实验三：

**Tdate.h**

#pragma once

#include<iostream>

using namespace std;

class TDate

{

public:

TDate();

TDate(int y, int m = 11, int d = 11)

{

this->year = y;

this->month = m;

this->day = d;

cout << "TDate 的构造函数！！！" << endl;

}

void SetDate(int,int,int);

void AddOneDay();

void print()

{

cout << year << "---" << month << "---" << day << endl;

}

~TDate();

private:

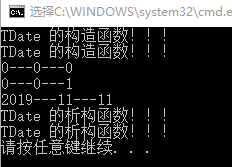
int year;

int month;

int day;

}

结果：



**Tdate.cpp**

#include "TDate.h"

#include<iostream>

using namespace std;

TDate::TDate()

{

this->day = 0;

this->month = 0;

this->year = 0;

cout << "TDate 的构造函数！！！" << endl;

}

void TDate::SetDate(int y, int m, int d)

{

year = y;

month = m;

day = d;

}

void TDate::AddOneDay()

{

this->day++;

}

TDate::~TDate()

{

cout << "TDate 的析构函数！！！" << endl;

}

**main.cpp**

#include"TDate.h"

#include<string>

#include<iostream>

using namespace std;

int main()

{

TDate s, s1(2019);

s.print();

s.AddOneDay();

s.print();

s1.print();

return 0;

}

实验四：

**Goods.h**

#pragma once

#include<string>

#include<iostream>

using namespace std;

class Goods

{

private:

const string Name;

static int weights;

int buy;

int sale;

public:

Goods(int b, int s, string name) :Name(name)

{

buy = b;

sale = s;

cout << "Goods 的构造函数！！！" << endl;

cout << Name << " " << "buy: " << buy << " " << "sale: " << sale << endl;

}

void initWeights()

{

weights = buy - sale;

if (weights <= 0){

cout << "目前的库存为0" << endl;

}

else{cout << "目前库存为" << weights << endl;}

}

void Buy(int b) {

weights += b;

cout << "买进" << b << "后，" << "目前库存为" << weights << endl;

}

void Sale(int s) {

weights -= s;

cout << "卖出" << s << "后，" << "目前库存为" << weights << endl;

}

~Goods() {cout << "Goods的析构函数！！！" << endl;}

};

// 初始化静态成员变量

int Goods::weights = 0;

**main.h**

#include"Goods.h"

#include<string>

#include<iostream>

using namespace std;

int main()

{

Goods g(8, 7,"百事可乐");

g.initWeights();

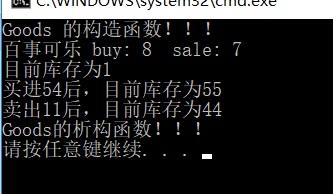
g.Buy(54);

g.Sale(11);

return 0;

}

结果：



实验四：

**People.h**

#pragma once

#include<iostream>

#include<string>

#include"TDate.h"

using namespace std;

class People

{

private:

string Name;

TDate date;

int Height;

string Address;

public:

People(string n, TDate t, int h, string a) :Name(n), date(t), Height(h), Address(a){}

People()

{

Name = "王佳豪";

date = TDate(1998, 8, 18);

Height = 175;

Address = "中北大学";

cout << "People 的构造函数！！！" << endl;

}

void setAddress(string newaddress)

{

this->Address = newaddress;

}

void display()

{

date.print();

cout << Name << "---" << "---" << Height << "---" << Address << endl;

}

~People()

{

cout << "People 的析构函数！！！" << endl;

}

};

**main.h**

#include"TDate.h"

#include"People.h"

#include<string>

#include<iostream>

using namespace std;

int main()

{

TDate t(1999, 1, 1);

People \*p1 = new People();

p1->display();

p1->setAddress("清华大学");

p1->display();

People \*p2 = new People("张三", t, 160, "野鸡大学");

p2->display();

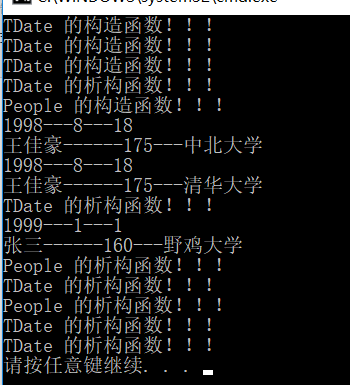
delete(p1);

delete(p2);

return 0;

}

**结果：**



实验六：

**Vehicle.h**

#pragma once

#include<iostream>

#include<string>

using namespace std;

class vehicle

{

private:

int MaxSpeed;

int Weight;

public:

vehicle(int m, int w)

{

MaxSpeed = m;

Weight = w;

}

void Run()

{

cout << "---调用run---" << endl;

cout << "MaxSpeed = " << MaxSpeed << endl;

}

void Stop() {

cout << "---调用Stop---" << endl;

}

};

**bicycle.h**

#pragma once

#include<iostream>

#include<string>

#include"vehicle.h"

using namespace std;

class bicycle:virtual public vehicle

{

private:

int height;

public:

bicycle():vehicle(5,10)

{

height = 2;

}

};

**motorcar.h**

#pragma once

#include<iostream>

#include<string>

#include"vehicle.h"

using namespace std;

class motorcar :virtual public vehicle

{

private:

int SetNum;

public:

motorcar():vehicle(7,8)

{

SetNum = 5;

}

};

**motorcyle.h**

#pragma once

#include<iostream>

#include<string>

#include"bicycle.h"

#include"motorcar.h"

using namespace std;

class motocycle :public bicycle, public motorcar

{

public :

motocycle(int m,int w):vehicle(m,w) {}

};

**main.h**

#include"vehicle.h"

#include"bicycle.h"

#include"motorcar.h"

#include"motorcycle.h"

#include<string>

#include<iostream>

using namespace std;

int main()

{

vehicle v(4,6);

motorcar mc;

bicycle b;

motocycle mo(10,20);

v.Run();

mc.Run();

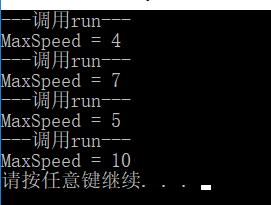
b.Run();

mo.Run(); // 没有virtual会出现二义性

return 0;

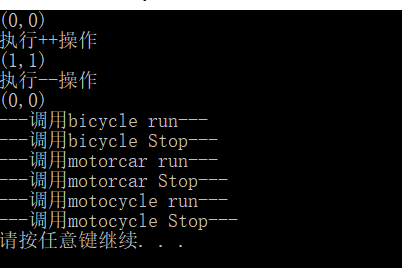
}

结果：



七：

结果：



实验7：

**Point.h**

#pragma once

#include<iostream>

#include<string>

using namespace std;

class Point

{

public:

Point() {

x = 0;

y = 0;

}

Point(int x, int y)

{

this->x = x;

this->y = y;

}

Point operator++()

{

++x;

++y;

return \*this; //返回当前对象值

}

Point operator--()

{

--x;

--y;

return \*this; //返回当前对象值

}

void display()

{

cout << "(" << x << "," << y << ")" << endl;

}

private:

int x, y;

};

**vehicle.h**

#pragma once

#include<iostream>

#include<string>

using namespace std;

class vehicle

{

private:

int MaxSpeed;

int Weight;

public:

vehicle(int m, int w)

{

MaxSpeed = m;

Weight = w;

}

virtual void Run()

{

cout << "---调用vehicle run---" << endl;

}

virtual void Stop()

{

cout << "---调用vehicle Stop---" << endl;

}

};

**bicycle.h**

#pragma once

#include<iostream>

#include<string>

#include"vehicle.h"

using namespace std;

class bicycle:virtual public vehicle

{

private:

int height;

public:

bicycle():vehicle(5,10)

{

height = 2;

}

virtual void Run() {cout << "---调用bicycle run---" << endl;}

virtual void Stop() {cout << "---调用bicycle Stop---" << endl;}

};

**motorcar.h**

#pragma once

#include<iostream>

#include<string>

#include"vehicle.h"

using namespace std;

class motorcar :virtual public vehicle

{

private:

int SetNum;

public:

motorcar():vehicle(7,8)

{

SetNum = 5;

}

virtual void Run()

{

cout << "---调用motorcar run---" << endl;

}

virtual void Stop()

{

cout << "---调用motorcar Stop---" << endl;

}

};

**motorcycle.h**

#pragma once

#include<iostream>

#include<string>

#include"bicycle.h"

#include"motorcar.h"

using namespace std;

class motocycle :public bicycle, public motorcar

{

public :

motocycle(int m,int w):vehicle(m,w) {}

virtual void Run()

{

cout << "---调用motocycle run---" << endl;

}

virtual void Stop()

{

cout << "---调用motocycle Stop---" << endl;

}

};

**main.h**

#include"vehicle.h"

#include"bicycle.h"

#include"motorcar.h"

#include"motorcycle.h"

#include"Point.h"

#include<string>

#include<iostream>

using namespace std;

int main()

{

p.display();

cout << "执行++操作" << endl;

++p;

p.display();

cout << "执行--操作" << endl;

--p;

p.display();

vehicle \*v;

bicycle b;

motorcar mc;

motocycle mo(1,1);

/\*

\* 动态关联

\*/

v = &b;

v->Run();

v->Stop();

v = &mc;

v->Run();

v->Stop();

v = &mo;

v->Run();

v->Stop();

return 0;

}

实验8：

**Shape.h**

#pragma once

class Shape

{

public:

virtual float area() { return 0; } //面积

virtual void DisPlayShape() = 0; // 纯虚类

virtual void Print() = 0;

};

**CPoint.h**

#pragma once

#include<iostream>

#include<string>

#include"shape.h"

using namespace std;

class CPoint : public Shape

{

public:

CPoint(float x, float y, string c)

{

this->color = c;

this->x = x;

this->y = y;

}

virtual void DisPlayShape() {cout << "这是一个点！！！" << endl;}

float GetX() {return x;}

float GetY() {return y;}

string GetColor() {return color;}

virtual void Print() {

cout << "点: （" << x << "," << y << ") " << color << endl;

}

private:

float x, y;

string color;

};

**CLine.h**

#pragma once

#include<iostream>

#include<string>

#include"CPoint.h"

using namespace std;

class CLine : public CPoint

{

public:

CLine(float x, float y, string color, float l) :CPoint(x, y, color), length(l) {}

virtual void DisPlayShape() {cout << "这是一条线!!!" << endl;}

float GetLength() {return length;}

void SetLength(float l) {length = l;}

virtual void Print()

{

cout << "线: （" << GetX() << "," << GetY() << ") " << GetColor() << " " << "l = " << length << endl;

}

private:

float length;

};

**CSquare.h**

#pragma once

#include<iostream>

#include<string>

#include"CLine.h"

using namespace std;

class CSquare :public CLine

{

public:

CSquare(float x, float y, string c, float l) :CLine(x, y, c, l) {}

virtual void DisPlayShape()

{

cout << "这是一个正方形！！！" << endl;

}

virtual float area() {return GetLength() \* GetLength();}

virtual void Print()

{

cout << "正方形: （" << GetX() << "," << GetY() << ") " << GetColor() << " " << "l = " << GetLength() << endl;

}

};

**CCircle.h**

#pragma once

#include<iostream>

#include<string>

#include<math.h>

#include"CPoint.h"

using namespace std;

class CCircle : public CPoint

{

public:

CCircle(float x, float y, string color, float r) :CPoint(x, y, color), Redius(r) {}

virtual void DisPlayShape()

{

cout << "这是一个圆！！！" << endl;

}

virtual float area()

{

return acos(-1)\*Redius\*Redius;

}

float GetRedius()

{

return Redius;

}

void SetRedius(float r)

{

Redius = r;

}

virtual void Print()

{

cout << "圆: （" << GetX() << "," << GetY() << ") " << GetColor() << " " << "r = " << Redius << endl;

}

private:

float Redius;

};

**main.h**

#include"shape.h"

#include"CPoint.h"

#include"CLine.h"

#include"CCircle.h"

#include"CSquare.h"

#include<string>

#include<iostream>

using namespace std;

int main()

{

CPoint p(1.1, 2.2, "red");

CLine L(2.2, 3.3, "blue", 9.9);

CCircle c(3.3, 4.4, "black", 5.5);

CSquare cs(4.4, 5.5, "yello", 6.6);

/\*

\* 静态关联（重写）

\*/

p.Print();

L.Print();

c.Print();

cs.Print();

cout << endl;

/\*

\* 动态关联（向上转型）

\*/

Shape \*s;

s = &p;

s->DisPlayShape();

cout << "点的坐标是: (" << p.GetX() << "," << p.GetY() << ")" << endl;

cout << "area = " << s->area() << endl << endl;

s = &L;

s->DisPlayShape();

cout << "中点的坐标是: (" << p.GetX() << "," << p.GetY() << ")" << endl;

cout << "area = " << s->area() << endl << endl;

s = &c;

s->DisPlayShape();

cout << "圆心的坐标是: (" << p.GetX() << "," << p.GetY() << ")" << endl;

cout << "area = " << s->area() << endl << endl;

s = &cs;

s->DisPlayShape();

cout << "中点的坐标是: (" << p.GetX() << "," << p.GetY() << ")" << endl;

cout << "area = " << s->area() << endl << endl;

return 0;

}

结果：

