

**面向对象程序课程设计**

学 号：

班 号：

学生姓名：

指导教师： 左泽均

**中国地质大学信息工程学院**

**2015年6月**

**实习题目一：根据坐标求球面距离**

【需求规格说明】

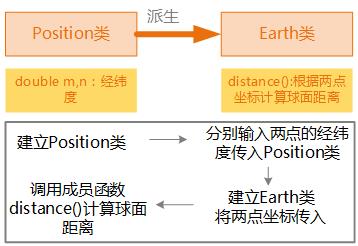
地球的平均半径为 6371 千米，已知地球上两个城市 A、B 的经度和纬度，编程序求出这 两个城市之间的球面距离（一般来说，球面上任意两点 A 和 B 都可以与球心确定唯一的大圆，而 在大圆上连接这两点的较短的一条弧的长度就是球面距离）。要求：定义坐标点类，地球类。

【程序设计】

（1）**设计思想**：

建立坐标点类Position和地球类Earth，Position作为Earth的基类进行公有继承，输入坐标调用距离计算公式C = sin(LatA)\*sin(LatB) + cos(LatA)\*cos(LatB)\*cos(MLonA-MLonB)计算两点间的距离。

（2）**设计表示**：



【调试报告】

东经和北纬用正数表示，西经和南纬用负数表示。

起初求球面距离采用r2=x2+y2公式计算，结果不准确，后来采用了经纬度和球面距离转换的公式，在求距离过程中要注意数据类型的转换。

【附录】

#include<iostream>

#include<cmath>

#include"string.h"

using namespace std;

#define R long(6371)

#define PI double(3.1415)

#define K double(2\*PI\*R/360)

class Position

{

public:

double m; //经度

double n; //纬度

Position(){};

Position(double a,double b)

{

m = a;

n = b;

}

void transform() //把小数转化为度分秒

{

int x, y, z;

int a, b, c;

x = (int)m;

y = (int)((m - x) \* 60);

z = (int)(((m - x) \* 60 - y) \* 60);

cout << "经度" << x << "°" << y << "’" << z << "”" << endl;

a = (int)n;

b = (int)((n - a) \* 60);

c = (int)(((n - a) \* 60 - b) \* 60);

cout << "纬度" << a << "°" << b << "’" << c << "”" << endl;

};

};

class Earth :public Position //计算球面距离

{

public:

Earth(){ cout << endl; }

void distance(Position& p1, Position p2)

{

double d = R\*acos(cos(p1.n /180\* PI)\*cos(p2.n /180\* PI)\*cos(p1.m / 180\*PI - p2.m / 180\*PI) + sin(p1.n /180\* PI)\*sin(p2.n / 180\*PI));

cout << "球面距离为" << d <<"千米"<<endl;

}

};

int main()

{

cout << "建议测试数据：(北京) 116 40；(武汉)：114 30" << endl;

double a,b,c,d;

cout << "请输入a城市坐标：";

cin >> a;

cin >> b;

cout << "请输入b城市坐标：";

cin >> c;

cin >> d;

Position P1(a, b), P2(c, d);

cout << "a城市经纬度为：\n";

P1.transform();

cout << "b城市经纬度为：\n";

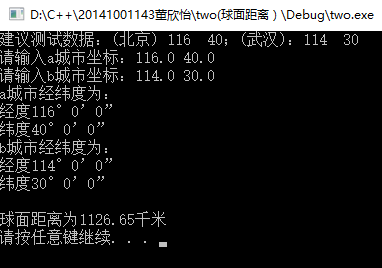
P2.transform();

Earth E1;

E1.distance(P1, P2); //计算球面距离

system("pause");

}



**实习题目二 ：省市GDP划分**

【需求规格说明】

我国每年都会发布各省市的生产总值（GDP）数据。试对全国省市进行等级划分，要求： （1） 分成 3 个等级；

（2） 每一等级的所有省市 GDP 之和基本相等；

（3） 第一个等级中任意省市 GDP 大于第二等级中任意省市 GDP；

（4） 第二个等级中任意省市 GDP 大于第三等级中任意省市 GDP；

（5） 输出每个等级中的省市数量及名称。

【程序设计】

1. **设计思想**：

将省市GDP数据写在GDP.txt文件中,建立GDPdevide类，在构造函数中将文件读入结构体数组，再调用成员函数Sequance()进行降序排序。调用成员函数devide()进行分组。

（2）**设计表示**：



【调试报告】

存取结构体数组时出现异常，经调试是由于动态数组释放多余内存后访问溢出，并缺少重载函数。在调整访问地址后恢复正常。

【附录】

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GDPdevide.h\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

#include<fstream>

#include<string>

#include<iomanip>

const int N = 40;

typedef struct

{

char city[12];

double amount;

}GDP;

class GDPdevide

{

int i = 0;

GDP gdp[N];

public:

GDPdevide(); //将文件读入结构体数组

void Display(); //数据显示

void Sequence(); //降序排序

void devide();

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GDPdevide.cpp\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include"GDPdevide.h"

#include<iostream>

#include<fstream>

#include<string>

#include<iomanip>

using namespace std;

GDPdevide::GDPdevide() //将文件读入结构体数组

{

ifstream myfiles;

myfiles.open("省市GDP.txt");

if (!myfiles.is\_open())

{

cout << "Can't open files!" << endl;

}

else

{

while (!myfiles.eof())

{

myfiles >> gdp[i].city >> gdp[i].amount;

i++;

}

}

myfiles.close();

cout << "文件读入测试：" << endl;

for (int j = 0; j < i-1; j++)

{

printf("%s", gdp[j].city);

cout << "\t";

cout << setprecision(7) << gdp[j].amount << endl;

}

cout << "共有" << i << "个城市" << endl;

}

void GDPdevide::Display()

{

cout << "文件读入测试：" << endl;

for (int j = 0; j < i-1; j++)

{

printf("%s", gdp[j].city);

cout << "\t";

cout << setprecision(7) << gdp[j].amount << endl;

}

}

void GDPdevide::Sequence() //选择法降序排序

{

GDP GDPtemp;

int j, k,t;

for (j = 0; j <i-1; j++)

{

for (k = j+1; k <i-1; k++)

{

if (gdp[k].amount > gdp[k - 1].amount)

{

t = k;

}

GDPtemp=gdp[j];

gdp[j] = gdp[t];

gdp[t] = GDPtemp;

}

}

}

void GDPdevide::devide() //分组

{

double amount = 0;

for (int j = 0; j < i-1; j++)

{

amount += gdp[j].amount;

}

cout << "GDP总值为：" << amount << endl<<endl;

amount = amount / 3;

int A = 0;

double amount1 = 0;

for (int j = 0; j < i-1; j++)

{

if (amount1 < amount)

{

amount1 += gdp[j].amount;

A++;

}

}

cout << "第一等级城市有" << A << "个。" << endl;

cout << "分别是：" << endl;

for (int j = 0; j < A; j++)

{

cout << gdp[j].city << "\t";

}

cout << endl;

cout << "GDP总值为" << setprecision(8) << amount1 << endl << endl;

int B = A;

double amount2 = 0;

for (int j = A; j < i-1; j++)

{

if (amount2 < amount)

{

amount2 += gdp[j].amount;

B++;

}

}

cout << "第二等级城市有" << B - A << "个。" << endl;

cout << "分别是：" << endl;

for (int j = A; j < B; j++)

{

cout << gdp[j].city << "\t";

}

cout << endl;

cout << setprecision(8) << "GDP总值为" << amount2 << endl << endl;

double amount3 = 0;

for (int j = B; j < i-1; j++)

{

amount3 += gdp[j].amount;

}

cout << "第三等级城市有" << i - B << "个。" << endl;

cout << "分别是：" << endl;

for (int j = B; j < i-1; j++)

{

cout << gdp[j].city << "\t";

}

cout << endl;

cout << setprecision(8) << "GDP总值为" << amount3 << endl << endl;

system("pause");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*main.cpp\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

#include<fstream>

#include<string>

#include<iomanip>

#include"GDPdevide.h"

using namespace std;

int main()

{

GDPdevide g;

g.Sequence(); //选择法降序

g.Display();

g.devide();

}

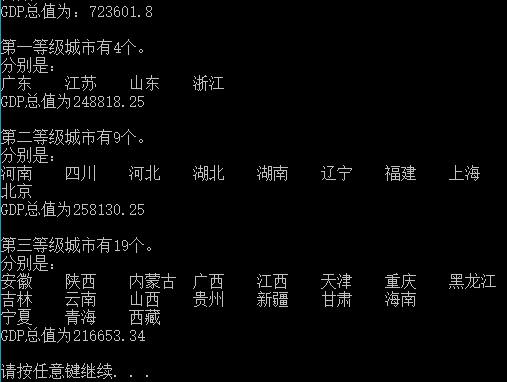
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*读入情况\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*排序显示\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*分组与统计\*\*\*\*\*\*\*\*\*\*\*\*\*\*/



**实习题目三 ：计算点到折线的距离**

【需求规格说明】

点到折线的距离定义为点到折线中每一条线段的距离中的最短距离，而点到线段的距离可 能是点到线段某个端点的距离（当垂足不在线段内部），也可能是点到垂足的距离（当垂足在线段 内部）。要求定义点类 CPoint 和折线类 CLine，并在折线类上实现距离计算成员函数。

【程序设计】

（1）**设计思想**：

建立一个point类用于存取点坐标，并计算出其相应解析式，由Point类派生出折线类CLine，Point类数组用于构成折线。在CLine类中可以计算定点到折线的距离。到折线的距离又分为两种情况讨论：点在直线上的投影在折线内还是折线外。比较点到各线段的距离得出点到线段的最短距离。

（2）**设计表示**：



【调试报告】

当直线平行于y轴，即直线的斜率无穷大时会出错，解决办法是另分出一种情况讨论。

【附录】

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*point.h\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#pragma once

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

class Point

{

public:

int x;

int y;

Point(int xx=0, int yy=0)

{

x = xx;

y = yy;

}

Point(Point& p)

{

x = p.x;

y = p.y;

}

virtual void GetPoint();

void DisplayPoint();

friend class Line;

};

class Line

{

double A;

double B;

double C;

int x1, x2, y1, y2;

public:

double d = 0;

Line();

Line(Line& L)

{

A = L.A;

B = L.B;

C = L.C;

x1 = L.x1;

x2 = L.x2;

y1 = L.y1;

y2 = L.y2;

d = L.d;

}

void GetLine(Point p1, Point p2);

void DisplayLine();

double GetDistance(Point P);

void Judge();

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*point.cpp\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "Point.h"

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

void Point::GetPoint()

{

cout << "请输入点坐标:" << endl;

int xx, yy;

cin >> xx >> yy;

x = xx;

y = yy;

}

void Point::DisplayPoint()

{

cout << "点坐标为：" << x << "," << y << endl;

}

Line::Line()

{

A = 0;

B = 0;

C = 0;

x1 = x2 = y1 = y2 = 0;

d = 0;

}

void Line::GetLine(Point p1, Point p2) //由两点生成线段

{

if (p1.x > p2.x)

{

Point p3;

p3 = p1;

p1 = p2;

p2 = p3;

}

x1 = p1.x;

x2 = p2.x;

y1 = p1.y;

y2 = p2.y;

if (x1 == x2) //形如x=3类的直线方程

{

B = 0;

A = 1;

C = -x1;

}

else

{

A = double(y2 - y1) / double(x2 - x1);

B = -1;

C = -A\*double(x1)-B\*double(y1) ;

}

}

void Line::DisplayLine() //显示线段的解析式

{

cout << "线的解析式：" << setprecision(4)<< A << "x+" << B<<"y+"<<setprecision(4) << C<< "=0" <<"("<<x1<<"≤x≤"<<x2<<")"<< endl;

}

double Line::GetDistance(Point P) //计算点到线段的距离

{

double r;

double ap, bp, cp;

if (B == 0)

{

cp = x1 - P.x;

d = cp;

cout << "点到线段的距离是：" << cp << endl;

return cp;

}

else

{

r = ((P.x - x1)\*(x2 - x1)+(P.y - y1)\*(y2 - y1)) / ((x2-x1)\*(x2-x1));

cout << "投影参数r=" << r << endl;

ap = sqrt((P.y - y1)\*(P.y - y1) + ((P.x - x1)\*(P.x - x1)));

bp = sqrt((P.y - y2)\*(P.y - y2) + ((P.x - x2)\*(P.x - x2)));

cp = abs((A\*double(P.x) + B\*double(P.y) + C) / sqrt(A\*A + B\*B));

cout << "点到线段的距离是：";

if (r < 0)

{

d = ap;

cout << ap << endl;

return ap;

}

else if (r > 1)

{

d = bp;

cout << bp << endl;

return bp;

}

else

{

d = cp;

cout << cp << endl;

return cp;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*main.cpp\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

#include"Point.h"

#include<iomanip>

#include<cmath>

using namespace std;

int n = 10;

int main()

{

cout << "测试数据：定点：0 0" << endl << "折线数：3" << endl << "坐标点依次为：" << endl << "0 1" << endl << "1 0" << endl << "4 4" << endl << "4 0" << endl << endl << endl;

Point P0;

P0.GetPoint();

P0.DisplayPoint();

cout << "输入折线的段数：";

cin >> n;

Line L[10]; //线段类数组构成折线

Point P[11]; //折线端点集

cout << "依次输入折线类点的坐标：" << endl;

for (int i = 0; i < n + 1; i++)

{

P[i].GetPoint();

}

for (int i = 0; i < n; i++)

{

L[i].GetLine(P[i], P[i + 1]);

}

for (int i = 0; i < n; i++)

{

L[i].DisplayLine();

L[i].GetDistance(P0);

}

cout << "所以点到折线的距离为：";

double distance=0;

if (n > 1)

{

distance = L[0].d;

for (int i = 0; i < n ; i++)

{

if (L[i].d < distance)

{

distance = L[i].d;

}

}

}

else

{

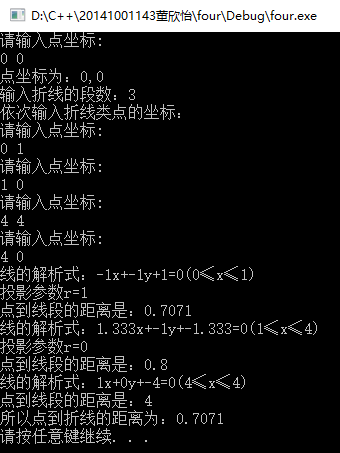
distance = L[0].d;

}

cout << distance << endl;

system("pause");

}



**实习题目四 ：抽象类shape**

【需求规格说明】

编写一个抽象基类 Shape，为其设计合适的数据成员和接口，提供对基本图元的名称、位 置、周长和面积等属性的获取、设置及计算功能；并从 Shape 派生出 Rectangle（矩形）和 Circle 类，设计合适的数据成员和接口，要求覆盖所有的纯虚函数。

【程序设计】

（1）**设计思想**：

通过抽象基类Shape定义名称，位置，周长面积等属性的纯虚函数，从Shape派生出矩形类和圆形类，分别实现从基类继承的纯虚函数。

（2）**设计表示**：

|  |
| --- |
| Point //坐标点类 |
| +x:double //点的坐标  +x:double |
| +Point(double xx, double yy)  +Point()//构造函数 |

|  |
| --- |
| *Shape* |
| +position:Point\*//存储图形的各个顶点的位置  +n:int//图形的顶点个数 |
| *<<const,abstract>>*+ShapeName():string //图形名称  *<<const,abstract>>*+ Position():Point\* //图形位置  *<<const,abstract>>*+GetPerimeter():double //图形周长  *<<const,abstract>>*+GetArea():double //图形面积  *<<const,abstract>>*+ShowPerimeter():void //图形周长显示  *<<const,abstract>>*+ShowArrea():void //图形面积显示  <<virtual>>+ShowPosition():void //图形位置显示 |

|  |  |
| --- | --- |
| Rectangle | Circle |
| -width:double //长  -length:double //宽  -lt:Point //左上角顶点  -rd:Point //右下角顶点 | -radius:double //半径  -O:Point //圆心 |
| <<const>>+ShapeName():string //图形名称  <<const>>+ Position():Point\* //图形位置  <<const>>+GetPerimeter():double //图形周长  <<const>>+GetArea():double //图形面积  <<const>>+ShowPerimeter():void //图形周长显示  <<const>>+ShowArrea():void //图形面积显示  +Rectangle(Point \_lt, Point \_rd)//构造函数 | <<const>>+ShapeName():string //图形名称  <<const>>+ Position():Point\* //图形位置  <<const>>+GetPerimeter():double //图形周长  <<const>>+GetArea():double //图形面积  <<const>>+ShowPerimeter():void //图形周长显示  <<const>>+ShowArrea():void //图形面积显示  +Circle(Point o, double r) //构造函数 |

【调试报告】

调试时Position函数的返回值出现问题，最终改为指针解决问题。

【附录】

#include<iostream>

#include<string>

#include<cmath>

using namespace std;

const double Pi = 3.14159;

class Point

{//坐标点类

public:

double x, y;

Point(double xx, double yy) :x(xx), y(yy){}

Point(){}

};

class Shape{ //基类

public:

virtual string ShapeName() = 0;

virtual Point\* Position() = 0;

virtual double GetPerimeter() = 0;

virtual double GetArea() = 0;

virtual void ShowPerimeter() = 0;

virtual void ShowArrea() = 0;

virtual void ShowPosition();

Point \*position;

int n;

};

void Shape::ShowPosition()

{//显示位置函数

this->Position();

for (int i = 0; i < n; i++)

cout << "Point " << i << ":" << "(" << position[i].x << "," << position[i].y << ")" << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Rectangle\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Rectangle :public Shape{

public:

Rectangle(Point \_lt, Point \_rd) :lt(\_lt), rd(\_rd){

length = rd.x - lt.x;

width = lt.y - rd.y;

position = new Point[4];

n = 4;

}

string ShapeName(){ return "Rectangle"; }

Point\* Position();

double GetPerimeter();

double GetArea();

void ShowPerimeter(){ cout << "The perimeter of rectangle is:" << GetPerimeter() << endl; }

void ShowArrea(){ cout << "The area of rectangle is:" << GetArea() << endl; }

//void ShowPosition();

private:

Point lt, rd;

double width, length;

};

Point\* Rectangle::Position(){

position[0] = lt;

position[3] = rd;

position[1] = Point(lt.x, rd.y);

position[2] = Point(rd.x, lt.y);

return position;

}

double Rectangle::GetPerimeter(){

return 2 \* (width+length);

}

double Rectangle::GetArea(){

return width\*length;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Circle\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Circle :public Shape{

public:

Circle(Point o, double r) :O(o), radius(r){

position = new Point[1];

n = 1;

}

string ShapeName(){ return "Circle"; }

Point\* Position();

double GetPerimeter();

double GetArea();

void ShowPerimeter(){ cout << "The perimeter of circle is:" << GetPerimeter() << endl; }

void ShowArrea(){ cout << "The area of circle is:" << GetArea() << endl; }

//void ShowPosition();

private:

Point O;

double radius;

};

double Circle::GetPerimeter(){

return 2 \* Pi\*radius;

}

double Circle::GetArea(){

return Pi\*radius\*radius;

}

Point\* Circle::Position(){

position[0] = O;

return position;

}

int main()

{

Rectangle R(Point(0, 4), Point(4, 0));

Circle C(Point(0, 0), 4);

cout << R.ShapeName() << ":" << endl;

R.ShowPerimeter();

R.ShowArrea();

R.ShowPosition();

cout << C.ShapeName() << ":" << endl;

C.ShowPerimeter();

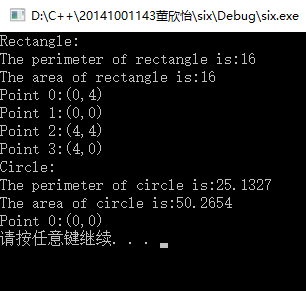
C.ShowArrea();

C.ShowPosition();

system("pause");

return 0;

}



**实习题目五 ：复数类与运算符重载**

【需求规格说明】

定义复数的类 Complex 并测试其功能：

(1) 复数由实部、虚部两个部分组成，在类的定义中应包含保存这两部分信息的内容。

(2) 在类中定义构造函数，使用户能够在构造对象的同时为对象赋初值。

(3)在类中定义复数的加法、减法、乘法三个二元操作符来完成复数间的加、减、乘的功能。

(4)通过重载“<<”，使得 Complex 类的对象能够显示其自身信息。

(5)通过显示定义一个成员函数完成对象的赋值操作，使用户能够对 Complex 类的对象进行赋值。

(6)编写一段主程序（main 函数），使用户能够通过这段主程序输入复数并进行复数的计算。

【程序设计】

（1）**设计思想**：

按照要求一步一步实现函数功能。

（2）**设计表示**：

|  |
| --- |
| Complex |
| -real:double //实部  -imag:double //虚部 |
| + operator+(Complex c):Complex//+运算符的重载  +operator-(Complex c) :Complex//-运算符的重载  +operator\*(Complex c) :Complex//\*运算符的重载  +operator=(const Complex &c) :Complex&//赋值运算符的重载  <<friend>>+operator<<(ostream &out, const Complex &c):ostream&//运算符<<的重载 |

【调试报告】

由于ostream为C++标准类库，无法向其中添加成员函数，将重载<<运算符定义为ostream的友元函数后解决问题。

【附录】

#include<iostream>

using namespace std;

class Complex

{//复数类，复数的实部和虚部

double real, imag;

public:Complex(double r, double i)

{

real = r;

imag = i;

}

Complex operator+(Complex c);

Complex operator-(Complex c);

Complex operator\*(Complex c);

Complex& operator=(const Complex &c);

friend ostream & operator<<(ostream &out, const Complex &c);

};

Complex Complex::operator+(Complex c)

{

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

}

Complex Complex::operator-(Complex c)

{

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

}

Complex Complex::operator\*(Complex c)

{

return Complex(real \*c.real - imag\*c.imag, imag\*c.real + real\*c.imag);

}

ostream & operator<<(ostream &out, const Complex &c)

{//重载<<,定义为类复数类的友元函数

return out << c.real << "+" << c.imag << "i" << endl;

}

Complex& Complex::operator=(const Complex &c){//赋值函数，重载"="

real = c.real;

imag = c.imag;

return \*this;

}

int main()

{

int a, b, c, d;

cout << "输入c1:";

cin >> a >> b;

cout << "输入c2:";

cin>> c >> d;

Complex c1(a, b);

Complex c2(c, d);

cout << "c1=" << c1 << endl;

cout << "c2=" << c2 << endl;

cout << "c2-c1=" << c2 - c1 << endl;

cout << "c2+c1=" << c2 + c1 << endl;

cout << "c2\*c1=" << c2 \* c1 << endl;

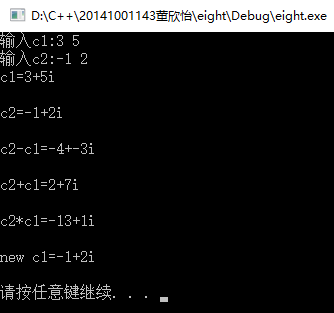
c1 = c2;

cout << "new c1=" << c1 << endl;

system("pause");

return 0;

}



**实习题目六 ：字符串类与运算符重载**

【需求规格说明】

编写一个字符串类，要求该类可以实现以下功能：

（1）取字符串长度 GetLength，判断字符串是否为空 IsEmpty，取特定字符在字符串中位置 GetAt。

（2）实现+和=的重载。

（3）实现插入一字符（或子串）Insert、删除一字符（或子串）Delete、替换一字符（或子串） Replace。

【程序设计】

（1）**设计思想**：

用动态字符数组来存储自定义的String类的对象，再通过运算符重载实现题目要求的字符串的功能。

（2）**设计表示**：

|  |
| --- |
| String |
| -Size: int//字符串长度  -\*str :char //字符指针指向动态内存 |
| +String()://构造函数  +String(String &s)://复制构造函数  +~String()://析构函数  +GetString()://获取字符串  + GetLength()://获得字符串的长度  + IsEmpty()://判断字符串是否为空  + GetAt()://取字符串某一字符  + operator+(const String &s1): //添加字符串  + operator=(const String &s1): //赋值运算  + Insert()://插入字符串  + Delete()://删除字符串  + Replace()://替换字符串 |

【调试报告】

开始时程序在结束时会出错，原因是析构函数会在返回this指针之前进行，故删去析构函数后恢复正常。

【附录】

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*string.h\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

#include<string>

const int n = 40;

using namespace std;

class String

{

char\* str = new char[n];

char\* str0 = new char[n];

int size = 0;

public:

String()

{

for (int i = 0; i < n; i++)str[i] = ' ';

}

String(String &s)

{

for (int i = 0; i < size; i++)

{

s.str[i] = str[i]; s.str0[i] = str0[i];

}

}

// ~String(){ delete[] str; delete[] str0; } //引起this指针出错 故注释掉

void GetString();

void GetLength();

void IsEmpty();

void GetAt();

void Insert();

void Delete();

void Replace();

friend String operator+(const String& s1, const String& s2);

String& operator=(const String& s1);

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*string.cpp\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include"String.h"

#include<iostream>

#include<string>

using namespace std;

void String::GetString()

{

cin >> str;

size = strlen(str);

}

void String::GetLength()

{

size = strlen(str);

cout << "Length is " << size << "." << endl;

}

void String::IsEmpty()

{

size = strlen(str);

if (size == 0)

{

cout << "string is empty!" << endl;

}

else

{

cout << "string is not empty!" << endl;

}

}

void String::GetAt()

{

cout << "输入要查找的字母：";

char a;

cin >> a;

int i;

cout << "字母的位置为：";

for (i = 0; i < size; i++)

{

if (a == str[i])

cout << i + 1 << " ";

}

cout << endl;

}

void String::Insert()

{

cout << "插在第几位：";

int m;

cin >> m;

if (m > size + 1)

{

cout << "请输入正确数字！";

}

else

{

cout << "输入s1要插入的字符串：";

char\* str1 = new char[n];

cin >> str1;

int size1 = 0;

size1 = strlen(str1);

for (int i = 0; i < m - 1; i++)

{

str0[i] = str[i];

}

for (int i = m - 1; i < m + size1 - 1; i++)

{

str0[i] = str1[i - m + 1];

}

for (int i = m + size1 - 1; i <size + size1; i++)

{

str0[i] = str[i - size1];

}

for (int i = 0; i < size + size1; i++)

{

cout << str0[i] ;

}

cout << endl;

delete str1;

}

}

void String::Delete()

{

cout << "输入s1要删除的子串位置：";

int a = 0;

int b = 0;

if (b > size)

{

cout << "请输入正确数字！";

}

else

{

cin >> a >> b;

char\* str1 = new char[n];

for (int i = 0; i < a - 1; i++)

{

str1[i] = str[i];

}

for (int i = a - 1; i < size - b + a; i++)

{

str1[i] = str[i + b - a + 1];

}

cout << str1 << endl;

delete str1;

}

}

void String::Replace()

{

cout << "输入s1要取代的位置：";

int a, b;

cin >> a >> b;

cout << "输入s1要取代的字符串：";

char\* str1 = new char[n];

cin >> str1;

int size1 = strlen(str1);

if (size1 != b - a + 1)

{

cout << "请输入正确长度字符串：";

}

else

{

for (int i = a - 1; i < b; i++)

{

str[i] = str1[i - a + 1];

}

cout << str << endl;

delete str1;

}

}

String operator+(const String& s1, const String& s2)

{

String t;

for (int i = 0; i < s1.size; i++)

{

t.str[i] = s1.str[i];

}

for (int i = s1.size; i < s1.size + s2.size; i++)

{

t.str[i] = s2.str[i - s1.size];

}

cout << "s1与s2拼接后的字符串为：";

for (int i = 0; i < s1.size + s2.size; i++)

{

cout <<t.str[i] ;

}

cout << endl;

return t;

}

String&String::operator=(const String& s1)

{

delete []str;

size = s1.size;

char \*str = new char[size];

for (int i = 0; i < size; i++)

{

str[i] = s1.str[i];

}

cout << "将s2赋值给s1,并输出赋值结果：";

for (int i = 0; i < size; i++)

{

cout<<str[i];

}

cout << endl;

return \*this;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*main.cpp\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

#include<string>

#include"String.h"

using namespace std;

int main()

{

cout << "测试数据：abcdabcd" << endl << endl;

String st;

cout << "请输入字符串s1:";

st.GetString();

st.GetLength();

st.IsEmpty();

st.GetAt(); //查找字符

st.Insert(); //插入字符

st.Delete(); //删除字符

st.Replace(); //取代字符

String st1;

cout << "请输入字符串s2:";

st1.GetString();

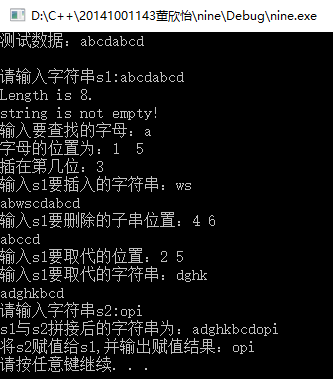
st + st1;

st = st1;

system("pause");

return 0;

}



**实习题目七 ： CSV文件的读取（MFC）**

【需求规格说明】

创建一个对话框，对话框中包含一个编辑框控件用于显示打开的CSV文件路径，三个按钮的控件功能分别是用于打开文件对话框、处理CSV记录文件和退出程序，以及一个列表框控件用于显示处理前后CSV文件中的记录数据。其中CSV表数据的分析处理流程和第七章相同，即删除读入数据中的冗余部分并计算入学年龄后将数据保存到一个新的CSV文件。

【程序设计】

（1）**设计思想**：

CSV文件用于存储数据，CtenDlg类映射Dialog中的消息函数，定义CDate类用于处理文件中日期的格式转化与计算。

（2）**设计表示**：



【调试报告】

起初读入CSV文件后发现日期相减后得出的结果不对，发现是缺少compare函数，在使用compare函数之前还要把字符串类型的日期转化为int类型才能计算。

【附录】

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CtenDlg.cpp部分代码\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool CtenDlg::GetCSVInfo(int &nRowNum, int &nColNum)

{

ifstream inFile(m\_strFileDirectory);

string strLine;

string strItem;

int lineNum = 0;

int itemNum = 0;

if (!inFile)return false;

while (getline(inFile, strLine))

{

istringstream linestream(strLine);

if (0 == lineNum)

{

while (getline(linestream, strItem, ','))

itemNum++;

nColNum = itemNum;

}

lineNum++;

}

nRowNum = lineNum;

return true;

}

bool CtenDlg::ReadCSV(string \*\*data)

{

ifstream inFile(m\_strFileDirectory);

string strLine;

string strItem;

int lineNum = 0;

int itemNum = 0;

if (!inFile)return false;

while (getline(inFile, strLine))

{

istringstream linestream(strLine);

itemNum = 0;

while (getline(linestream, strItem, ','))

{

data[lineNum][itemNum] = strItem;

itemNum++;

}

lineNum++;

}

return true;

}

bool CtenDlg::SaveCSV(string \*\*data, int nRowNum, int nColNum)

{

ofstream outFile(m\_strFileDirectoryOut);

int i, k;

if (!outFile)return false;

for (i = 0; i < nRowNum;i++)

{

for (k = 0; k < nColNum - 1; k++)

{

outFile << strRecordsOut[i][k] << ",";

}

outFile << strRecordsOut[i][k] << endl;

}

return true;

}

void CtenDlg::AllocateMemories()

{

strRecords = new string\*[nMaxRow];

strRecordsOut = new string\*[nMaxRow];

for (int i = 0; i < nMaxRow; i++)

{

strRecords[i] = new string[nMaxCol];

strRecordsOut[i] = new string[nMaxCol+1];

}

}

void CtenDlg::ReleaseMemories()

{

for (int i = 0; i < nMaxRow; i++)

{

delete[]strRecords[i];

delete[]strRecordsOut[i];

}

delete[]strRecords;

delete[]strRecordsOut;

}

void CtenDlg::OnBnClickedButtonFileOpen()

{

// TODO: 在此添加控件通知处理程序代码

string strFilter("CSV file(\*.CSV)|\*.CSV||");

int i;

CFileDialog FileDlg(TRUE, NULL, NULL, OFN\_HIDEREADONLY, strFilter.c\_str());

if (FileDlg.DoModal() != IDOK)return;

m\_strFileDirectory = FileDlg.GetPathName();

UpdateData(FALSE);

if (m\_bExistRecords)

{

m\_listRecords.DeleteAllItems();

ReleaseMemories();

}

if (!GetCSVInfo(nMaxRow, nMaxCol))return;

AllocateMemories();

if (!ReadCSV(strRecords))return;

m\_bExistRecords=true;

for (i = 0; i < nMaxRow; i++)

{

m\_listRecords.InsertItem(i, strRecords[i][0].c\_str());

for (int j = 1; j < nMaxCol; j++)

m\_listRecords.SetItemText(i, j, strRecords[i][j].c\_str());

}

}

void CtenDlg::OnBnClickedOk()

{

// TODO: 在此添加控件通知处理程序代码

//CDialogEx::OnOK();

for (int m = 0; m < nMaxRow; m++)

{

for (int n = 0; n < nMaxCol; n++)

{

strRecordsOut[m][n] = strRecords[m][n];

}

}

string strBirthday, strEnrollmentDate;

CDate cBirthday, cEnrollmentDate;

bool bRepeat;

int i, j, k;

int nPos = -1;

int nEnrollmentDays;

if (!m\_bExistRecords)

{

MessageBox("Please open a CSV file!","Information",MB\_OK);

return;

}

nActualRowNum = 0;

for (i = 0; i < nMaxRow; i++)

{

bRepeat = false;

for (j = 0; j < nActualRowNum; j++)

if (strRecords[i][0] == strRecordsOut[j][0])bRepeat = true;

if (!bRepeat)

{

for (k = 0; k < nMaxCol; k++)

strRecordsOut[nActualRowNum][k] = strRecords[i][k];

nActualRowNum++;

}

}

for (i = 0; i < nActualRowNum; i++)

{

strBirthday = strRecordsOut[i][2];

cBirthday.Set(strBirthday);

strEnrollmentDate = strRecordsOut[i][3];

cEnrollmentDate.Set(strEnrollmentDate);

nEnrollmentDays = cEnrollmentDate.ComplexCompare(cBirthday);

ostringstream streamEnrollmentAge;

streamEnrollmentAge << setprecision(4) << nEnrollmentDays / 365.25;

strRecordsOut[i][nMaxCol] = streamEnrollmentAge.str();

}

nPos = m\_strFileDirectory.Find(".");

if (-1 == nPos)return;

m\_strFileDirectoryOut = m\_strFileDirectory;

m\_strFileDirectoryOut.Insert(nPos, "\_modified");

if (!SaveCSV(strRecordsOut, nActualRowNum, nMaxCol + 1))

{

MessageBox("Cannot save the records to a CSV file!", "Information", MB\_OK);

return;

}

else

{

m\_listRecords.DeleteAllItems();

for (i = 0; i < nActualRowNum; i++)

{

m\_listRecords.InsertItem(i, strRecordsOut[i][0].c\_str());

for (j = 1; j < nMaxCol + 1; j++)

m\_listRecords.SetItemText(i, j, strRecordsOut[i][j].c\_str());

}

ShellExecute(NULL, "open", m\_strFileDirectoryOut, NULL, NULL, SW\_SHOW);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Date.h\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#pragma once

#include<string>

using namespace std;

class CDate

{

string date;

int nYear;

int nMonth;

int nDay;

public:

CDate();

CDate(CDate& D)

{

date = D.date;

nYear = D.nYear;

nMonth = D.nMonth;

nDay = D.nDay;

}

string Set(string s);

int ComplexCompare(CDate D1);

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*date.cpp\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "stdafx.h"

#include "Date.h"

CDate::CDate()

{

}

string CDate::Set(string s)

{

date = s;

string y = "/";

int iPosition1 = date.find(y, 0);//从0位置开始寻找字符串y，返回的是找到的字母的下标

//如果没找到，将返回－1

string str1 = date.substr(0, 4);

nYear = atoi(str1.c\_str());

int iPosition2 = date.find(y, 5);//从5位置开始寻找字符串y，返回的是找到的字母的下标

//如果没找到，将返回－1

string str2 = date.substr(5, iPosition2 - 5);

nMonth = atoi(str2.c\_str());

string str3 = date.substr(iPosition2 + 1);

nDay = atoi(str3.c\_str());

return date;

}

int CDate::ComplexCompare(CDate D1)

{

int i;

int month\_day[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

int year1 = nYear - 1; // 因为欲求距离1年1月1日的距离

int days1 = year1 \* 365 + nYear / 4 - nYear / 100 + nYear / 400; //求得之前闰年的数量并在天数上进行想加

if (nYear % 4 == 0 && nYear % 100 != 0 || nYear % 400 == 0) month\_day[1]++; //当前年为闰年，二月加 1

for (i = 0; i<nMonth - 1; i++)

days1 += month\_day[i];

days1 += nDay - 1;

int month\_day1[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

int year2 = D1.nYear - 1; // 因为欲求距离1年1月1日的距离

int days2 = year2 \* 365 + D1.nYear / 4 - D1.nYear / 100 + D1.nYear / 400; //求得之前闰年的数量并在天数上进行想加

if (nYear % 4 == 0 && nYear % 100 != 0 || nYear % 400 == 0) month\_day1[1]++; //当前年为闰年，二月加 1

for (i = 0; i<D1.nMonth - 1; i++)

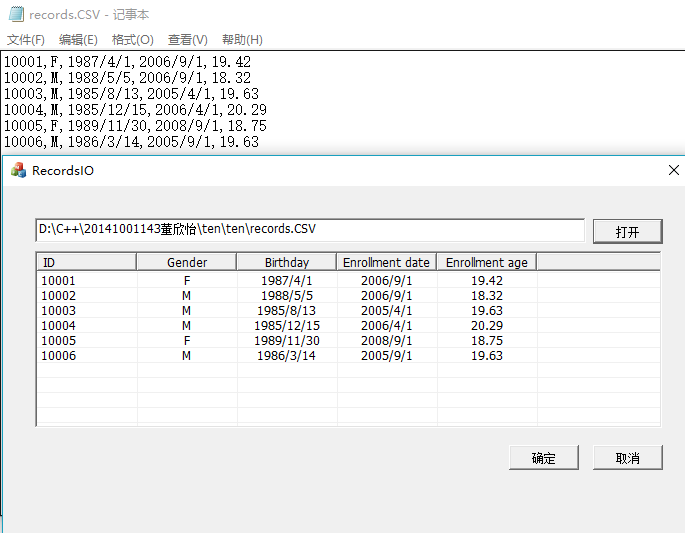
days2 += month\_day[i];

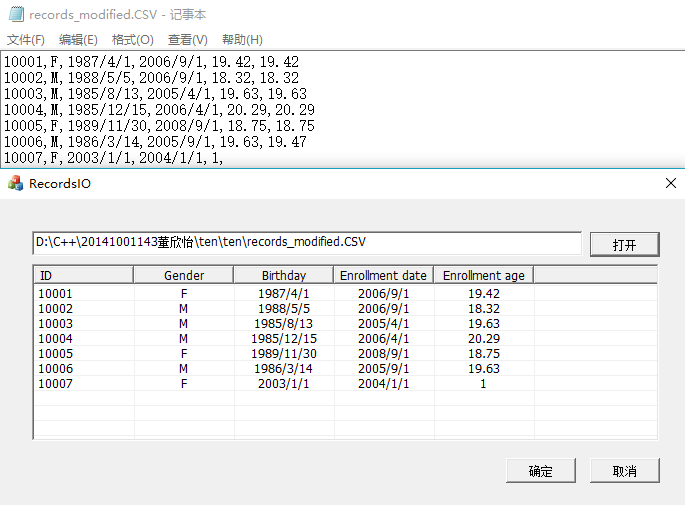
days2 += D1.nDay - 1;

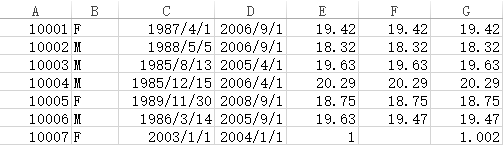
int days = days1 - days2;//两个距离相减得到相距天数

return days;

}







**总结**

通过这次课程设计实习，主要收获是加深了对各种运算符重载以及继承派生的知识的运用，还有取系统时间和字符转化的方法，复习了I/O流的使用，最大的收获是了解到了各种MFC控件的用法和单文档结构的使用，VS和VC不仅在功能上有很大的不同，在一些函数的实现上也有很大的不同，在使用的时候会出现一些问题。

这次课程设计也学会了基本的MFC的编程应用， MFC有点难难，但比起控制台有很大发挥的空间，所以要好好学习。