# Geometria Computacional

Círculos: problemas resolvidos

Prof. Edson Alves

2018

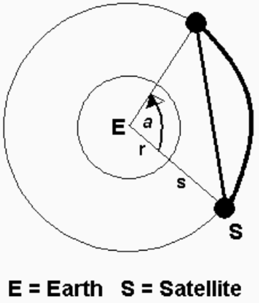
Faculdade UnB Gama

## Sumário

## UVA 10221 – Satellites

## Problema

The radius of earth is 6440 Kilometer. There are many Satellites and Asteroids moving around the earth. If two Satellites create an angle with the center of earth, can you find out the *distance* between them? By *distance* we mean both the *arc* and *chord* distances. Both satellites are on the same orbit (However, please consider that they are revolving on a circular path rather than an elliptical path).



## Entrada e saída

**Input**

The input file will contain one or more test cases.

Each test case consists of one line containing two-integer $s$ and $a$, and a string '`min`' or '`deg`'. Here $s$ is the distance of the satellite from the surface of the earth and $a$ is the angle that the satellites make with the center of earth. It may be in minutes ($'$) or in degrees ($^\circ$). Remember that the same line will never contain minute and degree at a time.

**Output**

For each test case, print one line containing the required distances i.e. both arc *distance* and *chord distance* respectively between two satellites in Kilometer. The distance will be a floating-point value with six digits after decimal point.

## Exemplo de entradas e saídas

**Sample Input**

```
500 30 deg
700 60 min
200 45 deg
```

**Sample Output**

```
3633.775503 3592.408346
124.616509 124.614927
5215.043805 5082.035982
```

## Solução

- A primeira etapa da solução é uniformizar as unidades de medida
- Sabendo que um grau tem 60 minutos, a conversão de $x$ minutos para $y$ graus é
$$y = \frac{x}{60}$$
- Sabendo que $\pi$ radianos correspondem a 180°, segue que o ângulo $\alpha$, em radianos, será de
$$\alpha = \frac{a\pi}{180}$$
- O comprimento $c$ do arco será dado por
$$c = \alpha R,$$
onde $R$ é o raio até o satélite, isto é, $r + s$
- A corda terá comprimento $L$ igual a
$$L = 2R\sin(\frac{\alpha}{2})$$
- Com estas informações, a solução terá complexidade $O(T)$, onde $T$ é o número de casos de teste

## Solução AC com complexidade $O(T)$

```cpp
#include <iostream>
#include <cmath>

using namespace std;

const int R { 6440 };
const double PI = acos(-1.0);

double arc(double s, double a)
{
    auto r = s + R;

    return r*a;
}

double chord(double s, double a)
{
    auto r = s + R;

    return 2*r*sin(a/2);
}
```

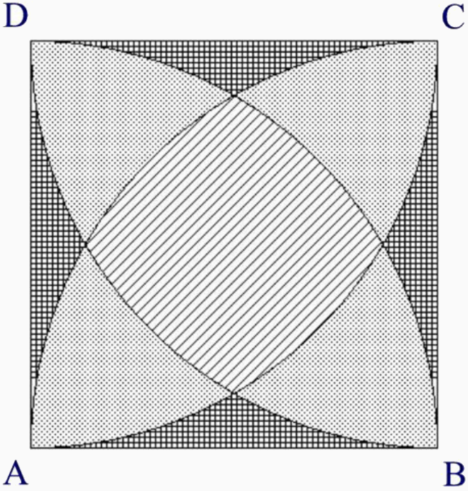
## Solução AC com complexidade $O(T)$

```
int main()
{
    double s, a;
    string unit;

    while (cin >> s >> a >> unit)
    {
        if (unit[0] == 'm')
            a /= 60;

        if (a > 180)
            a = 360 - a;

        a *= (PI/180.0);

        printf("%.6f %.6f\n", arc(s, a), chord(s, a));
    }

    return 0;
}
```

## UVA 10209 – Is This Integration?

## Problema

In the image below you can see a square $ABCD$, where $AB = BC = CD = DA = a$. Four arcs are drawn taking the four vertexes $A, B, C, D$ as centers and $a$ as the radius. The arc that is drawn taking $A$ as center, starts at neighboring vertex $B$ and ends at neighboring vertex $D$. All other arcs are drawn in a similar fashion. Regions of three different shapes are created in this fashion. You will have to determine the total area these different shaped regions.



## Entrada e saída

**Input**

The input file contains a floating-point number $a (0 \leq a \leq 10000)$ in each line which indicates the length of one side of the square. Input is terminated by end of file.

**Output**

For each line of input, output in a single line the total area of the three types of region (filled with different patterns in the image above).

These three numbers will of course be floating point numbers with three digits after the decimal point. First number will denote the area of the striped region, the second number will denote the total area of the dotted regions and the third number will denote the area of the rest of the regions.

## Exemplo de entradas e saídas

**Sample Input**

```
0.1
0.2
0.3
```

**Sample Output**

```
0.003 0.005 0.002
0.013 0.020 0.007
0.028 0.046 0.016
```

## Solução

- Observe, inicialmente, que a área tracejada é composta por um quadrado cujo lado é o comprimento da corda $c$ gerada por um ângulo igual a um terço de $\pi/2$ , somado a quatro segmentos $g$
- Assim, $A_T = c^2 - 4g$, onde

$$
\begin{aligned}
c &= 2a\sin\left(\frac{\pi}{6}\right) \\
s &= \frac{a+a+c}{2} \\
T &= \sqrt{(s-a)(s-a)(s-c)} \\
S &= \frac{\pi a^2}{12} \\
g &= S - T
\end{aligned}
$$

- Já a área pontilhada é a metade do que resta quando subtraída, da área de um semi-círculo de raio $a$, a área de um quadrado de lado $a$ e a área tracejada

## Solução

- Portanto,
$$A_P = \frac{\pi a^2}{2} - a^2 - A_T$$
- Por fim, a área restante $A_R$ é o que sobra da diferença entre a área de um quadrado de lado $a$ e as duas áreas já computadas
- Logo,
$$A_R = a^2 - A_T - A_P$$
- As expressões apresentadas permitem a implementação de uma solução $O(T)$, onde $T$ é o número de casos de teste

## Solução AC com complexidade $O(T)$

```cpp
#include <iostream>
#include <cmath>

using namespace std;

const double PI { acos(-1.0) };

struct Circle
{
    double r;

    double arc(double theta) const
    {
        return theta * r;
    }

    double chord(double a) const
    {
        return 2 * r * sin(a/2);
    }

```

## Solução AC com complexidade $O(T)$

```
    double sector(double theta) const
    {
        return (theta * r * r)/2;
    }

    double segment(double a) const
    {
        auto c = chord(a);
        auto s = (r + r + c)/2.0;
        auto T = sqrt(s*(s - r)*(s - r)*(s - c));

        return sector(a) - T;
    }
};

```

## Solução AC com complexidade $O(T)$

```
int main()
{
    double a;

    while (scanf("%lf", &a) == 1)
    {
        Circle circle { a };

        double c = circle.chord(PI/6);
        double g = circle.segment(PI/6);

        double stripped = c * c + 4*g;
        double dotted = 4*(-stripped + (PI/2.0 - 1.0)*a*a)/2.0;
        double rest = a*a - stripped - dotted;

        printf("%.3f %.3f %.3f\n", stripped, dotted, rest);
    }

    return 0;
}
```

## Referências

1. UVA 10221 – Satellites
2. UVA 10209 – Is This Integration?