**Data Structures**

Circle

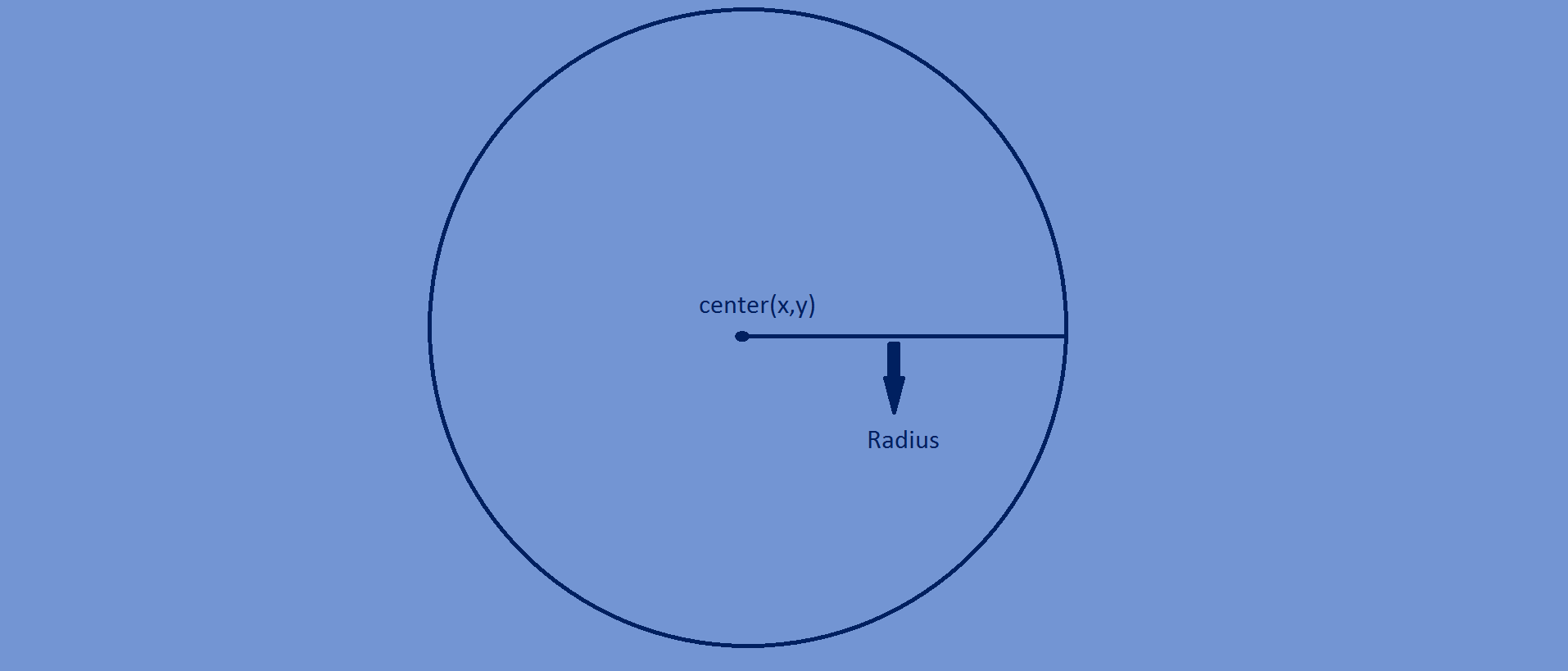
**Contents:**

* Find the area and circumference of generated circles .
* Write a function that finds the number of overlaps of each of *n* CIRCLEs

and prints the count.

* Print the generated circles.
* Writing whole program.

**Finding the area and Circumference of generated circles.**

* Formulae for calculating the area of circle is

Area=π×r2

Where r is the radius of the circle.

value of π=3.14

* Formulae for calculating the Circumference

of a circle is

Circumference=2×π×r

Where r is the radius of the circle.

* Radius of a circle :

r=c[i].radius

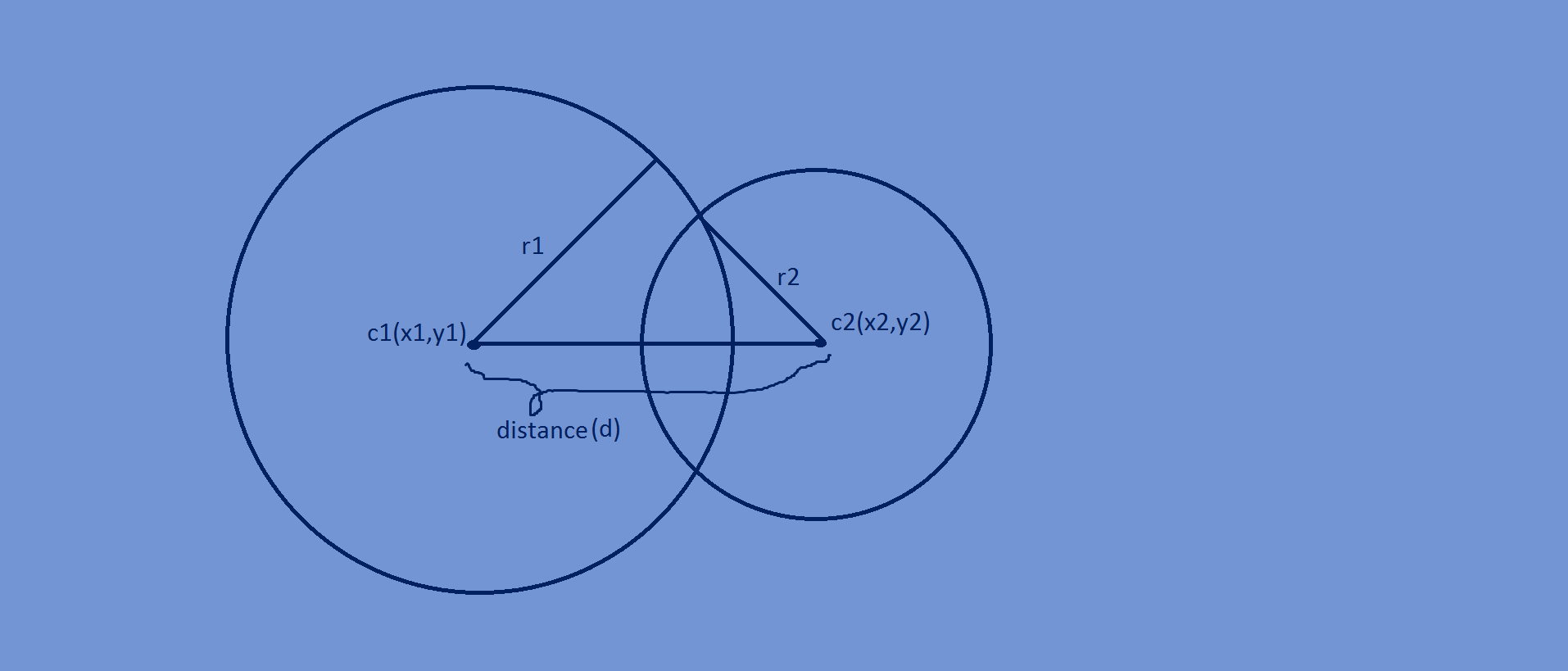
Where i is a particular circle in n circles.

//calculating area of n circles  
**void** findArea(CIRCLE \*c, **int** n) {  
 **int** i;  
 **float** pi=3.14;  
 **for**(i=0;i<n;i++) {  
 c[i].area=pi\*c[i].radius\*c[i].radius;  
 }  
}

//calculating circumference of n circles

**void** findCircumference(CIRCLE \*c,**int** n){  
 **int** i;  
 **float** pi=3.14;  
 **for**(i=0;i<n;i++) {  
 c[i].circumference=2\*pi\*c[i].radius;  
 }  
}

**Write a function that finds the number of overlaps of each of *n* CIRCLEs**

**and prints the count.**

* The condition for overlapping of two

Circles is that the distance between the centre’s

Of two circles must be less than the sum of radii

Of the two circles.

Distance(d)<r1+r2

here d=√(x2-x1)2+(y2-y1)2

x=c[i].p.x,y=c[i].p.y

* Here we have to check overlapping condition of a circle with the

Remaining n-1 circles to count total no of overlaps.

//printing number of overlaps  
**void** findOverlapCounts(CIRCLE \*c, **int** n)  
{  
 **int** i,j,count=0;  
 **float** totalrad,distance;  
 **for**(i=0;i<n;i++)  
 {  
 count=0;  
 **for**(j=0;j<n;j++)  
 {  
 totalrad=c[i].radius+c[j].radius;  
 distance=sqrt((c[i].p.x-c[j].p.x)\*(c[i].p.x-c[j].p.x)+(c[i].p.y-c[j].p.y)\*(c[i].p.y-c[j].p.y));  
 **if**(distance<totalrad && i!=j)  
 {  
 count++;  
 }  
 }  
 printf("%d\n",count);  
 }  
}

**Complete program:**

#include<stdio.h>  
#include<time.h>  
#include<stdlib.h>  
#include<math.h>  
  
**typedef struct** {  
 **float** x,y;  
}point;  
  
**typedef struct** {  
 **float** radius;  
 **float** area,circumference;  
 point p;  
}CIRCLE;  
  
CIRCLE \*genCircles(**int** n){  
 **int** i;  
 srand(time(NULL));  
 CIRCLE \*c;  
 c=(CIRCLE\*)malloc(n\***sizeof**(CIRCLE));  
 **for**(i=0;i<n;i++) {  
 c[i].p.x=rand()%13+8;  
 c[i].p.y=rand()%13+8;  
 c[i].radius=rand()%5+2;  
 }  
 **return** c;  
}  
  
//area of n circles  
**void** findArea(CIRCLE \*c, **int** n) {  
 **int** i;  
 **float** pi=3.14;  
 **for**(i=0;i<n;i++) {  
 c[i].area=pi\*c[i].radius\*c[i].radius;  
 }  
}  
  
**void** findCircumference(CIRCLE \*c,**int** n){  
 **int** i;  
 **float** pi=3.14;  
 **for**(i=0;i<n;i++) {  
 c[i].circumference=2\*pi\*c[i].radius;  
 }  
}  
  
//printing center,radius,vertices  
**void** printCircles(CIRCLE \*c, **int** n) {  
 **int** i;  
 **float** pi=3.14;  
 **for**(i=0;i<n;i++){  
 printf("(%f,%f) radius=%f area=%f circumference=%f \n",c[i].p.x,c[i].p.y,c[i].radius,c[i].area, c[i].circumference);  
 }  
}  
  
//printing number of overlaps  
**void** findOverlapCounts(CIRCLE \*c, **int** n) {  
 **int** i,j,count=0;  
 **float** totalrad,distance;  
 **for**(i=0;i<n;i++) {  
 count=0;  
 **for**(j=0;j<n;j++) {  
 totalrad=c[i].radius+c[j].radius;  
 distance=sqrt((c[i].p.x-c[j].p.x)\*(c[i].p.x-c[j].p.x)+(c[i].p.y-c[j].p.y)\*(c[i].p.y-c[j].p.y));  
 **if**(distance<totalrad && i!=j) {  
 count++;  
 }  
 }  
 printf("%d\n",count);  
 }  
}  
  
**int** main(**int** argc,**char**\*argv[]) {  
 **int** n=3;  
 CIRCLE \*c;  
 c=genCircles(n);  
 findArea(c,n);  
 findCircumference(c,n);  
 printCircles(c,n);  
 findOverlapCounts(c,n);  
 **return** 0;  
}

(14.000000,13.000000) radius=5.000000 area=78.500000 circumference=31.400002

(16.000000,15.000000) radius=2.000000 area=12.560000 circumference=12.560000

(18.000000,12.000000) radius=2.000000 area=12.560000 circumference=12.560000

2

2

2