setcin();

void SetV();

private:

double length;

double width;

double height;

};

void rectangle::Setcin() {

cout << "请输入长，宽，高 ";

cin >> length;

cin >> width;

cin >> height;

}

void rectangle::SetV() {

double V = length \* width \* height;

cout << "长方柱的体积为：" << V << endl;;

}

int main() {

rectangle c1,c2,c3;

c1.Setcin();

c1.SetV();

c2.Setcin();

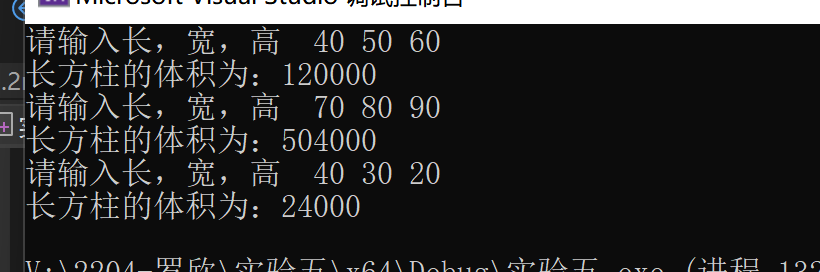
c2.SetV();

c3.Setcin();

c3.SetV();

return 0;

}



4#include <iostream>

using namespace std;

class Shape {

private:

int X, Y;

public:

virtual double area() {

return 0.0;

}

};

const double PI = 3.14159;

class Circle :public Shape

{

private:

double radius;

public:

Circle(int X, int Y, double R) :Shape()

{

radius = R;

}

double area()

{

return PI \* radius \* radius;

}

};

class Square :public Shape

{

private:

double height;

public:

Square(int X, int Y, double H) :Shape()

{

height = H;

}

double area()

{

return height \* height;

}

};

class Rectangle :public Shape

{

private:

double weight, height;

public:

Rectangle(int X, int Y,int W,int H) :Shape()

{

weight = W; height = H;

}

double area()

{

return weight\*height ;

}

};

class Trapezoid :public Shape

{

private:

double height, Sh, Dh;

public:

Trapezoid(int X, int Y, double H,double S,double D) :Shape()

{

height = H; Sh = S; Dh = D;

}

double area()

{

return (Sh + Dh) \* height / 2;

}

};

class Triangle :public Shape

{

private:

double PD, ID, HD;

public:

Triangle(int X, int Y, double P, double I,double H) :Shape()

{

PD = P; ID = I; HD = H;

}

double area()

{

double S, K;

S = (PD + ID + HD) / 2;

K = S \* (S - PD) \* (S - HD) \* (S - ID);

return sqrt(K);

}

};

double total(Shape\* shape[], int num) {

double tot = 0;

for (int i = 0; i < num; i++) {

tot += shape[i]->area();

}

return tot;

}

int main() {

Circle circle(0, 0, 10.2);

cout << "circle.area()=" << circle.area() << endl;

Square square(0, 0, 3);

cout << "square.area()=" << square.area() << endl;

Rectangle rectangle(0,0,3, 4);

cout << "rectangle.area()=" << rectangle.area() << endl;

Trapezoid trapezoid(0,0,2.0, 4.5, 3);

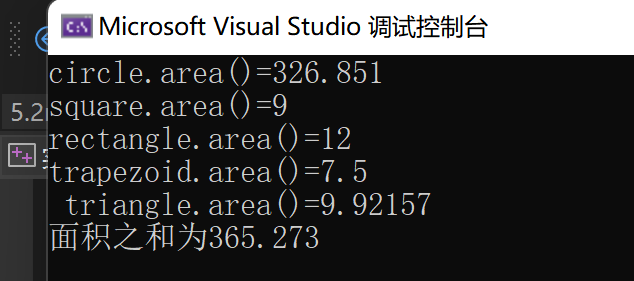
cout << "trapezoid.area()=" << trapezoid.area() << endl;

Triangle triangle(0,0,4, 5, 6);

cout << " triangle.area()=" << triangle.area() << endl;

Shape\* shape[5] = { &circle,&square,&rectangle,&trapezoid,&triangle };

cout << "面积之和为" << total(shape, 5) << endl;

}  
.

5.#include <iostream>

#include <vector>

using namespace std;

template<typename T>

vector<T> intersection(const vector<T>& v1, const vector<T>& v2);

int main() {

vector <string> v1;

vector <string> v2;

for (int i = 0; i < 5; i++)

{

string s;

cout << "请输入向量1";

cin >> s;

v1.push\_back(s);

}

for (int i = 0; i < 5; i++)

{

string s;

cout << "请输入向量2";

cin >> s;

v2.push\_back(s);

}

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

if (v1[i]==v2[j])

{

cout << "相同的部分为"<<v1[i] << endl;;

}

}

}

}

