

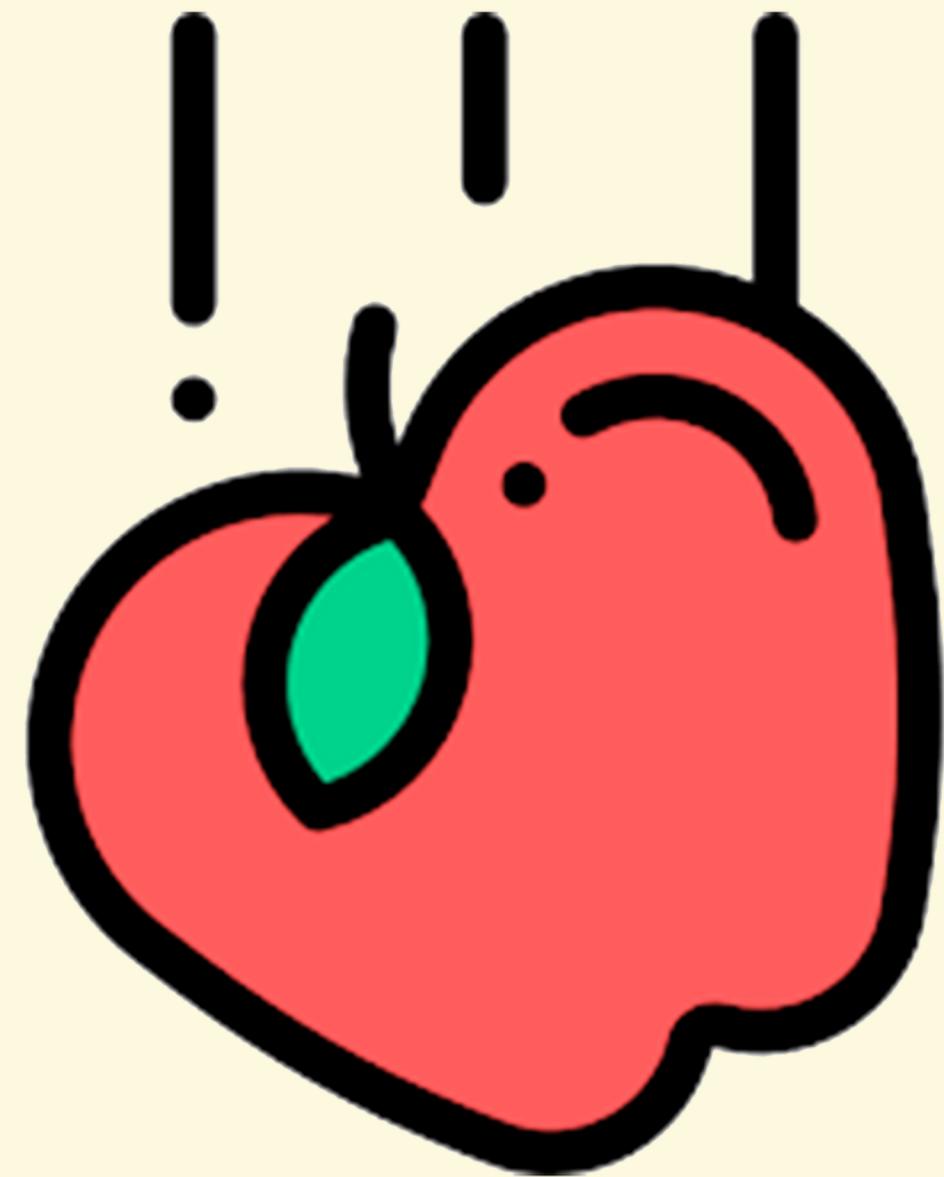
# Corpos celestes: Planeta e asteroide.

Programação Orientado a objetos; C3



# Explicação Física e científica do código

Este programa simula conceitos fundamentais da astrofísica e da mecânica celeste, modelando corpos celestes como planetas e asteroides. Ele utiliza princípios da gravitação universal de Newton para calcular a aceleração gravitacional de um planeta e o terceiro lei de Kepler para determinar o período orbital de um asteroide.



# Gravidade planetária.

O programa calcula a aceleração gravitacional na superfície de um planeta usando a equação de Newton:

$$g = \frac{G \cdot M}{R^2}$$

- $G = 6,674 \times 10^{-11}$  (constante gravitacional universal),
- $M$  é a massa do planeta,
- $R$  é o raio do planeta.
- Assim, ele determina a força gravitacional que um corpo experimentaria ao estar na superfície do planeta.



# Órbitas e leis de Kepler

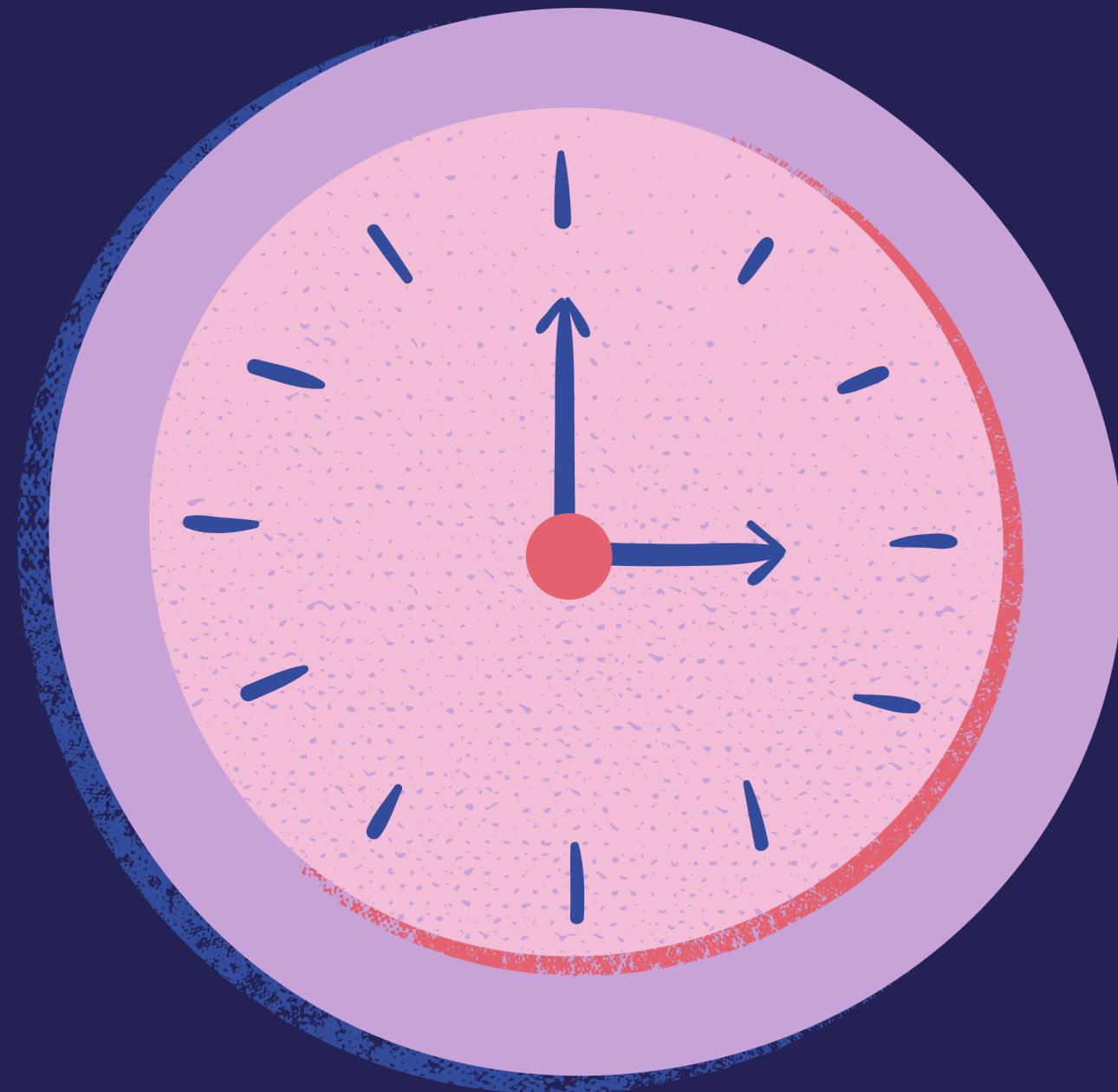
O asteroide tem uma órbita ao redor de um planeta, e seu período orbital é calculado usando a Terceira Lei de Kepler

$$T = 2\pi\sqrt{\frac{a^3}{G \cdot M}}$$

Onde:

- $a$  é o semi-eixo maior da órbita do asteroide,
- $M$  é a massa do planeta ao redor do qual ele orbita,
- $T$  representa o tempo que o asteroide leva para completar uma volta ao redor do planeta.

Esse cálculo é fundamental para entender os movimentos de asteroides, luas e até exoplanetas em torno de estrelas.



## Órbita Elíptica e Excentricidade

A excentricidade define o quanto oval é a órbita do asteroide. Quando  $e=0$ , a órbita é circular; quando  $e$  está próximo de 1, a órbita é extremamente alongada.

# Ceres

O asteroide mencionado no código é Ceres, o maior objeto do Cinturão de Asteroides entre Marte e Júpiter. Ceres é classificado como um planeta anão, pois tem massa suficiente para ter uma forma esférica, mas não limpou sua órbita de outros corpos menores.

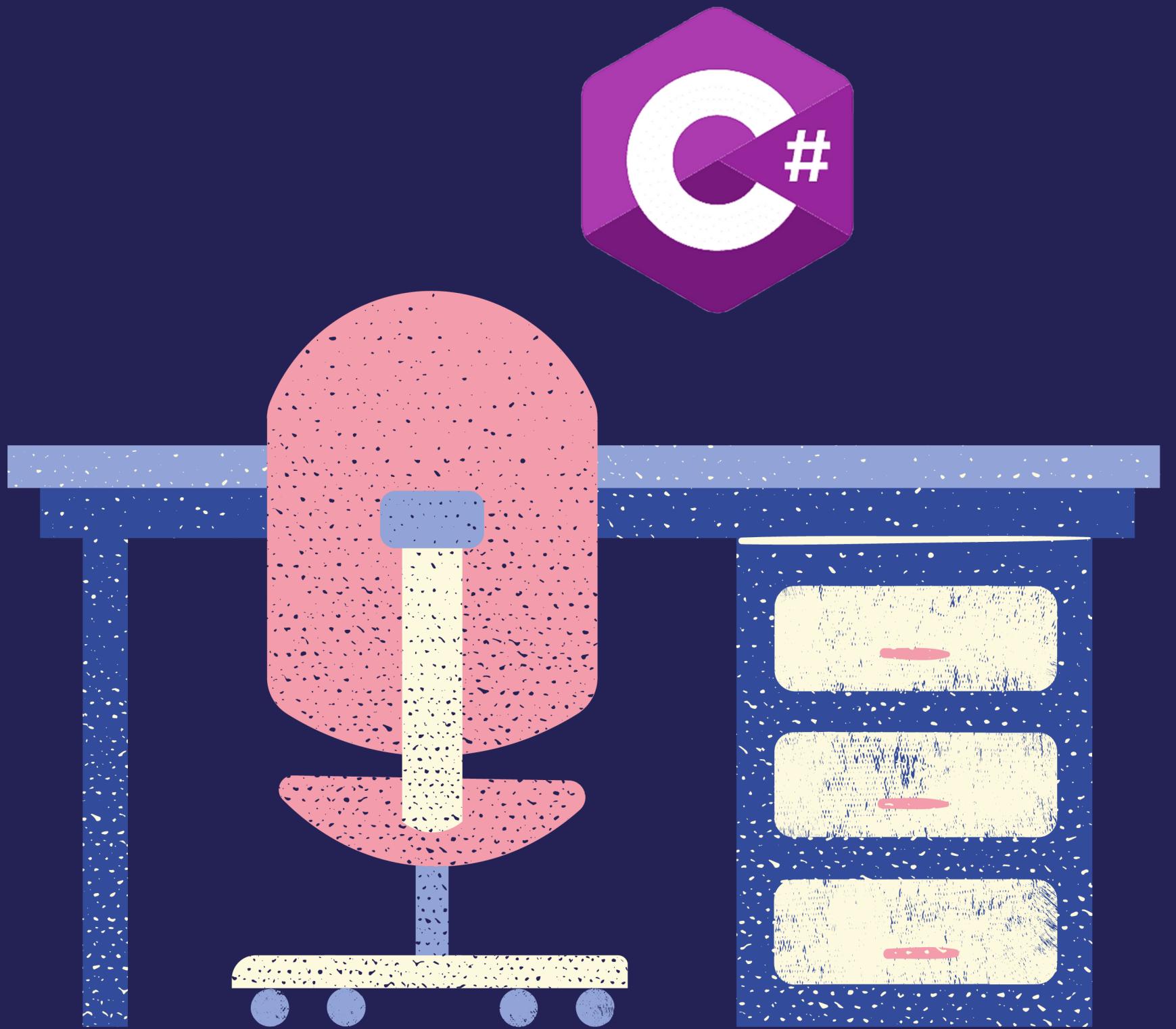


Descoberto em: 1801, por Giuseppe Piazzi

# Conclusão

Este código representa uma simulação simplificada do Sistema Solar, aplicando conceitos de física clássica e astrofísica. Ele pode ser expandido para incluir mais corpos celestes, interações gravitacionais mais complexas e até mesmo visualizações gráficas do movimento dos astros.

# Explicação do código. C#



# IMPORTAÇÃO O DAS BIBLIOTECAS

```
using System;
using
System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks
```

# DEFINIÇÃO DA CLASSE "corpoCeleste"

```
public class corpoCeleste
```

Esta é uma classe base que representa qualquer corpo celeste (planetas, asteroides, luas, etc.).

Encapsula informações comuns a todos os corpos celestes, como:

- Nome (n)
- Massa (m)
- Raio (r)

# ATRIBUTOS DA CLASSE "corpoCeleste"

```
protected string n;  
protected double m;  
protected double r;
```

Os atributos são **protected**, ou seja, podem ser acessados apenas pelas subclasses (exemplo: planeta e asteroide). Definem as propriedades básicas de um corpo celeste.

# CONSTRUTOR DA CLASSE

```
public corpoCeleste() { }
```

# MÉTODO "setDados"

```
public virtual void SetDados(string nome, double massa, double raio)
{
    this.n = nome;
    this.m = massa;
    this.r = raio;
}
```

- Objetivo: definir os dados básicos do corpo celeste.
- Uso do `this`: diferencia a variável de instância (`n`, `m`, `r`) dos parâmetros do método.
- Palavra-chave `virtual`: permite que esse método seja sobreescrito (`override`) nas subclasses.

# MÉTODO "exibir\_informações"

```
public virtual void exibir_informações()
{
    Console.WriteLine($" -Corpo Celeste: {n}");
    Console.WriteLine($" - Massa: {m} kg");
    Console.WriteLine($" - Raio: {r} m");
}
```

- Objetivo: Exibir as informações do corpo celeste no console.
- Uso do \$ (interpolação de string): facilita a exibição dos valores.
- Uso de virtual: permite que subclasses alterem esse método.

# MÉTODO “setDados”

```
public double getmassa()
{
    return m;
}
```

```
public string getnome()
{
    return n;
}
```

- Objetivo: Permitir acesso à massa e ao nome do corpo celeste de forma controlada.

# DEFINIÇÃO DA CLASSE "planeta"

```
public class planeta : corpoCeleste
```

A classe planeta herda de corpoCeleste, ou seja, recebe seus atributos e métodos. Essa é uma aplicação de herança da Programação Orientada a Objetos (POO).

## CONSTRUTOR DA CLASSE

```
public planeta()
```

## Método SetDados (Sobrescrito)

```
public override void SetDados(string nome,  
double massa, double raio)  
{  
    base.SetDados(nome, massa, raio);  
}
```

Usa override para sobrescrever o método da classe base.  
base.SetDados(...) chama o método da classe corpoCeleste, garantindo que o planeta armazene suas informações corretamente.

# MÉTODO PARA CALCULAR A GRAVIDADE

```
public double GetGravidade()  
{  
    double G = 6.674e-11;  
    return (G * m) / (r * r);  
}
```

Retorna o valor da gravidade na superfície do planeta.

# EXIBIR INFORMAÇÕES DO PLANETA

```
public override void exibir_informações()  
{  
    base.exibir_informações();  
    Console.WriteLine($" - Gravidade:  
{GetGravidade():F2} m/s²");  
}
```

Chama `exibir_informações()` da classe base e adiciona a gravidade.  
F2 formata a saída com duas casas decimais.

# Definição da Classe "asteroide"

```
public class asteroide : corpoCeleste
```

Herda de corpoCeleste, assim como planeta.

Adiciona novas propriedades específicas de asteroides, como:

- semiEixoMaior
- excentricidade
- orbitaEmTornoDe (referência ao planeta que ele orbita).

# MÉTODO PARA CALCULAR O PERÍODO ORBITAL

```
public double GetPeriodoOrbital()
{
    if (orbitaEmTornoDe == null)
    {
        throw new InvalidOperationException("O asteroide não está orbitando nenhum planeta!");
    }

    double G = 6.674e-11;
    double M = orbitaEmTornoDe.getmassa();
    return 2 * Math.PI * Math.Sqrt(Math.Pow(semiEixoMaior, 3) / (G * M));
}
```

Aplica a Terceira Lei de Kepler para calcular o tempo que o asteroide leva para orbitar um planeta.  
Verifica se o asteroide tem um planeta associado (orbitaEmTornoDe != null).

# MÉTODO PARA EXIBIR INFORMAÇÕES

```
public override void exibir_informações()
{
    base.exibir_informações();
    Console.WriteLine($" - Orbita ao redor de: {orbitaEmTornoDe != null ? orbitaEmTornoDe.getnome() : "Desconhecido"}");
    Console.WriteLine($" - Semi-eixo maior: {semiEixoMaior:F0} m");
    Console.WriteLine($" - Excentricidade: {excentricidade:F2}");

    if (orbitaEmTornoDe != null)
    {
        Console.WriteLine($" - Período orbital: {GetPeriodoOrbital() / (60 * 60 * 24):F2} dias");
    }
}
```

Exibe os parâmetros da órbita do asteroide.  
Se houver um planeta de órbita definido, exibe o período orbital.

# PROGRAMA PRINCIPAL

```
static void Main(string[] args)
```

Função principal do programa, onde os objetos são criados e testados.

# CRIANDO UM PLANETA (TERRA)

```
planeta planeta = new planeta();
planeta.SetDados("Terra", 5.972e24, 6.371e6);
planeta.exibir_informações();
Cria e define a Terra, exibindo suas informações.
```

# CRIANDO UM ASTEROIDE (CERES)

```
astroide ceres = new astroide("Ceres", 9.393e20, 469730,
2.77e11, 0.08);
ceres.SetOrbita(planeta);
ceres.exibir_informações();
```

Cria Ceres, define sua órbita ao redor da Terra e exibe suas informações.

# FINALIZAÇÃO

```
Console.ReadKey();
```

Aguarda o usuário pressionar uma tecla para manter o console aberto.

José Ulika Chiseva