/\*\*

\* 日期:2018-11-22

\* 题目要求：复数类

\* 编写：dextercai（CWZ）

\* IT IS A WINDOWS-936 FILE

\*\*/

#include <iostream>

#include <cmath>

using namespace std;

class Complex

{

public:

double R,I;

Complex();

~Complex();

Complex(const Complex &c);

Complex(float,float);

void printout();

double mod();

Complex &operator+(Complex &c);

Complex &operator+(float c);

};

Complex::Complex()

{

R=I=0.0;

}

Complex::Complex(float r,float i)

{

R = r;

I = i;

}

Complex::~Complex(){}

Complex::Complex(const Complex &c){

this->R = c.R;

this->I = c.I;

}

void Complex::printout(){

cout << R << "+" << I << "i";

}

double Complex::mod(){ //取模方法

return sqrt(R\*R+I\*I);

}

Complex &Complex::operator+(Complex &c){ //操作符+重载

R += c.R;

I += c.I;

return \* this;

}

Complex &Complex::operator+(float c){ //操作符+重载

R += c;

return \* this;

}

int main(){

Complex C1 = Complex(1,1);

Complex C2 = Complex(4,2);

C1 = C1 + C2;

C1.printout();

cout << endl;

C2 = C2 + 5;

C2.printout();

cout << endl;

cout << C2.mod();

}

/\*\*

\* 日期:2018-11-22

\* 题目要求：学生类

\* 编写：dextercai（CWZ）

\* IT IS A WINDOWS-936 FILE

\*\*/

#include <iostream>

#include <cstring>

using namespace std;

class Student{

public:

int no,score1,score2,score3,average;

string name;

Student(const Student &p);

Student();

Student(string, int, int, int, int);

~Student();

void print();

};

Student::Student(const Student &p){ //深拷贝

name = p.name;

no = p.no;

score1 = p.score1;

score2 = p.score2;

score3 = p.score3;

average = (score1 + score2 + score3)/3.0;

}

Student::Student(){ //初始化

name = {0};

no = score1 = score2 = score3 = average = 0;

cout << "Constructor" << endl;

}

Student::Student(string namea, int noa, int score1a, int score2a, int score3a){ //有参初始化

no = noa;

name = namea;

score1 = score1a;

score2 = score2a;

score3 = score3a;

average = (score1 + score2 + score3)/3.0;

}

Student::~Student(){ //析构函数

cout << "Destructor" << endl;

}

void Student::print(){ //数据打印

cout << this->no << "\t"

<< this->name.c\_str() << "\t" <<

this->score1 << "\t" <<

this->score2 << "\t" <<

this->score3 << "\t";

cout << this->average << endl;

}

int main(){

void sortf(Student \*arr);

void Studentprint(Student \*arr);

Student arr[5]{

Student("Yang",102,85,80,78),

Student("Chen",103,77,70,83),

Student("Qian",104,70,67,60),

Student("Li",105,72,70,69),

Student("Zhou",101,93,89,87)

};

sortf(arr);

Studentprint(arr);

}

void sortf(Student \*arr){ //冒泡排序

for(int i = 4; i >=0 ; i--){

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

if(arr[i].average > arr[j].average){

Student stu = arr[i];

arr[i] = arr[j];

arr[j] = stu;

}

}

}

}

void Studentprint(Student \*arr){ //数据打印

cout << "No\tname\tscore1\tscore2\tscore3\taverage\r\n";

for(int i = 0; i <= 4; i++){

arr[i].print();

}

}

#include <iostream>

#include <cstring>

using namespace std;

enum sex{ //枚举 定义男女

Female=0,

Male=1

};

struct DATE{ //日期

int year,month,day;

};

class Library{ //定义图书馆类

string itemName;

public:

Library(string iN);

Library(const Library &c);

~Library();

void display();

};

Library::Library(string iN){

this->itemName = iN;

}

Library::Library(const Library &c){

this->itemName = c.itemName;

}

Library::~Library(){

cout << "destructing Library class." << endl;

}

void Library::display(){

cout<< "ItemName:" << this->itemName <<"\t";

}

class Book:public Library{ //书本类继承图书馆类

char zone;

int barCode;

public:

Book(string iN,char z,int bC);

Book(const Book &c);

~Book();

void display();

};

Book::Book(string iN, char z, int bC):Library(iN){ //初始化方法

this->zone = z;

this->barCode = bC;

}

Book::Book(const Book &c):Library(c){ //深拷贝方法

this->zone = c.zone;

this->barCode = c.barCode;

}

Book::~Book() { //析构方法

cout << "destructing Book class." << endl;

}

void Book::display(){ //显示数据方法

Library::display()

cout << "Zone:" << this->zone << "\tBarCode:" << this->barCode << "\t";

}

class Student:public Library{ //学生类继承图书馆类

sex sexid;

int stuNum;

int borrowBookNum;

public:

Student(string iN,sex si,int sN,int bBN);

Student(const Student &c);

~Student();

void display();

};

Student::Student(string iN, sex si, int sN, int bBN) :Library(iN) { //初始化方法

sexid = si;

stuNum = sN;

if(bBN >5)

throw -5;

else

borrowBookNum = bBN;

}

Student::Student(const Student &c) :Library(c) { //深拷贝方法

this->sexid = c.sexid;

this->stuNum = c.stuNum;

this->borrowBookNum = c.borrowBookNum;

}

Student::~Student() { //析构方法

cout << "destructing Student class." << endl;

}

void Student::display(){ //显示数据方法

Library::display()

cout << "Sexid:" << this->sexid

<< "\tStudentNumber:" << this->stuNum

<< "\tBorrowBookNumber:" << this->borrowBookNum << "\t";

}

class Journal:public Book{ //期刊杂志继承图书类

DATE publishDate;

public:

Journal(string iN,char z,int bC,DATE pD);

Journal(const Journal &c);

~Journal();

void display();

};

Journal::Journal(string iN, char z, int bC, DATE pD) :Book(iN, z, bC) {

this->publishDate = pD;

}

Journal::Journal(const Journal &c) : Book(c) {

this->publishDate = c.publishDate;

}

Journal::~Journal() {

cout << "destructing Journal class." << endl;

}

void Journal::display() {

Book::display();

cout << "PublishDATE:" << "Year:" << this->publishDate.year << "\t"

<< "Month:" << this->publishDate.month << "\t"

<< "Day:" << this->publishDate.day << "\t";

}

int main(){

Library L("ItemName Sample");

L.display();

cout << endl;

Book B("Book Name", 'K', 1000104);

B.display();

cout << endl;

DATE D;

D.year = 2000;

D.month = 12;

D.day = 14;

Journal J("Journal Name", 'C', 4000365, D);

J.display();

cout << endl;

try{

Student S("CWZ", Male, 21118233, 30);

S.display();

}

catch(int ex)

{

if(ex == -5)

cout << "TOO MANY BOOKS";

}

}

#include <iostream>

#include <cmath>

#include <cstring>

using namespace std;

class VectorClass

{

public:

virtual void distance() = 0;

};

class D2Vector:public VectorClass

{

public:

double x,y;

D2Vector();

D2Vector(double,double); ///有参初始化

~D2Vector();

void distance(); ///虚函数欧式距离

void display(); ///数据打印

D2Vector &operator+(const D2Vector &c);///操作符重载

D2Vector &operator-(const D2Vector &c);///操作符重载

double operator\*(const D2Vector &c);///操作符重载

D2Vector &operator=(const D2Vector &c);///操作符重载

};

class D2Vector\_A:public D2Vector

{

public:

void distance(); ///一范距离打印

D2Vector\_A();

D2Vector\_A(double,double);

};

D2Vector::D2Vector()

{

this->x = this->y = 0.0;

}

D2Vector::D2Vector(double nx, double ny)

{

this->x = nx;

this->y = ny;

}

D2Vector::~D2Vector()

{

cout << "destructing D2Vector class." << endl;

}

void D2Vector::distance()

{

cout << sqrt(x\*x + y\*y);

cout << endl;

}

void D2Vector::display()

{

cout << "X:" << this->x;

cout << " Y:" << this->y;

cout << endl;

}

D2Vector &D2Vector::operator+(const D2Vector &c)

{

this->x += c.x;

this->y += c.y;

return \*this;

}

D2Vector &D2Vector::operator-(const D2Vector &c)

{

this->x -= c.x;

this->y -= c.y;

return \*this;

}

double D2Vector::operator\*(const D2Vector &c)

{

this->x \*= c.x;

this->y \*= c.y;

return this->x + this->y;

}

D2Vector &D2Vector::operator=(const D2Vector &c)

{

this->x = c.x;

this->y = c.y;

return \*this;

}

void D2Vector\_A::distance()

{

(this->x-this->y)?(cout <<this->x-this->y):(cout << this->y - this ->x);

cout << endl;

}

D2Vector\_A::D2Vector\_A(double a,double b):D2Vector(a,b)

{}

int main()

{

D2Vector d2v1 = D2Vector(0,4);

D2Vector d2v2 = D2Vector(3,0);

D2Vector d2v3 = d2v1 + d2v2;

d2v3.display();

D2Vector \*d2V4 = new D2Vector\_A(2,3);

cout << endl << endl;

d2V4->display();

cout << "一范距离：";

d2V4->distance();

d2V4 = &d2v1;

d2V4->display();

cout << "欧式距离：" ;

d2V4->distance();

}







