1。已知圆形类定义如下：

class Circle {

protected:

float radius, area;

public:

Circle ( ) { radius = area = 0.0f ; }

Circle ( float r ) {

radius = r;

area = 3.14159\* radius\* radius;

}

virtual float getArea( ) { return area; }

float getRadius( ) { return radius; }

};

要求:1）在Circle类基础之上公有派生出圆柱体类Cylinder，新增一个私有数据成员：height，用于表示圆柱体高度，并在圆柱体类中设计成员函数getArea，同名覆盖Circle中的同名虚函数，用于计算圆柱体的**表面积**。

2）设计main函数对进行测试，其中Cylinder类中getArea函数要求使用基类Circle的指针通过赋值兼容规则进行调用。

class Circle {

protected:

float radius, area;

public:

Circle ( ) { radius = area = 0.0f ; }

Circle ( float r ) {

radius = r;

area = 3.14159\* radius\* radius;

}

virtual float getArea( ) { return area; }

float getRadius( ) { return radius; }

};

class Cylinder: public Circle {

protected:

float height;

public:

Cylinder (float, float);

float getHeight ( ) { return height; }

float getArea( ) {

return(2\*3.14159\*radius\* height);

}

};

Cylinder:: Cylinder (float r, float h) : Circle(r)

{ height = h ;

}

void main()

{

Circle \*p ,r(10);

p = &r;

cout<<p->getArea()<<endl;

Cylinder c(10,15);

p = &c;

cout<<p->getArea()<<endl;

}

2 class vehicle // 定义汽车类

{

protected:

　　int wheels; // 车轮数

　　float weight; // 重量

　 public:

　　vehicle(int wh,float we) {

　　 wheels=wh;

　　 weight=we;

　　}

　　int get\_wheels(){return wheels; }

　　float get\_weight(){return weight}

**virtual float efficiency( ) {return 0.0;}**

　　void show(){

　　cout << "车轮:" << wheels << "个" << endl;

　　cout << "重量:" << weight << "公斤" << endl;

　　}

};

要求：

（1）在类vehicle上派生出小汽车类car，新增成员：载人数passenger\_car，并设计虚函数**efficiency**同名覆盖基类，该函数功能为返回小汽车类的效率（效率=载人数\*80/重量），公式中的80是假设平均每人重80公斤。

（2）在类vehicle上派生出卡车类truck，新增成员：载人数passenger\_truck，载重量payload，并设计虚函数**efficiency**同名覆盖基类，该函数功能为返回货车类的效率（效率=载重量/（载重量+重量）），每个类都有相关数据的输出方法。

（3）并再main函数中进行功能测试。如其中，小汽车车轮：4个；重量：1500公斤；载人：5人；卡车车轮：10个，重量：8000公斤，载人：3人，载重：340000公斤。并通过基类的指针（或引用）分别指向两个子类，用以测试运行时的多态虚函数efficiency的调用。

#include<iostream>

using namespace std;

class vehicle // 定义汽车类

{

protected:

int wheels; // 车轮数

float weight; // 重量

public:

vehicle(int wh,float we)

{

wheels=wh;

weight=we;

}

int get\_wheels(){return wheels; }

float get\_weight(){return weight;}

void show()

{

cout << "车轮:" << wheels << "个" << endl;

cout << "重量:" << weight << "公斤" << endl;

}

virtual float efficiency( ) {return 0;}

};

class car:public vehicle // 定义小车类

{

int passenger\_car; // 载人数

public:

car(int wh,float we,int passengers=4);

int get\_passengers();

void show();

float efficiency( ) {return passenger\_car/weight;}

};

class truck:public vehicle // 定义卡车类

{

int passenger\_truck; // 载人数

float payload; // 载重量

public:

truck(int wh,float we,int passengers=2,float max\_load=24000.00);

int get\_passengers();

float efficiency();

void show();

};

car::car(int wh, float we,int passengers) :vehicle (wh, we) {passenger\_car=passengers;}

int car::get\_passengers (){ return passenger\_car;}

void car::show()

{

cout <<" 车型:小车" << endl;

vehicle::show();

cout << "载人:" << passenger\_car << "人" << endl;

cout << endl;

}

truck::truck(int wh, float we, int passengers, float max\_load):vehicle(wh,we)

{

passenger\_truck=passengers;

payload=max\_load;

}

int truck::get\_passengers( ) {return passenger\_truck;}

float truck::efficiency( ) {return payload/weight;}

void truck::show( )

{

cout <<"车型:卡车" << endl;

vehicle:: show ();

cout << "载人:" << passenger\_truck << "人" << endl;

cout << "载重:" << payload << "吨" << endl;

cout << endl;

}

void main ()

{

vehicle \*p;

car car1(4,2000,5);

truck tru1(10,8000,3,340000);

cout << "输出结果" << endl;

car1. show ();

tru1. show ();

p = &car1;

cout<<"小汽车的效率为："<<p->efficiency()<<endl;

p = &tru1;

cout<<"小汽车的效率为："<<p->efficiency()<<endl;

}

3. 已知一个抽象类base，要求：

（1）由它派生出工程师类engineer，包含职称（prof）和专业（major）数据成员以及相关的成员函数。

（2）再由工程师类engineer派生出主任工程师类leader，包含职务（job）和部门（dep）数据成员以及相关的成员函数。

（3）并在主函数中用“李明，42岁，职称——高级工程师；专业——电站锅炉设计；职务——处长；部门——设计处，”的个人信息进行测试， 读入并输出李明的个人信息。

class base // 基类

{protected:

char\* name;// 姓名

int age; // 年龄

public:

virtual void setname(char na[] )=0; //设置姓名

virtual void setage(int a )=0;//设置年龄

char\* getname() { return name; }

int getage() { return age; }

};

答案：

#include<iostream>

#include<string>

using namespace std;

class base // 基类

{protected:

char\* name;// 姓名

int age; // 年龄

public:

virtual void setname(char na[] )=0; //设置姓名

virtual void setage(int a )=0;//设置年龄

char\* getname() { return name; }

int getage() { return age; }

};

class engineer: public base // 工程师类

{protected:

char \*major; // 专业

char \*prof; // 职称

public:

void setname(char na[])

{

name=new char[strlen(na)+1];

strcpy(name,na);

}

void setage(int a) { age=a; }

void setmajor(char maj [])

{

major=new char[strlen(maj)+1];

strcpy (major,maj);

}

void setprof(char pf[])

{

prof=new char[strlen(pf)+1];

strcpy (prof, pf);

}

char\*getmajor() {return major; }

char\*getprof() {return prof; }

char\* getname() { return name; }

int getage() { return age; }

} ;

class leader: public engineer // 主任工程师类

{

char \*job;//职务

char \*dep;//部门

public:

void setjob(char jb[])

{

job=new char[strlen(jb)+1];

strcpy (job, jb);

}

void setdep(char dp[])

{

dep=new char [strlen (dp) +1] ;

strcpy (dep, dp);

}

char \*getjob() { return job; }

char \*getdep() { return dep; }

};

int main()

{

leader c;

c.setname("李明");

c.setage(42);

c.setjob("处长");

c.setdep("设计处");

c.setmajor("电站锅炉设计");

c.setprof("高级工程师");

cout<< "输出结果："<<endl;

cout << " " << c.getname() << "，年龄" << c.getage() << "岁，担任" <<c.getdep() <<c.getjob() <<"，"<<endl;

cout << " " << c.getprof() << "，从事" << c.getmajor()<< "专业" << "。 " << endl;

return 0;

}