第六章代码：

/\*

\* ex6\_1.cpp

\*

\* 1. 定义下列类的数据成员并提供适当的初始化方法：

\* （1）日期Date

\* （2）员工Employee

\* （3）学生Student

\*

\*

\*/

#include<string>

#include<vector>

#include<iostream>

using namespace std;

class Date{

int year = 1970;

int month = 1;

int day = 1;

public:

//constructor//构造函数

Date(int y, int m, int d){

if (y > 0)

year = y;

else

year = 1970;

if(m >= 1 && m <= 12)

month = m;

else

month = 1;

if(leapYear() && month == 2){

if(d >= 1 && d <= 29)

day = d;

else

day = 1;

}

else{

if(d >= 1 && d <= days[month])

day = d;

else

day = 1;

}

}

//default constructor//默认构造函数

Date() : Da

te(1970, 1, 1){}

private:

int days[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

bool leapYear() const{判断是否为闰年的函数

return ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0));

}

};

class Employee{

string name;

unsigned int id;

string department;

string title;

double salary = 0;

string address;

vector<string> phone;

public:

Employee(string ename, unsigned eid) : name(ename), id(eid){}

Employee(string ename, unsigned eid, string dept, string ttl)

: name(ename), id(eid), department(dept), title(ttl) {}

};

class Student{

string name;

unsigned int id;

string department;

public:

Student(string sname, unsigned sid) : name(sname), id(sid){}

Student(string sname, unsigned sid, string dept) : name(sname), id(sid), department(dept){}

};

int main()

{

return 0;

}

/\*

\* ex6\_2.cpp

\*

\* 2. 设计表示时间的Time类，要求：

\* （1）有表示时、分、秒的数据成员；

\* （2）正确的初始化操作，检验数据的合法性；

\* （3）有时、分、秒的get和set函数；

\* （4）以24小时格式或12小时加上午下午的格式输出当前时间。

\*

\*/

#include <string>

#include <iostream>

#include <sstream>

using namespace std;

//class declaration

class Time{

int hour = 0;

int minute = 0;

int second = 0;

public:

Time();

Time(int h, int m, int s);

//getter and setter

int getHour() const;

int getMinute() const;

int getSecond() const;

void setHour(int h);

void setMinute(int m);

void setSecond(int s);

//display

string toString24()const;

string toString12()const;

void display24()const;

void display12()const;

};

//member functions definition

Time::Time() : hour(0), minute(0), second(0)

{}

Time::Time(int h, int m, int s) : Time(){

if(h >= 0 && h < 24)

hour = h;

if(m >= 0 && m < 60)

minute = m;

if(s >= 0 && s < 60)

second = s;

}

//getter and setter

int Time::getHour() const{

return hour;

}

int Time::getMinute() const{

return minute;

}

int Time::getSecond() const{

return second;

}

void Time::setHour(int h){

if(h >= 0 && h < 24)

hour = h;

}

void Time::setMinute(int m){

if(m >= 0 && m < 60)

minute = m;

}

void Time::setSecond(int s){

if(s >= 0 && s < 60)

second = s;

}

//display

string Time::toString24()const{

ostringstream os;

os << hour << ":" << minute << ":" << second;

return os.str();

}

string Time::toString12()const{

ostringstream os;

if(hour > 0 && hour < 12)

os << hour << ":" << minute << ":" << second << " A.M.";

else if(hour == 0)

os << (hour + 12) << ":" << minute << ":" << second << " A.M.";

else if(hour == 12)

os << hour << ":" << minute << ":" << second << " P.M.";

else

os << (hour - 12) << ":" << minute << ":" << second << " P.M.";

return os.str();

}

void Time::display24()const{

cout << toString24();

}

void Time::display12()const{

cout << toString12();

}

int main()

{

Time t1(23, 15, 35);

Time t2(12, 35, 7);

Time t3(0, 23, 19);

Time t4;

cout << t1.toString12() << "\t" << t1.toString24() << endl;

cout << t2.toString12() << "\t" << t2.toString24() << endl;

cout << t3.toString12() << "\t" << t3.toString24() << endl;

cout << t4.toString12() << "\t" << t4.toString24() << endl;

return 0;

}

/\*

\* ex6\_3.cpp

\*

\* 3. 设计一个Account类：

\* （1）static数据成员annual表示每个存款人的年利率；

\* （2）private数据成员savings表示当前存款额；

\* （3）成员函数calculate()用于计算月利息，并将利息加进savings；

\* （4）static成员函数modify()改变annual；

\* 在主程序中实例化两个不同的Account对象saver1和saver2，

\* 账户余额分别为2000.00和3000.00，将annual设置为3%，

\* 计算每个存款人的月息并打印结果，再将annual设置为4%，

\* 重新计算每个存款人的月息并打印新的结果。

\*

\*

\*/

#include<iostream>

using namespace std;

class Account{

public:

Account(double s):savings(s){}

static void modify(double newAnnual){

annual += newAnnual;

}

double caculate(){

double interest = savings\*annual/12;

savings+= interest;

return interest;

}

double getSavings(){return savings;}

private:

double savings;

static double annual;

};

double Account::annual = 0.03;

int main()

{

Account saver1(2000),saver2(3000);

cout<<saver1.getSavings()<<"\t"<<saver1.caculate()<<endl;

cout<<saver2.getSavings()<<"\t"<<saver2.caculate()<<endl;

Account::modify(0.04);

cout<<saver1.getSavings()<<"\t"<<saver1.caculate()<<endl;

cout<<saver2.getSavings()<<"\t"<<saver2.caculate()<<endl;

}

/\*

\* ex6\_4.cpp

\*

\* 4.设计一个名为Fan的类表示风扇，要求包括：

（1）3个名为SLOW、MEDIUM和FAST的常量，值分别为1、2、3，表示风速；

（2）int类型的私有数据成员speed，表示风扇的速度，默认值为SLOW；

（3）bool类型的私有数据成员on，表示风扇是否打开，默认值为false；

（4）double类型的私有数据成员radius，表示风扇的半径，创建后不可修改，默认值5；

（5）string类型的私有数据成员color，表示风扇的颜色，默认值为white；

（6）4个数据成员的访问器，非const成员的修改器；

（7）创建默认风扇的构造函数；

（8）根据参数值创建风扇的构造函数；

（9）成员函数status()，返回风扇的状态字符串：

如果风扇是打开的，字符串中包括风扇的速度、颜色和半径；否则，返回的字符串中包括风扇关闭、颜色和半径。

编写测试程序，创建三个不同颜色和大小的风扇，进行改变速度、开、关等操作，输出这些风扇的状态。

\*

\*/

#include <string>

#include <iostream>

#include <sstream>

using namespace std;

class Fan{

public:

Fan();

Fan(string color, double radius);

const string& getColor() const;

void setColor(const string& color = "white");

bool isOn() const;

void turnOn();

void turnOff();

double getRadius() const;

int getSpeed() const;

void setSpeed(int speed = SLOW);

string status()const;

static const int SLOW = 1;

static const int MIDUM = 2;

static const int FAST = 3;

private:

int speed = SLOW;

bool on = false;

string color = "white";

const double radius = 5;

}; //end of class definition

//--------------------------------------

Fan::Fan(){}

Fan::Fan(string c, double r) : color(c), radius(r){}

const string& Fan::getColor() const {

return color;

}

void Fan::setColor(const string& color) {

this->color = color;

}

bool Fan::isOn() const {

return on;

}

void Fan::turnOn() {

on = true;

}

void Fan::turnOff(){

on = false;

}

double Fan::getRadius() const {

return radius;

}

int Fan::getSpeed() const {

return speed;

}

void Fan::setSpeed(int speed) {

this->speed = speed;

}

string Fan::status()const{

ostringstream os;

if(isOn())

os << "Fan status: " << speed << " " << color << " " << radius;

else

os << "Fan is OFF: " << color << " " << radius;

return os.str();

}

//--------------------------------------

int main()

{

Fan fan1;

cout << "fan 1:" << endl;

cout << fan1.status() << endl;

fan1.turnOn();

fan1.setSpeed(Fan::FAST);

cout << fan1.status() << endl;

Fan fan2("blue", 8);

cout << "fan 2:" << endl;

cout << fan2.status() << endl;

fan2.turnOn();

cout << fan2.status() << endl;

Fan fan3("red", 3);

cout << "fan 3:" << endl;

cout << fan3.status() << endl;

fan3.turnOn();

fan3.setColor("green");

cout << fan3.status() << endl;

fan3.turnOff();

cout << fan3.status() << endl;

return 0;

}

/\*

\* ex6\_5.cpp

\*

\* 5.设计并实现一个平面点类Point，要求：

\* （1）用x,y两个坐标值表示一个点；

\* （2）正确初始化每个点；默认坐标值为原点；

\* （3）计算点到原点的距离；

\* （4）计算到另一个点的距离；

\* （5）获取点的x、y坐标值；

\* （6）设置点的x、y坐标；

\* （7）移动点到新位置；

\* 编写测试程序。

\*

\*

\*/

#include <iostream>

#include <cmath>

using namespace std;

//---------------------------------

class Point{

public:

Point(int px=0, int py=0):x(px),y(py){}

double distance()const;

double distance(Point& pt)const;

int getX()const {return x;}

int getY()const {return y;}

void setX(int px){x = px;}

void setY(int py){y = py;}

void moveto(const Point& pt);

void moveto(int px, int py);

private:

int x = 0;

int y = 0;

};

double Point::distance()const {

return sqrt(x\*x+y\*y);

}

double Point::distance(Point& pt)const{

int a = pt.x - x;

int b = pt.y - y;

return sqrt(a\*a + b\*b);

}

void Point::moveto(const Point& pt){

x = pt.x; y = pt.y;

}

void Point::moveto(int px, int py){

x = px; y = py;

}

//-------------------------------------------

int main()

{

Point p1(1,3), p2(4,5), p3(2);

cout<<"p1: "<<p1.getX()<<","<<p1.getY()<<endl;

cout<<"p1 to origin:"<<p1.distance()<<endl;

cout<<"p2:"<<p2.getX()<<","<<p2.getY()<<endl;

cout<<"p1 to p2:"<<p1.distance(p2)<<endl;

p1.moveto(p3);

cout<<"p1 moveto p3:"<<p1.getX()<<","<<p1.getY()<<endl;

}

/\*

\* ex6\_6.cpp

\*

\* 6. 设计并实现一个日期类Date，要求：

（1）有表示年、月、日的数据成员；

（2）正确初始化年月日，并验证数据合法性，初始化后日期值不变；

（3）有获取年、月、日的get函数；

（5）实现Date nextDay()函数，返回表示下一天的Date对象；

（6）实现int difference(Date)函数，返回当前对象和参数指定日期之间的天数差；

（7）以yyyy-mm-dd的格式输出当前日期。

编写测试程序。

\*

\*/

#include <string>

#include <iostream>

#include <sstream>

using namespace std;

class Date{

public:

//constructor

Date(int y, int m, int d);

Date();

int getDay() const;

int getMonth() const;

int getYear() const;

Date nextDay()const;

int difference(const Date& date);

string toString()const;

private:

int year = 1970;

int month = 1;

int day = 1;

int days[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

static bool leapYear(int year);

bool leapYear()const;

int daysInTheYear()const;

};

Date::Date(int y, int m, int d){

if (y > 0)

year = y;

else

year = 1970;

if(m >= 1 && m <= 12)

month = m;

else

month = 1;

if(leapYear() && month == 2){

if(d >= 1 && d <= 29)

day = d;

else

day = 1;

}

else{

if(d >= 1 && d <= days[month])

day = d;

else

day = 1;

}

}

Date::Date() : Date(1970, 1, 1){}

Date Date::nextDay() const {

if(day == 31 && month == 12)

return Date(year + 1, 1, 1);

if(day == 29 && month == 2 && leapYear())

return Date(year, 3, 1);

if(day == days[month] && month != 12)

return Date(year, month + 1, 1);

return Date(year, month, day+1);

}

int Date::difference(const Date& date)

{

int dif = 0;

if(this->year == date.year)

dif = this->daysInTheYear() - date.daysInTheYear();

else if(this->year < date.year)

{

for(int y = this->year; y < date.year; ++y)

{

if(leapYear(y))

dif += 366;

else

dif += 365;

}

dif = dif - this->daysInTheYear() + date.daysInTheYear();

}

else{

for(int y = date.year; y < this->year; ++y)

{

if(leapYear(y))

dif += 366;

else

dif += 365;

}

dif = dif - date.daysInTheYear() + this->daysInTheYear();

}

return dif;

}

int Date::getDay() const {

return day;

}

int Date::getMonth() const {

return month;

}

int Date::getYear() const {

return year;

}

bool Date::leapYear(int year){

return ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0));

}

bool Date::leapYear() const{

return leapYear(year);

}

string Date::toString() const {

ostringstream os;

os << year << "-" << month << "-" << day;

return os.str();

}

int Date::daysInTheYear()const{

int sum = 0;

for(int i = 1; i < month; ++i)

sum += days[i];

sum += day;

if(leapYear() && month > 2)

++sum;

return sum;

}

//---------------------------------------------

int main()

{

Date d1(2016, 2, 29);

Date d2(2015, 2, 28);

Date d3(2011, 9, 30);

Date d4(2011, 12, 31);

Date d5(2010, 9, 10);

cout << d1.toString() << endl;

cout << d1.nextDay().toString() << endl;

cout << d1.toString() << "\t" << d2.toString() << "\t"

<< d1.difference(d2) << endl;

cout << d4.toString() << "\t" << d4.nextDay().toString() << endl;

cout << d5.toString()<< "\t" << d3.toString()

<< "\t" << d5.difference(d3) << endl;

cout << d3.toString() << "\t" << d3.nextDay().toString() << endl;

cout << d5.toString() << "\t" << d5.nextDay().toString() << endl;

return 0;

}

/\*

\* ex6\_7.cpp

\*

\* 7. 实现复数类ComplexNumber

\* 复数加、减、乘、除的公式如下：

(a+bi) + (c+di) = (a+c) + (b+d)i

(a+bi) - (c+di) = (a-c) + (b-d)i

(a+bi) \* (c+di) = (ac-bd) + (bc + ad)i

(a+bi) / (c+di) = (ac+bd)/(c2+d2) + (bc-ad)i/(c2+d2)

\*

\*/

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

class ComplexNumber{

public:

ComplexNumber(double r = 0, double i = 0): real(r), imag(i){}

double getImaginary() const;

double getReal() const;

string toString()const;

ComplexNumber add(const ComplexNumber& c)const;

ComplexNumber sub(const ComplexNumber& c)const;

ComplexNumber mul(const ComplexNumber& c)const;

ComplexNumber div(const ComplexNumber& c)const;

private:

double real = 0;

double imag = 0;

};

double ComplexNumber::getImaginary() const {

return imag;

}

double ComplexNumber::getReal() const {

return real;

}

string ComplexNumber::toString() const {

ostringstream os;

if(real == 0){

if(imag == 0)

os << 0;

else

os << imag << "i";

}

else{

if(imag == 0)

os << real;

else if(imag > 0)

os << real << "+" << imag << "i";

else

os << real << imag << "i";

}

return os.str();

}

ComplexNumber ComplexNumber::add(const ComplexNumber& c) const {

return ComplexNumber(real + c.real, imag + c.imag);

}

ComplexNumber ComplexNumber::sub(const ComplexNumber& c) const {

return ComplexNumber(real - c.real, imag - c.imag);

}

//(a+bi) \* (c+di) = (ac-bd) + (bc + ad)i

//(a+bi) / (c+di) = (ac+bd)/(c2+d2) + (bc-ad)i/(c2+d2)

ComplexNumber ComplexNumber::mul(const ComplexNumber& c) const {

ComplexNumber cn;

cn.real = real \* c.real - imag \* c.imag;

cn.imag = imag \* c.real + real \* c.imag;

return cn;

}

ComplexNumber ComplexNumber::div(const ComplexNumber& c) const {

ComplexNumber cn;

double cds = c.real \* c.real + c.imag \* c.imag;

cn.real = (real \* c.real + imag \* c.imag) / cds;

cn.imag = (imag \* c.real - real \* c.imag) / cds;

return cn;

}

//------------------------------

int main()

{

ComplexNumber c1(1, 2), c2(3, 4), c3(-3, 2), c4(2, -4), c5(-1, -2), c6;

cout <<"c1: " << c1.toString() << endl;

cout <<"c2: " << c2.toString() << endl;

cout <<"c3: " << c3.toString() << endl;

cout <<"c4: " << c4.toString() << endl;

cout <<"c5: " << c5.toString() << endl;

cout <<"c6: " << c6.toString() << endl;

cout <<"c1 + c2 = " << c1.add(c2).toString() << endl;

cout <<"c3 + c2 = " << c3.add(c2).toString() << endl;

cout <<"c4 + c2 = " << c4.add(c2).toString() << endl;

cout <<"c1 - c2 = " << c1.sub(c2).toString() << endl;

cout <<"c1 \* c2 = " << c1.mul(c2).toString() << endl;

cout <<"c1 / c2 = " << c1.div(c2).toString() << endl;

return 0;

}

/\*

\* ex6\_8.cpp

\*

\* 8. 设计并实现一个圆形类Circle，要求：

\* （1）由圆心和半径描述圆形（可以使用第5题定义的Point类）；

\* （2）能够计算面积、计算周长、获取半径、移动和缩放；

\* （3）合理的初始化；默认圆心为原点，默认半径为1；

\* 编写测试程序。

\*

\*/

#include <iostream>

using namespace std;

//--------------------------------

class Circle{

public:

Circle(double r, int px=0, int py=0);

double area()const;

double perimeter()const;

double getRadius()const{return radius;}

void moveto(int px, int py){ x = px; y = py; }

void scale(double fac){ radius \*= fac; }

private:

double radius;

int x,y;

static const double PI;

};

const double Circle::PI = 3.1415926;

Circle::Circle(double r, int px, int py){

radius =r;

x = px;

y = py;

}

double Circle::area()const{

return PI\*radius\*radius;

}

double Circle::perimeter()const{

return PI\*radius\*2;

}

//--------------------------------

int main()

{

Circle c1(2.5);

cout<<"c1: radius="<<c1.getRadius()<<endl;

cout<<"area="<<c1.area()<<endl;

cout<<"perimeter="<<c1.perimeter()<<endl;

c1.scale(2);

cout<<"c1: radius="<<c1.getRadius()<<endl;

cout<<"area="<<c1.area()<<endl;

cout<<"perimeter="<<c1.perimeter()<<endl;

}

/\*

\* ex6\_9.cpp

\* 9. 设计并实现一个矩形类Rectangle，要求：

\* （1）用左上角和右下角的坐标描述矩形（可以使用第5题定义的Point类）；

\* （2）计算面积、周长；

\* （3）移动矩形；

\* （4）合理的初始化；默认宽和高都是1；

\* 编写测试程序。

\*

\*

\*/

#include <iostream>

#include <cmath>

using namespace std;

//-------------------------------

class Rectangle;

class Point{

public:

Point(int px =0, int py =0):x(px),y(py){}

friend class Rectangle;

private:

int x,y;

};

//-------------------------------

class Rectangle{

public:

Rectangle(int lux, int luy, int rdx=0, int rdy=0):leftup(lux,luy),rightdown(rdx,rdy){}

double area()const{

int x,y;

x = leftup.x - rightdown.x;

y = leftup.y - rightdown.y;

return abs(x\*y);

}

double perimeter()const{

int x,y;

x = abs(leftup.x - rightdown.x);

y = abs(leftup.y - rightdown.y);

return (x+y)\*2;

}

void move(int x, int y=0){

leftup.x += x;

leftup.y += y;

rightdown.x += x;

rightdown.y += y;

}

private:

Point leftup, rightdown;

};

//-------------------------------

int main()

{

Rectangle r(1,2,4,5);

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

cout<<"perimeter="<<r.perimeter()<<endl;

r.move(10,1);

}

/\*

\* ex6\_10.cpp

\*

\* 10. 设计并实现一个三角形类Triangle，要求：

\* （1）用三条边描述三角形；

\* （2）计算面积、周长；

\* （3）合理的初始化，注意检验三条边长度的合法性；默认三边长都为1；

\* 编写测试程序。

\*

\*

\*/

#include<cassert>

#include<cmath>

#include <iostream>

using namespace std;

//-----------------------------------

class Triangle

{

public:

Triangle(double a=1, double b=1, double c=1);

~Triangle();

double area()const;

double perimeter()const;

double getA()const;

double getB()const;

double getC()const;

void setA(double a);

void setB(double b);

void setC(double c);

private:

double a, b, c;

};

//----------------------------------

//global function

bool validate(double a, double b, double c)

{

if((a<=0) || (b<=0) || (c<=0))

return false;

if((a+b<=c) || (a+c<=b) || (b+c<=a))

return false;

return true;

}

//--------------------------------

Triangle::Triangle(double x, double y, double z)

{

bool v = validate(x,y,z);

assert(v);

a=x; b=y; c=z;

}

Triangle::~Triangle()

{}

//virtual functions

double Triangle::area()const

{

double p = (a+b+c)/2;

double s = p\*(p-a)\*(p-b)\*(p-c);

return sqrt(s);

}

double Triangle::perimeter()const

{

return a+b+c;

}

//others......

double Triangle::getA()const

{ return a; }

double Triangle::getB()const

{ return b; }

double Triangle::getC()const

{ return c; }

void Triangle::setA(double d)

{

bool v = validate(d, b,c);

assert(v);

a = d;

}

void Triangle::setB(double d)

{

bool v = validate(a, d,c);

assert(v);

b = d;

}

void Triangle::setC(double d)

{

bool v = validate(a, b,d);

assert(v);

c = d;

}

//--------------------------

int main(int argc, char \*argv[])

{

Triangle t(5,3,4);

cout<<"Triangle t:\t"<<"a: "<<t.getA()

<<" b: "<<t.getB()<<" c: "<<t.getC()<<endl;

cout<<"area="<<t.area()<<"\tperimeter="<<t.perimeter()<<endl;

}