**1.**

#include <iostream>

using namespace std;

class Point

{

private:

int x;

int y;

public:

Point() { x = y = 0; };

Point(int \_x, int \_y) : x(\_x), y(\_y) {};

Point(const Point& \_Point) : x(\_Point.x), y(\_Point.y) {};

void Set(int \_new\_x, int \_new\_y) { x = \_new\_x; y = \_new\_y; }

Point& operator=(const Point& \_new\_Point) { x = \_new\_Point.x, y = \_new\_Point.y; return \*this; }

void Print() const { cout << "(" << x << ", " << y << ")"; }

~Point() { cout << "The destructor of Point "; Print(); cout << " called!" << endl; }

friend class Rectangle;

};

class Rectangle

{

private:

Point left\_bottom;

Point right\_top;

static int Abs(int x) { return x > 0 ? x : -x; }

public:

Rectangle() {}

Rectangle(int \_left, int \_bottom, int \_right, int \_top) : left\_bottom(\_left, \_bottom), right\_top(\_right, \_top) {}

Rectangle(const Point& \_left\_bottom, const Point& \_right\_top) : left\_bottom(\_left\_bottom), right\_top(\_right\_top) {}

Rectangle(const Rectangle& \_Rect) : left\_bottom(\_Rect.left\_bottom), right\_top(\_Rect.right\_top) {}

void Set(int \_new\_left, int \_new\_bottom, int \_new\_right, int \_new\_top);

void Set(const Point& \_new\_left\_bottom, const Point& \_new\_right\_top);

void Print() const;

int Area() const;

~Rectangle() { cout << "The destructor of Rectangle "; left\_bottom.Print(); cout << ", "; right\_top.Print(); cout << " called!" << endl; }

};

void Rectangle::Set(int \_new\_left, int \_new\_bottom, int \_new\_right, int \_new\_top)

{

left\_bottom.Set(\_new\_left, \_new\_bottom);

right\_top.Set(\_new\_right, \_new\_top);

}

void Rectangle::Set(const Point& \_new\_left\_bottom, const Point& \_new\_right\_top)

{

left\_bottom = \_new\_left\_bottom;

right\_top = \_new\_right\_top;

}

void Rectangle::Print() const

{

cout << "The left bottom corner: ";

left\_bottom.Print();

cout << "; The right top corner: ";

right\_top.Print();

cout << "." << endl;

}

int Rectangle::Area() const

{

return Abs(right\_top.x - left\_bottom.x) \* Abs(right\_top.y - left\_bottom.y);

}

int main()

{

Point p1(6, 6), p2(8, 9);

Rectangle rec1, rec2(1, 1, 3, 4), rec3(p1, p2);

Rectangle rec4(rec2);

rec1.Print();

rec2.Print();

rec3.Print();

rec4.Print();

rec1.Set(3, 4, 100, 400);

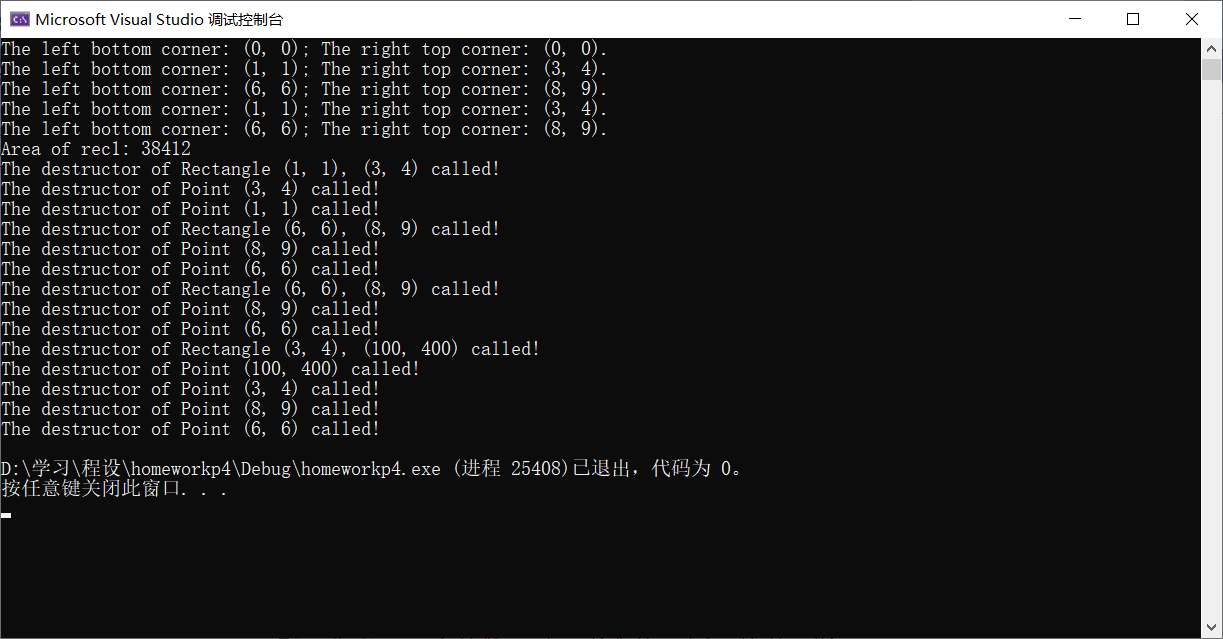
rec2.Set(p1, p2);

rec2.Print();

cout << "Area of rec1: " << rec1.Area() << endl;

return 0;

}



**2.**

#include <iostream>

#include <cmath>

#include <iomanip>

#include <cstdarg>

#include <ctime>

using namespace std;

class Ball

{

private:

int x;

int y;

int z;

int r;

const double PI = 3.14159265358979324;

public:

Ball() { x = y = z = r = 0; }

Ball(int \_x, int \_y, int \_z, int \_r) : x(\_x), y(\_y), z(\_z), r(abs(\_r)) {};

Ball(const Ball& \_new\_Ball) : x(\_new\_Ball.x), y(\_new\_Ball.y), z(\_new\_Ball.z), r(\_new\_Ball.r) {}

int GetX() { return x; }

int GetY() { return y; }

int GetZ() { return z; }

int GetR() { return r; }

void Set(int \_new\_x, int \_new\_y, int \_new\_z, int \_new\_r);

void Print() const;

double Volume() const { return 4.0 / 3.0 \* PI \* r \* r \* r; }

void Swap(Ball& b);

~Ball() { cout << "Destructor (" << x << ", " << y << ", " << z << " | " << r << ") called!" << endl; }

friend void Sort(Ball\* b, int num);

};

void Ball::Set(int \_new\_x, int \_new\_y, int \_new\_z, int \_new\_r)

{

x = \_new\_x;

y = \_new\_y;

z = \_new\_z;

r = abs(\_new\_r);

}

void Ball::Print() const

{

cout << "Centre: (" << setw(5) << x << ", " << setw(5) << y << ", " << setw(5) << z << "), radium: " << setw(5) << r << ". ";

}

void Ball::Swap(Ball& b)

{

struct

{

int x;

int y;

int z;

int r;

} tmp = {x, y, z, r};

Set(b.x, b.y, b.z, b.r);

b.Set(tmp.x, tmp.y, tmp.z, tmp.r);

}

void Sort(Ball\* b, int num)

{

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

for (int j = i + 1; j < num; ++j)

if (b[i].r > b[j].r)

b[i].Swap(b[j]);

}

int main()

{

srand(0);

Ball b1[10];

Ball\* b2 = new Ball[20];

cout << "Static balls:" << endl;

for (int i = 0; i < 10; ++i)

{

b1[i].Set(rand(), rand(), rand(), abs(rand()));

b1[i].Print();

cout << "Volume: " << b1[i].Volume() << endl;

}

cout << endl << "-------------------------" << endl << endl << "Dynamic balls:" << endl;

for (int i = 0; i < 20; ++i)

{

b2[i].Set(rand(), rand(), rand(), abs(rand()));

b2[i].Print();

cout << "Volume: " << b2[i].Volume() << endl;

}

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

Sort(b1, 10);

Sort(b2, 20);

cout << "After sorting: " << endl << endl << "static balls: " << endl;

for (int i = 0; i < 10; ++i)

{

b1[i].Print();

cout << "Volume: " << b1[i].Volume() << endl;

}

cout << endl << "-------------------------" << endl << endl << "Dynamic balls:" << endl;

for (int i = 0; i < 20; ++i)

{

b2[i].Print();

cout << "Volume: " << b2[i].Volume() << endl;

}

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

int sumx = 0, sumy = 0, sumz = 0, sumr = 0;

double avex, avey, avez, aver, avev = 0.0;

for (int i = 0; i < 10; ++i)

{

sumx += b1[i].GetX();

sumy += b1[i].GetY();

sumz += b1[i].GetZ();

sumr += b1[i].GetR();

avev += b1[i].Volume() / 10.0;

}

avex = sumx / 10.0;

avey = sumy / 10.0;

avez = sumz / 10.0;

aver = sumr / 10.0;

cout << "Average:" << endl << endl;

cout << "Static balls: " << "Centre: (" << avex << ", " << avey << ", " << avez << "), radium: " << aver << ", Volume: " << avev << endl;

sumx = 0;

sumy = 0;

sumz = 0;

sumr = 0;

avev = 0.0;

for (int i = 0; i < 20; ++i)

{

sumx += b2[i].GetX();

sumy += b2[i].GetY();

sumz += b2[i].GetZ();

sumr += b2[i].GetR();

avev += b2[i].Volume() / 20.0;

}

avex = sumx / 20.0;

avey = sumy / 20.0;

avez = sumz / 20.0;

aver = sumr / 20.0;

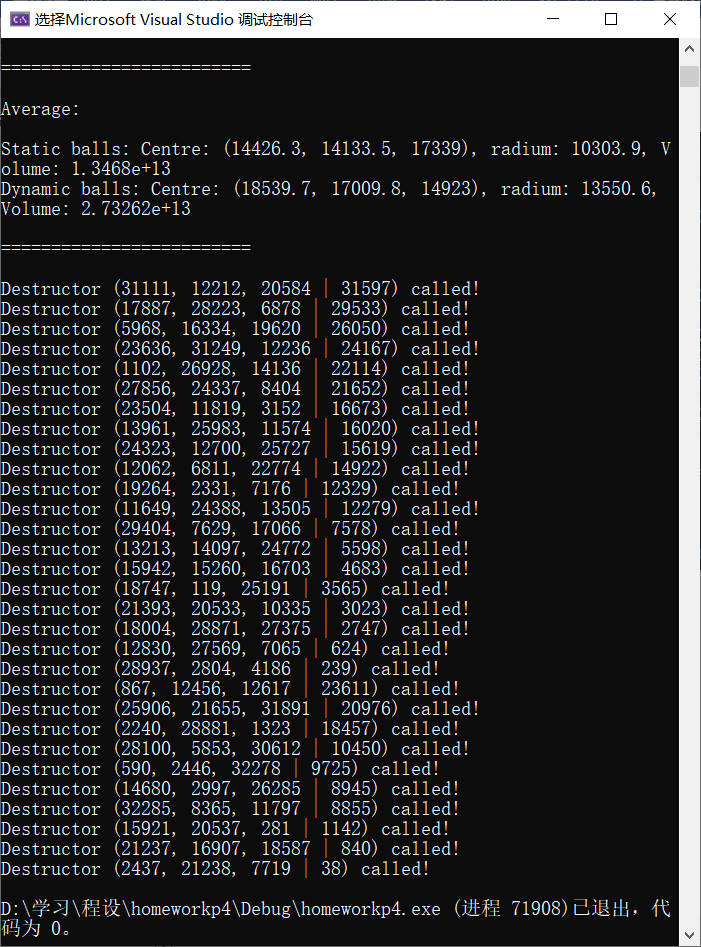
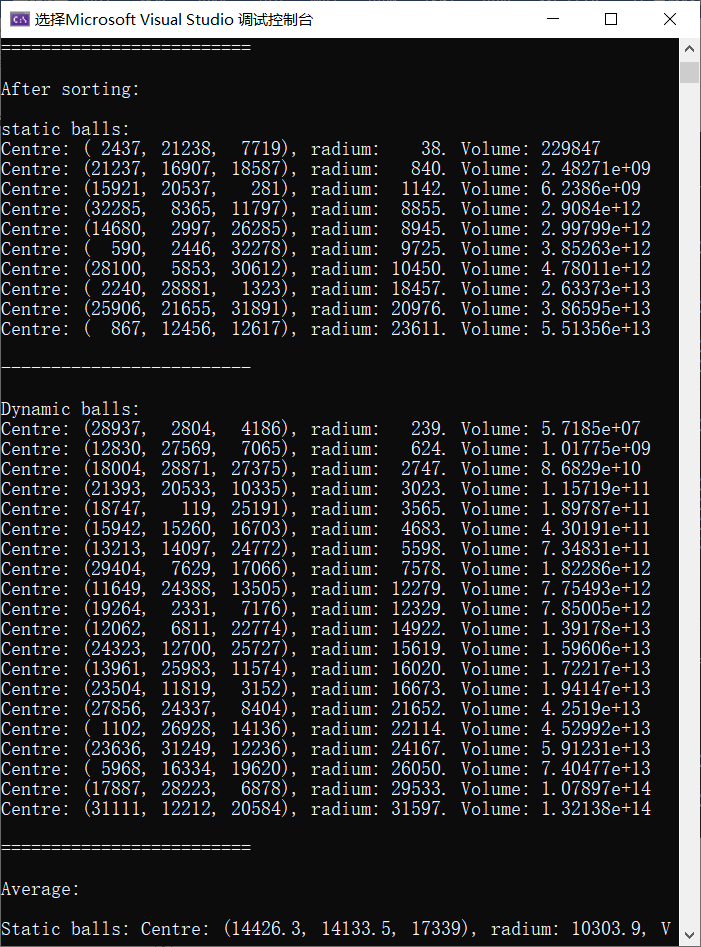
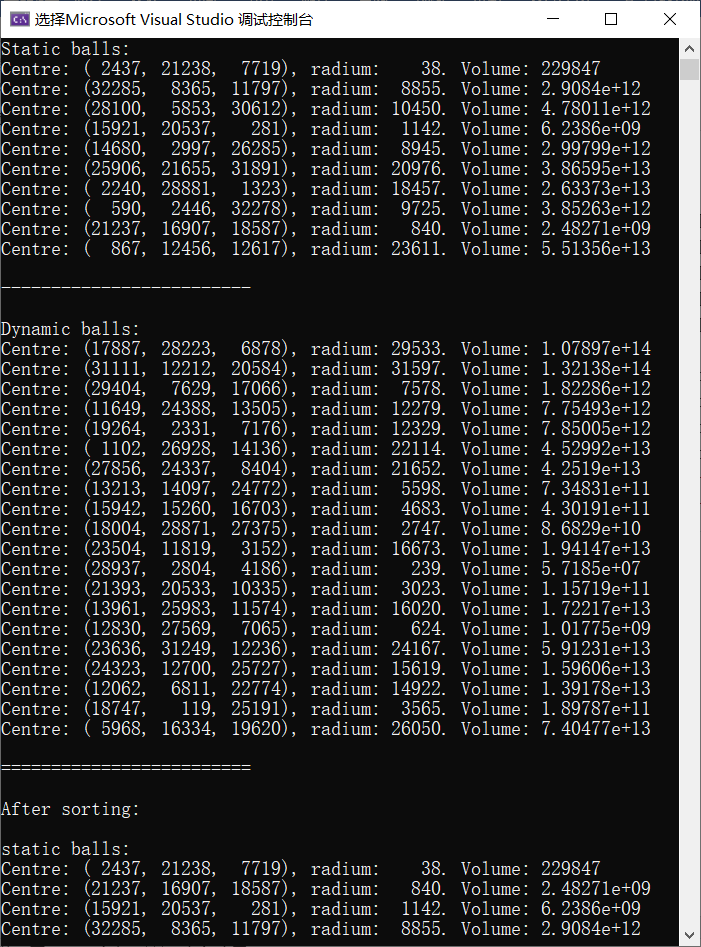
cout << "Dynamic balls: " << "Centre: (" << avex << ", " << avey << ", " << avez << "), radium: " << aver << ", Volume: " << avev << endl;

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

delete[] b2;

return 0;

}



**3（探究题）.**

#include <iostream>

using namespace std;

class list\_node

{

int data;

list\_node\* next;

friend class ordered\_list;

};

class ordered\_list

{

public:

ordered\_list() { head = NULL; }

void insert(int ins);

void visit(void(\*vis)(int&));

~ordered\_list();

private:

list\_node\* head;

};

void ordered\_list::insert(int ins)

{

list\_node\* newnode = new list\_node;

newnode->data = ins;

if (head == NULL || head->data > ins)

{

newnode->next = head;

head = newnode;

return;

}

list\_node\* p = head, \* q = head->next;

while (q)

{

if (q->data >= ins)

{

newnode->next = p->next;

p->next = newnode;

return;

}

p = q;

q = q->next;

}

p->next = newnode;

newnode->next = NULL;

}

void ordered\_list::visit(void(\*vis)(int&))

{

list\_node\* p = head;

while (p)

{

vis(p->data);

p = p->next;

}

}

ordered\_list::~ordered\_list()

{

list\_node\* p = head;

while (head)

{

p = head->next;

delete head;

head = p;

}

}

void output(int& i)

{

cout << i << " ";

}

int main()

{

ordered\_list ol;

int k;

while (cin >> k)

ol.insert(k);

ol.visit(output);

return 0;

}

运行结果如下：

