**南京航空航天大学**

**《面向对象程序设计语言》课程设计报告**

**空间数据分析系统**

**学号：161740229**

**姓名：高远**

**日期：2018.5.13**

**任务1**

实现类 Point, 类 Rectangle、类 Line 的成员函数，包括构造函数、拷贝构造函数、 赋值函数、Area()函数、operator<<重载函数（采用 cout 能输出 Point、Rect、Line）、静态成员函数 static int GetCount(){return count;} 要求: 采用多文件技术 Point.h, Rectangle.h, Line.h, main.cpp

**Shape.h：**

Shape(int ob = 1,std::string D = " "):obj\_id(ob),des(D){};//构造函数

Shape(const Shape& in):obj\_id(in.obj\_id),des(in.des){};//拷贝构造函数

void put\_in(int k,std::string L){obj\_id = k; des = L; }

int get\_OB(){return obj\_id;};

std::string get\_des(){return des;};

virtual float Area() = 0 ;//纯虚函数

**Point.h:**

Point(int ob,float xx, float yy, std::string DD):x(xx),y(yy),Shape(ob,DD){count\_n++;};//构造函数

Point(const Point& in):x(in.x),y(in.y),Shape(in){};//拷贝构造函数

Point(){};//构造函数

void put\_in\_data(int b,float m,float n,std::string F){x = m; y = n; Shape::put\_in(b,F);}//赋值函数

float getX(){return x;}

float getY(){return y;}

void Change\_shape(int ob,std::string str){Shape::put\_in(ob,str);}//对基类赋值

static int Get\_count(){return count\_n;}

virtual float Area(){return 0;}

ostream& operator<<(ostream &o, Point m)//重载point的<<

{

o<<m.get\_OB()<<" "<<m.getX()<<" "<<m.getY()<<" "<<m.get\_des()<<endl;

}

**Line.h:**

Line(int ob,float p1\_x,float p1\_y,float p2\_x,float p2\_y,double LL,std::string D):len(LL),Shape(ob,D){

p1.put\_in\_data(1,p1\_x,p1\_y," "); p2.put\_in\_data(1,p2\_x,p2\_y," "); count\_m++; };//构造函数

Line(const Line& in):p1(in.p1),p2(in.p2),Shape(in),len(in.len){};//拷贝构造函数

static int Get\_count(){return count\_m;}

virtual float Area(){

float p1\_x,p1\_y;

float p2\_x,p2\_y;

p1\_x = p1.getX();

p1\_y = p1.getY();

p2\_x = p2.getX();

p2\_y = p2.getY();

float length;

length = sqrt((p1\_x-p2\_x)\*(p1\_x-p2\_x)+(p1\_y-p2\_y)\*(p1\_y-p2\_y));

return length;

};

ostream& operator<<(ostream &o, Line in)//重载line的<<

{

o<<in.get\_OB()<<" "<<in.get\_p1x()<<" "<<in.get\_p1y()<<" "<<in.get\_p2x()<<" "<<in.get\_p2y()<<" "<<in.get\_len()<<endl;

}

**Rectangle.h:**

Rect(int ob,float lp\_x,float lp\_y,float rp\_x,float rp\_y,float area,std::string D):or\_area(area),Shape(ob,D){

lp.put\_in\_data(1,lp\_x,lp\_y," "); rp.put\_in\_data(1,rp\_x,rp\_y," "); count\_j++; };//构造函数

Rect(const Rect& in):lp(in.lp),rp(in.rp),or\_area(in.or\_area),Shape(in){};//拷贝

static int Get\_count(){return count\_j;}

virtual float Area(){

float lpx = lp.getX();

float lpy = lp.getY();

float rpx = rp.getX();

float rpy = rp.getY();

return (lpx - rpx)\*(lpy -rpy);

};

ostream& operator<<(ostream& o,Rect q)//重载rect的<<

{

o<<q.get\_OB()<<" "<<q.get\_lpx()<<" "<<q.get\_lpy()<<" "<<q.get\_rpx()<<" "<<q.get\_rpy()<<" "<<q.get\_arae()<<" "<<q.get\_des()<<endl;

}

**任务2：**

1. 从三个文件分别读入所有数据（Point 对象, Rect 对象, Line 对象） ，保存在 vector 中(利用 文件流)，通过调用 GetCount()输出 Point、Rect、Line 对象个数

Point：

vector<Point> point\_list;

ifstream q\_point;

q\_point.open("point.txt",ios::in);

char x\_point[M],y\_point[M],useless[M];

char id[S],de\_s[M];

while(!q\_point.eof()){

q\_point.getline(id,S,' ');

q\_point.getline(x\_point,M,' ');

q\_point.getline(y\_point,M,' ');

q\_point.getline(useless,M,' ');

q\_point.getline(de\_s,M);

int id\_pi = atof(id);

float x\_pi = atof(x\_point);

float y\_pi = atof(y\_point);

string desi = de\_s;

Point temp(id\_pi,x\_pi,y\_pi,desi);

point\_list.push\_back(temp);

}

int count\_point = Point::Get\_count()-1;

cout<<count\_point<<endl;

q\_point.close();

Line:

vector<Line> line\_list;

ifstream q\_Line;

q\_Line.open("line.txt",ios::in);

char a\_x\_p[M],a\_y\_p[M];

char b\_x\_p[M],b\_y\_p[M];

char Leng[M];

while(!q\_Line.eof()){

q\_Line.getline(id,S,' ');

q\_Line.getline(a\_x\_p,M,' ');

q\_Line.getline(a\_y\_p,M,' ');

q\_Line.getline(b\_x\_p,M,' ');

q\_Line.getline(b\_y\_p,M,' ');

q\_Line.getline(Leng,M);

int id\_li = atof(id);

float a\_x\_pi = atof(a\_x\_p);

float a\_y\_pi = atof(a\_y\_p);

float b\_x\_pi = atof(b\_x\_p);

float b\_y\_pi = atof(b\_y\_p);

double leni = atof(Leng);

Line temp(id\_li,a\_x\_pi,a\_y\_pi,b\_x\_pi,b\_y\_pi,leni," ");

line\_list.push\_back(temp);

}

int count\_line = Line::Get\_count()-1;

cout<<count\_line<<endl;

q\_Line.close();

Rect:

vector<Rect> rect\_list;

ifstream q\_rect;

q\_rect.open("rect.txt",ios::in);

while(!q\_rect.eof()){

q\_rect.getline(id,S,' ');

q\_rect.getline(a\_x\_p,M,' ');

q\_rect.getline(a\_y\_p,M,' ');

q\_rect.getline(b\_x\_p,M,' ');

q\_rect.getline(b\_y\_p,M,' ');

q\_rect.getline(Leng,M,' ');

q\_rect.getline(de\_s,M);

int id\_li = atof(id);

float a\_x\_pi = atof(a\_x\_p);

float a\_y\_pi = atof(a\_y\_p);

float b\_x\_pi = atof(b\_x\_p);

float b\_y\_pi = atof(b\_y\_p);

double leni = atof(Leng);

string dese = de\_s;

Rect temp(id\_li,a\_x\_pi,a\_y\_pi,b\_x\_pi,b\_y\_pi,leni,dese);

rect\_list.push\_back(temp);

}

int count\_rect = Rect::Get\_count()-1;

cout<<count\_rect<<endl;

q\_rect.close();

3. 任意给一个 Point 对象，找出所有和该对象成员变量 des(小写形式)相同的 Rect 对象(即 Point 的变量 des 和 Rect 的变量 des 相同)，输出所有的 Rect 对象到屏幕；任意给一个 Rect对象，找出所有和该对象的des相同的Point对象，输出所有的Point对象到屏幕

int num;

string to\_find;

cout<<"输入Point序号：";

cin>>num;

vector<Point>::iterator iter\_point = point\_list.begin();

vector<Rect>::iterator iter\_rect = rect\_list.begin();

for(;iter\_point != point\_list.end(); iter\_point++ )

{

Point temp\_in = \*iter\_point;

if(num == temp\_in.get\_OB())

{

to\_find = temp\_in.get\_des();

break;

}

}

if(iter\_point == point\_list.end()) cout<<"未找到"<<endl;

else

{

for(;iter\_rect != rect\_list.end(); iter\_rect++ )

{

Rect Temp = \*iter\_rect;

if(to\_find == Temp.get\_des())

{

cout<<Temp;

}

}

}

cout<<"输入Rect序号：";

cin>>num;

iter\_point = point\_list.begin();

iter\_rect = rect\_list.begin();

for(;iter\_rect != rect\_list.end(); iter\_rect++ )

{

Rect temp\_in = \*iter\_rect;

if(num == temp\_in.get\_OB())

{

to\_find = temp\_in.get\_des();

break;

}

}

if(iter\_rect == rect\_list.end()) cout<<"未找到"<<endl;

else

{

for(;iter\_point != point\_list.end(); iter\_point++ )

{

Point Temp = \*iter\_point;

if(to\_find == Temp.get\_des())

{

cout<<Temp;

}

}

}

**任务3：**遍历存储 Rect 的结构(vector 或 List)，找出面积最小和最大的 Rect。将最小和最 大的 Rect 写入到文件 Rect\_data.txt。 要求：如果使用 vector，必须使用迭代器(iterator)

iter\_rect = rect\_list.begin();

Rect in\_temp = \*iter\_rect;

float AREA\_max = in\_temp.get\_arae() ;

float AREA\_min = in\_temp.get\_arae() ;

Rect area\_max = \*iter\_rect;

Rect area\_min = \*iter\_rect;

for(;iter\_rect != rect\_list.end()-1; iter\_rect++)

{

Rect temp = \*iter\_rect;

if(temp.get\_arae() > AREA\_max)

{

AREA\_max = temp.get\_arae();

area\_max = \*iter\_rect;

}

else if(temp.get\_arae() < AREA\_min)

{

AREA\_min = temp.get\_arae();

area\_min = \*iter\_rect;

}

}

string N = area\_max.get\_des();

ofstream rect\_data;

rect\_data.open("Rect\_data.txt",ios::out);

area\_max.WriteToFile(rect\_data);

area\_min.WriteToFile(rect\_data);

遍历存储 Line 的结构(vector 或 List)，找出长度最小和最大的 Line。(类 Line 用 Area()函数返回长度)。将最小和最大的 Rect 写入到文件 Line\_data.txt

vector<Line>::iterator iter\_line = line\_list.begin();

Line l\_temp\_max = \*iter\_line;

Line l\_temp\_min = \*iter\_line;

float line\_area\_min = l\_temp\_min.Area();

float line\_area\_max = l\_temp\_max.Area();

for(;iter\_line != line\_list.end()-1; iter\_line++ )

{

Line temp = \*iter\_line;

if(line\_area\_max < temp.Area())

{

l\_temp\_max = \*iter\_line;

line\_area\_max = temp.Area();

}

else if(line\_area\_min > temp.Area())

{

l\_temp\_min = \*iter\_line;

line\_area\_min = temp.Area();

}

}

ofstream line\_data;

line\_data.open("Line\_data.txt",ios::out);

l\_temp\_max.WriteToFile(line\_data);

l\_temp\_min.WriteToFile(line\_data);

**任务4：**

1. 对类 Point 实现运算符重载 operator<

bool operator<(Point& a, Point& b)//重载point的<

{

if(a.getX()<b.getX())

return true;

else if(a.getX() == b.getX() ){

if(a.getY() < b.getY())

return true;

}

else

return false;

}

1. 对类 Point、类 Rect 实现运算符重载 operator+

Point: x1+x2, y1+y2

Rect: lp = min(lp1, lp2), rp = max(rp1, rp2)

Point operator+(Point& a, Point& b)//重载point的+

{

Point temp(0,(a.getX()+b.getX()),(a.getY()+b.getY())," sum ");

return temp;

}

Rect operator+(Rect& a,Rect& b)//重载rect的+

{

float lp1 = (a.get\_lpx()\*a.get\_lpx())+(a.get\_lpy()\*a.get\_lpy());

float lp2 = (b.get\_lpx()\*b.get\_lpx())+(b.get\_lpy()\*b.get\_lpy());

float rp1 = (a.get\_rpx()\*a.get\_rpx())+(a.get\_rpy()\*a.get\_rpy());

float rp2 = (b.get\_rpx()\*b.get\_rpx())+(b.get\_rpy()\*b.get\_rpy());

float lp\_x,lp\_y,rp\_x,rp\_y;

if((lp1 - lp2) > 0.0001){

lp\_x = b.get\_lpx();

lp\_y = b.get\_lpy();

}else if((lp2 - lp1) > 0.0001){

lp\_x = a.get\_lpx();

lp\_y = a.get\_lpy();

}else if(lp2 == lp1){

lp\_x = a.get\_lpx();

lp\_y = a.get\_lpy();

}

if((rp1 - rp2) > 0.0001){

rp\_x = b.get\_rpx();

rp\_y = b.get\_rpy();

}else if((rp2 - rp1) > 0.0001){

rp\_x = a.get\_rpx();

rp\_y = a.get\_rpy();

}else if(rp2 == rp1){

rp\_x = a.get\_rpx();

rp\_y = a.get\_rpy();

}

Rect temp(0,lp\_x,lp\_y,rp\_x,rp\_y,0," sum ");

return temp;

}

**任务5：**

类 Point 实现成员函数 Distance()，求解两个点之间的距离

float Point::Distance(Point &in)

{

return sqrt((x-in.x)\*(x-in.x)+(y-in.y)\*(y-in.y));

}

类 Rect 实现成员函数 Distance()，求解距离(最近的两点之间的距离)

float Line::Distance(Line &in)

{

int i;

float a[4],mi;

a[0] = p1.Distance(in.p1);

a[1] = p1.Distance(in.p2);

a[2] = p2.Distance(in.p1);

a[3] = p2.Distance(in.p2);

mi = a[0];

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

if(a[i] < mi)

mi = a[i];

}

return mi;

}

类 Line 实现成员函数 Distance()，求解距离(最近的两点之间的距离)

float Rect::Distance(Rect &in)

{

int i;

float a[4],mi;

a[0] = lp.Distance(in.lp);

a[1] = lp.Distance(in.rp);

a[2] = rp.Distance(in.lp);

a[3] = rp.Distance(in.rp);

mi = a[0];

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

if(a[i] < mi)

mi = a[i];

}

return mi;

}

类 Point 实现成员函数 Distance\_R()，求解 Point 对象和 Rect 对象之间的距离(最 近的两点之间的距离)；类 Point 实现成员函数 Distance\_L()，求解 Point 对象和 Line 对象之间的距离(最近的两点之间的距离)

float Point::Distance\_R\_L(Point &in,Point &im)

{

float in\_x = in.x,in\_y = in.y;

float im\_x = im.x,im\_y = im.y;

float a,b;

a = (in\_x - x)\*(in\_x - x) + (in\_y - y)\*(in\_y - y);

b = (im\_x - x)\*(im\_x - x) + (im\_y - y)\*(im\_y - y);

if(a < b) return sqrt(a);

else return sqrt(b);

}

**任务6:**

体现多态性，分别在 Point、Rect 和 Line 中实现 WriteToFile()虚函数将数据写入 到文件，格式为 Point: ID (x, y) des Rect: ID (x1, y1) (x2, y2) des Line: ID (x1, y1) (x2, y2) len

要求: 在 main 函数中调用 WriteData(Shape\* s, ofstream& out\_file) void WriteData(Shape\* s, ofstream& out\_file)

{

}

1.将所有 Point 对象的字符串变量 des 变成小写，按坐标排序后（任务 4 中的①， 利用 sort 排序，p1<p2<p3…<pn）写入到 Point2.txt；

2.将 Rect 对象按照 des(字符串排序)，写入到 Rect2.txt；

3.将 Line 对象按照 len 递增数序排序，写入到文件 Line2.txt

提示: 重载运算符 operator<，Rect 按照字符串排序，Line 按照 len 排序，运用 sort 排序

bool operator<(Line& a, Line& b)//重载line的<

{

if(a.get\_len()<b.get\_len())

return true;

else

return false;

}

bool operator<(Rect& a, Rect& b)//重载rect的<

{

if(a.get\_des()<b.get\_des())

return true;

else

return false;

}

void Point::WriteToFile(ofstream& o)

{

o<<Point::get\_OB()<<" "<<x<<" "<<y<<" "<<Point::get\_des()<<endl;

}

void Rect::WriteToFile(ofstream& o)

{

o<<Rect::get\_OB()<<" "<<Rect::get\_lpx()<<" "<<Rect::get\_lpy()<<" "<<Rect::get\_rpx()<<" "<<Rect::get\_rpy()<<" "<<Rect::get\_des()<<endl;

}

void Line::WriteToFile(ofstream& o)

{

o<<Line::get\_OB()<<" "<<Line::get\_p1x()<<" "<<Line::get\_p1y()<<" "<<Line::get\_p2x()<<" "<<Line::get\_p2y()<<" "<<Line::get\_des()<<" "<<Line::get\_len()<<endl;

}

void WriteData(Shape\* s, ofstream& out\_file)

{

s->WriteToFile(out\_file);

}

bool compare(int a,int b)

{

return a > b;

}

Main:

iter\_point = point\_list.begin();

for(;iter\_point != point\_list.end(); iter\_point++){

Point chan\_temp = \*iter\_point;

int ob = chan\_temp.get\_OB();

string change;

change = chan\_temp.get\_des();

transform(change.begin(),change.end(),change.begin(),::tolower);

chan\_temp.Change\_shape(ob,change);

\*iter\_point = chan\_temp;

}

sort(point\_list.begin(),point\_list.end()-1);

ofstream write\_to\_point;

write\_to\_point.open("Point2.txt");

for(iter\_point = point\_list.begin();iter\_point != point\_list.end()-1; iter\_point++)

{

Point temp = \*iter\_point;

WriteData(&temp,write\_to\_point);

}

sort(line\_list.begin(),line\_list.end()-1);

ofstream write\_to\_line;

write\_to\_line.open("Line2.txt");

for(iter\_line = line\_list.begin();iter\_line != line\_list.end()-1; iter\_line++)

{

Line temp = \*iter\_line;

WriteData(&temp,write\_to\_line);

}

sort(rect\_list.begin(),rect\_list.end()-1);

ofstream write\_to\_rect;

write\_to\_rect.open("Rect2.txt");

for(iter\_rect = rect\_list.begin();iter\_rect != rect\_list.end()-1; iter\_rect++)

{

Rect temp = \*iter\_rect;

WriteData(&temp,write\_to\_rect);

}