#ifndef Slump\_H

#define Slump\_H

// 水果

class Slump

{

private:

protected:

int \_price;

public:

Slump(int price):\_price(price){};

Slump(Slump&) = default;

// 虚函数，如果希望子类可以有自己特定的版本，应使用virtual标识

virtual double Price(){

return \_price;

};

// 纯虚函数

// 含有纯虚函数的类是抽象类，抽象类负责定义接口，子类实现接口

virtual int Color() = 0;

// 当我们删除一个指向子类的父类指针时将产生未定义行为

// 定义析构函数为虚析构函数，一般父类都将析构函数定义为虚析构函数，这样子类的析构函数都是虚的

virtual ~Slump() = default;

// 定义析构函数将阻止编译器合成移动操作，父类没有移动操作将阻止子类合成移动操作，如果需要，我们需要显示定义

Slump &operator=(Slump&&) = default;

Slump(Slump&&) = default;

};

#endif

#include <utility>

#ifndef Apple\_H

#define Apple\_H

#include "./Slump.h"

// 继承 Slump，继承只会继承 Slump 的public和protected的成员

// public 指定 Slump 的 public 和 protected 成员放到 Apple 的 public 和 protected 中

// protected 指定 Slump 的 public 和 protected 成员放到 Apple 的 protected 中

// private 指定 Slump 的 public 和 protected 成员放到 Apple 的 private 中

// 虽然子类无法访问父类的 private 成员，但他们却是存在在当前对象中

// final 指示当前类为最终类，不能再被继承

// c++ 允许多继承

class Apple final : public Slump

{

// 友元可以通过Apple访问从Slump继承的成员

// 友元只作用于当前类，不能传递，也不能继承

friend double sum(Apple apple);

// // Apple的友元不能访问Slump的protected和private成员

// friend double sum(Slump slump);

private:

/\* data \*/

public:

// 初始化父类

// 类不能继承默认，拷贝，移动构造函数，如果派生类没有直接定义，编译器将合成这些函数

Apple(int price) : Slump(price){};

Apple(Apple &apple) : Slump(apple){}; // 拷贝构造

Apple(Apple &&apple) : Slump(std::move(apple)){}; // 移动构造

// override 标识符，覆盖虚函数

double Price() override

{

return Slump::Price(); // 使用基类的方法

};

// // final 标识符，不允许后续的基类覆盖

// double Price() final;

int Color() override { return 1; }

// 更改继承成员的访问级别

using Slump::\_price;

};

void fun1()

{

Apple apple1(0);

// 调用基类的 Price

apple1.Slump::Price();

}

#endif

#include <iostream>

#include <vector>

#include <memory>

#include "./Apple.h"

#include "./Slump.h"

using namespace std;

int sum(int l, int r);

int main()

{

// // 错误，Slump是一个抽象类

// vector<Slump> slumps;

// // 如果Slump不是一个抽象类，你以为slumps[0]存放的是Apple，错误，存放的是Slump

// slumps.push\_back(Apple(0));

// 解决以上问题

vector<shared\_ptr<Slump>> slumps1;

slumps1.push\_back(make\_shared<Apple>(0));

cout << "enter key" << endl;

cin.get();

return 0;

}