Program 1

Write a program that first reads an integer for the array size, then reads number into the array, computes their average, and finds out how many numbers are above the average.

Program code

#include<iostream>

using namespace std;

double average(double a[],int size);

int above(double a[],int size,double aver);

int main()

{

int size;

cin>>size;

double a[80];

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

{

cin>>a[i];

}

cout<<average(a,size)<<endl;

cout<<above(a,size,average(a,size))<<endl;

return 0;

}

double average(double a[],int size)

{

double sum=0;

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

sum=sum+a[i];

return sum/size;

}

int above(double a[],int size,double aver)

{

int count=0;

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

if(a[i]>aver)

count++;

return count;

}

Program analysis

1.Using array as a parameter to do the calculate.

Program result

Program 2

Write two overloaded functions that return the average of an array with the following header:

int average(const int\* array, int size);

double average(const double\* array, int size);

Write a test program that prompts the user to enter ten double values, invokes this function, and displays the average value

Program code

#include<iostream>

using namespace std;

int average(const int\* array, int size);

double average(const double\* array, int size);

int main()

{

int size;

cin>>size;

double a[80];

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

{

cin>>a[i];

}

double \*array;

array=a;

cout<<average(array,size)<<endl;

return 0;

}

double average(const double\* array, int size)

{

double sum=0;

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

sum=sum+\*(array+i);

return sum/size;

}

int average(const int\* array, int size)

{

double sum=0;

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

sum=sum+\*(array+i);

return sum/size;

}

Program analysis

1. using a pointer to point the array.
2. The overloaded function can figure out which function to invoke.

Program result

Program 3

Use pointers to write a function that finds the smallest element in an array of integers. Use {1,2,4,5,10,100,2,-22} to test the function.

Program code

#include<iostream>

using namespace std;

double small(double \*pa, int size);

int main()

{

int size;

cin>>size;

double a[80];

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

{

cin>>a[i];

}

double \*pa;

pa=a;

cout<<small(a,size)<<endl;

return 0;

}

double small(double \*pa, int size)

{

double min=\*pa;

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

{

if (\*(pa+i)<=min)

min=\*(pa+i);

}

return min;

}

Program analysis

1.use a pointer to point to the array

Program result

Program 4

Define the Circle2D class that contains:

1) Two double data fields named x and y that specify the center of the circle with constant get functions;

2) A double data field radius with a constant get function;

3) A no-arg constructor that creates a default circle with (0,0) for (x,y) and 1 for radius;

4) A constructor that creates a circle with the specified x, y and radius;

5) A constant function getArea() that returns the area of circle;

6) A constant function getPerimeter() that returns the perimeter of the circle;

7) A constant function contains(double x, double y) that return true if the specified point (x,y) is inside this circle;

8) A constant function contains(const Circle2D& circle) that returns true if the specified circle is inside this circle;

9) A constant function overlaps(const Circle2D& circle) that returns true if the specified circle overlaps with this circle;

Implement the class. Write a test program that creates Circle2D objects, c1(2,2,5.5), c2(2,2,2.5) and c3(4,5,10.5), displays c1’s area and perimeter, the result of c1.contains(3,3), c1.contains(c2) and c1.overlaps(c3);

Program code

#include<iostream>

#include<math.h>

using namespace std;

class Circle2D

{

private:

double x,y;

double r;

public:

const void getx(double px)

{

x=px;

}

const void gety(double py)

{

y=py;

}

const void getr(double pr)

{

r=pr;

}

Circle2D()

{

x=0;y=0;r=1;

}

Circle2D(double px,double py,double pr)

{

x=px;y=py;r=pr;

}

double getArea() const

{

return 3.1416\*r\*r;

}

double getPerimeter() const

{

return 3.1416\*2\*r;

}

bool contains(double x, double y) const

{

return( (pow(x-this->x,2) + pow(y-this->y,2) -r\*r )<0 );

}

bool contains(const Circle2D& circle)const

{

return ((sqrt(pow(x-circle.x,2) + pow(y-circle.y,2))+circle.r)<=r);

}

bool overlaps(const Circle2D& circle)const

{

return( x==circle.x && y==circle.y && r==circle.r);

}

};

int main()

{

Circle2D c1(2,2,5.5);

Circle2D c2(2,2,2.5);

Circle2D c3(4,5,10.5);

cout<<c1.getArea()<<endl;

cout<<c1.getPerimeter()<<endl;

cout<<c1.contains(3,3)<<endl;

cout<<c1.contains(c2)<<endl;

cout<<c1.overlaps(c3)<<endl;

return 0;

}

Program analysis

1. use const get function to get the private member.
2. Define all the function as const to avoid be changed.

Program result