**《软件工程》实验报告三**

班级： 软件工程1501班

姓名： 部长好 吴怀文

朱全(软工1502)

学号： U201516991

U201516993

U201517031

**1.引言**

在一个有界[-1，1]的立体盒子里，有m个气球(它们不能重叠)有可变的半径r和位置mu。在给定位置的盒子里有一些小块，气球不能和这些块重叠。求r ^ 2 和最大时的半径和位置。

2.**核心算法**

（1）本算法所要考虑的是在所有情况中寻找最优的解，其中假设情况是有限的。

（2）本题的题意转化为数学语言，即在圆心坐标x，y的精度为小数点后两位时，通过计算满足情况的最大半径r，然后从这些半径中找出最大值。

（3）每一种情况下的最大半径精度为小数点后五位。

（4）通过链表进行判断，来选出其中满足条件的情况。

/\*有关气球的结构体构造\*/：

typedef struct ball

{

double x; //圆心x坐标

double y; //圆心y坐标

double z; //球心z坐标

double r; //圆半径

}Ball;

/\*用来储存已经放置的气球\*/

typedef struct ballList

{

struct ballList \* next;

Ball ball;

}BallList;

void insert(Ball ball); //插入气球

double distance(Ball b1, Ball b2); //判断气球之间的距离

double abs(double num); //取绝对值

int judge(Ball b); //判断新加入的气球是否符合规则

void putBall(); //改变气球的初始位置，求的满足条件的气球

void putPoint(double x, double y, double z); //放置点

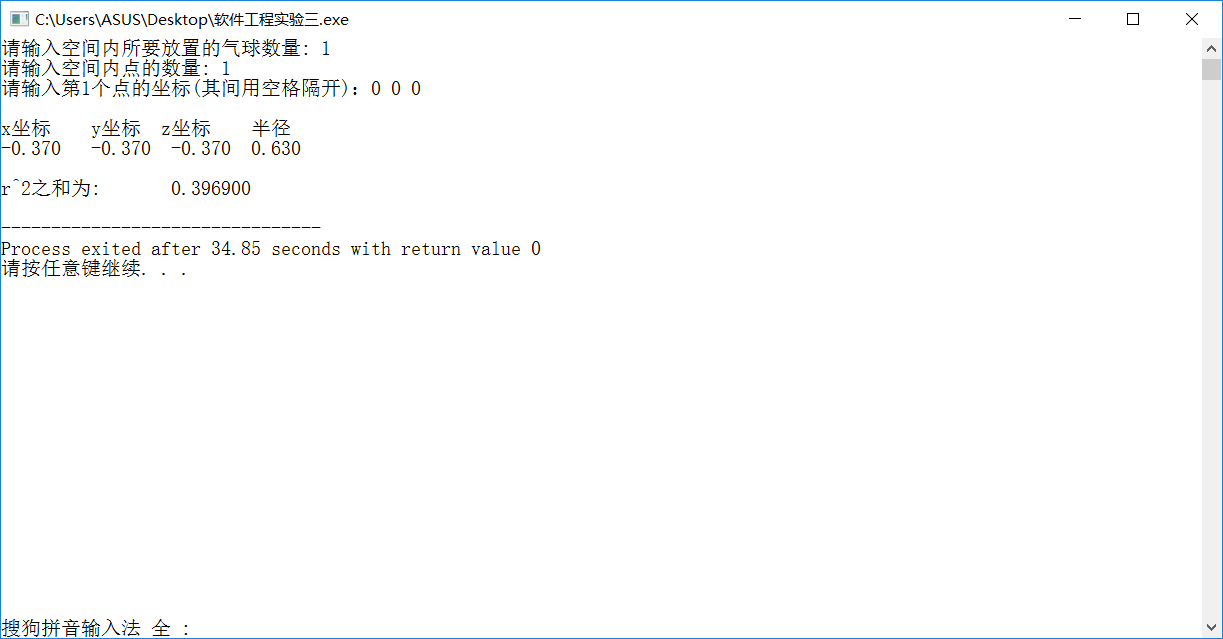
BallList \*head = NULL;

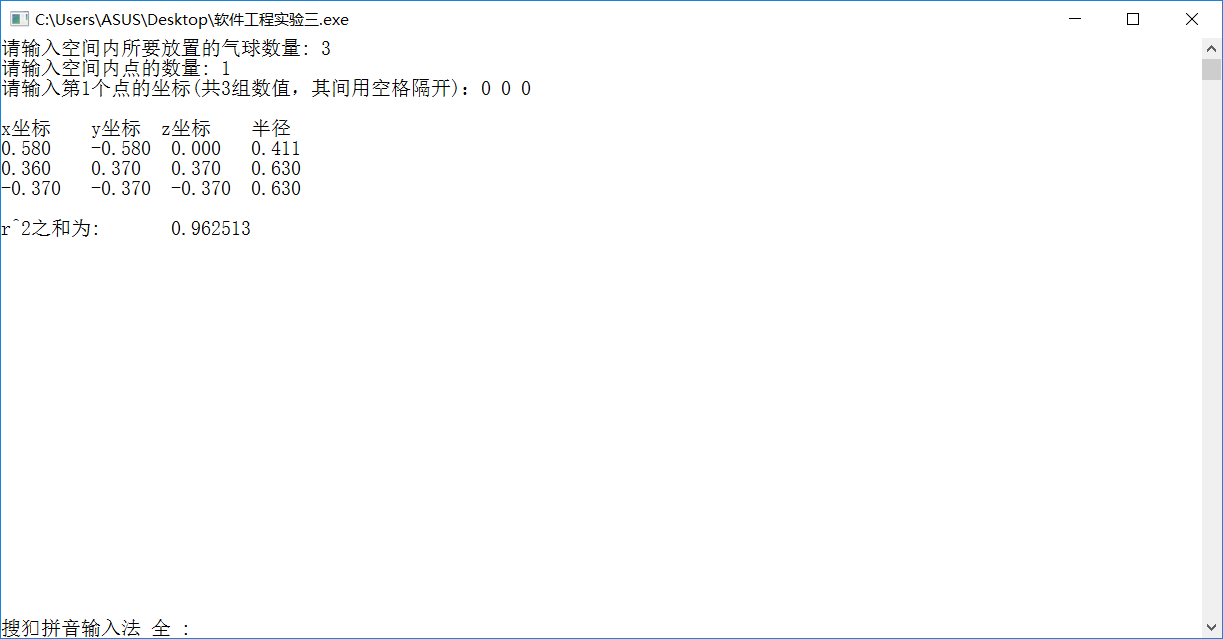
double step = 0.01; //改变气球位置的最小单位

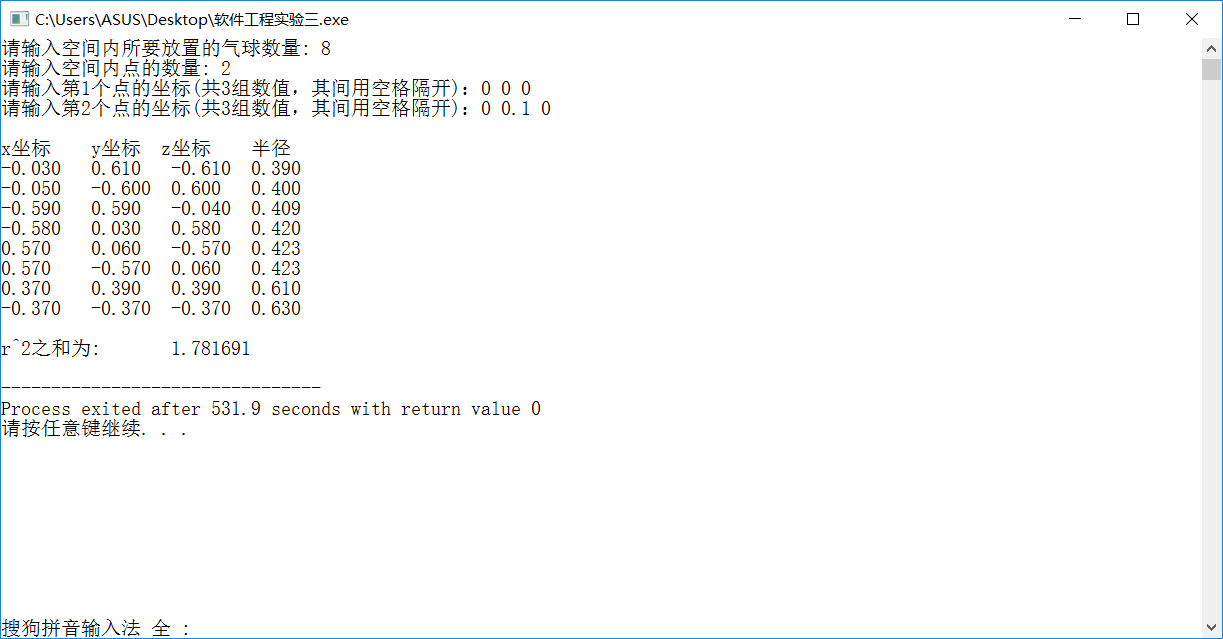
int num = 0; //放置气球的个数

double sumr = 0; //用来记录r^2之和

**3.实验截图**







**4.实验结果**

基本上解决了在一个边界为[-1, 1]的立体盒子,有m个半径为r和位置mu的气球(它们不能重叠)，在给定位置的盒子里有一些小块，气球不能和这些块重叠。求在r ^ 2为最大情况时的半径r和位置mu的问题

**5.实验结论**

边界为[-1, 1]的立体盒子,有m气球(他们不能重叠)变量半径r和位置mu,求r ^ 2最大情况的半径r和位置mu的问题基本解决，但是如果要得到更加精确的结果需要更加精细的计算，计算过程较为暴力，在运行开销较大的情况是不是一种好的算法。

**6.源代码**

#include <iostream>

#include <stdlib.h>

#include <stdio.h>

#include<unistd.h>//sleep函数

#include <math.h>

//气球

typedef struct ball

{

double x; //圆心x坐标

double y; //圆心y坐标

double z; //球心z坐标

double r; //圆半径

}Ball;

//用来储存已经放置的气球

typedef struct ballList

{

struct ballList \* next;

Ball ball;

}BallList;

void insert(Ball ball); //插入气球

double distance(Ball b1, Ball b2); //判断气球之间的距离

double abs(double num); //取绝对值

int judge(Ball b); //判断新加入的气球是否符合规则

void putBall(); //改变气球的初始位置，求的满足条件的气球

void putPoint(double x, double y); //放置点

BallList \*head = NULL;

double step = 0.01; //改变气球位置的最小单位

int num = 0; //放置气球的个数

double sumr = 0; //用来记录r^2之和

int main()

int n, pointNum, i;

printf("请输入空间内所要放置的气球数量: ");

scanf("%d",&n);

printf("请输入空间内点的数量: ");

scanf("%d",&pointNum);

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

{

printf("请输入第%d个点的坐标(共3组数值，其间用空格隔开)：",i+1);

double x, y,z;

scanf("%lf %lf%lf", &x, &y,&z);

putPoint(x, y,z);

}

printf("\nx坐标\t y坐标\tz坐标\t 半径\n");

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

{

putBall();

}

ballList \*tmp = head,\*kz = head;

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

{kz=kz->next;

}

while(kz)

{

Ball ball = tmp->ball;

printf("%.3lf\t %.3lf\t %.3lf\t %.3lf \n", ball.x, ball.y,ball.z ,ball.r);

tmp = tmp->next;

kz=kz->next;

}

printf("\nr^2之和为:\t %lf\n", sumr);

sleep(3);//两秒后退出

return 0;

}

void putPoint(double x, double y,double z)

{

Ball ball = {x, y, z,0};

insert(ball);

}

void insert(Ball ball)

{

ballList \* newBall = (ballList \*)malloc(sizeof(ballList));

newBall->ball = ball;

newBall->next = head;

head = newBall;

}

void putBall()

{

Ball ball = {-1 + step, -1 + step,-1+step, 0};

Ball maxBall = ball;

int i, j,k;

for (i = 0; ball.x < 1; ++i)

{

ball.x += step;

ball.y = -1 + step;

ball.z=-1+step;

for (j = 0; ball.y < 1; ++j)

{

ball.y += step;

ball.z=-1+step;

for(k=0;ball.z<1;++k){

ball.z+=step;

ball.r = 0;

double rstep = 0.1;

while(rstep > 0.00001)

{

if(ball.r > maxBall.r)

{

maxBall = ball;

}

ball.r += rstep;

if(!judge(ball))

{

ball.r -= rstep;

rstep /= 10;

}

}

}

}

}

if(judge(maxBall))

{

insert(maxBall);

num++;

sumr += maxBall.r \* maxBall.r;

// printf("%.3lf\t %.3lf\t %.3lf\t %.3lf\t %lf \n", maxBall.x, maxBall.y,maxBall.z ,maxBall.r, sumr);

}

}

int judge(Ball b)

{

if((abs(b.x) + b.r) > 1 || (abs(b.y) + b.r) > 1||(abs(b.z)+b.r)>1)

{

return false;

}

ballList \*tmp = head;

while(tmp)

{

Ball ball = tmp->ball;

if(distance(b, ball) < b.r + ball.r )

{

return false;

}

tmp = tmp->next;

}

return true;

}

double abs(double num)

{

if(num > 0)

return num;

return 0 - num;

}

double distance(Ball b1, Ball b2)

{

double x1 = b1.x;

double y1 = b1.y;

double z1=b1.z;

double x2 = b2.x;

double y2 = b2.y;

double z2=b2.z;

return pow((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2)+(z1-z2)\*(z1-z2), 0.5);

}