# Atividade

O cálculo das raízes da equação do segundo grau é algo rotineiro para os alunos de ensino médio. Visando facilitar a conferência das respostas corretas a partir dos exercícios passados pelo professor, vamos construir uma solução (em Python) que calcule as raízes por Bhaskara a partir de valores fornecidos pelo usuário até que ele não queira mais utilizar a solução que você desenvolveu.

Antes, vamos lembrar como as raízes são calculadas por Bhaskara com um exemplo:

**Expressão da Equação do segundo grau:**



Para aplicar, vamos utilizar um exemplo que calcule duas raízes distintas:

Texto

Descrição gerada automaticamente

**Calculando o Delta:** Construir um subalgoritmo para isso

Uma imagem contendo Calendário

Descrição gerada automaticamente

**Calculando as Raízes x1 e x2:** Construir um subalgoritmo para isso

Fórmula de Bhaskara:

Diagrama

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente com confiança média

Neste exemplo, obtivemos duas raízes distintas (x1 e x2) porque o delta resultou um valor positivo. Contudo, nem sempre a resolução é possível ou é desta forma. Então, considere os seguintes requisitos, restrições e orientações:

* Se o valor de **a** for 0 (zero), exibir a mensagem **“Esta equação não é do segundo grau, sim do primeiro”.**  (Não tem como calcular). Construir um subalgoritmo para isso.
* Se b ou c forem 0, exibir os valores encontrados (para delta, x1 e x2), em seguida a mensagem **“Equação do segundo grau INCOMPLETA”**. Construir um subalgoritmo para isso
* Se b e c não forem 0, exibir os valores encontrados (para delta, x1 e x2), em seguida a mensagem **“Equação do segundo grau COMPLETA”**. Construir um subalgoritmo para isso
* Se o delta resultar um valor negativo, não é possível calcular por Bhaskara com números Reais porque não há raiz quadrada negativa; então exibir o valor de delta e a mensagem **“Não é possível calcular x1 e x2 porque o delta é negativo”**. Construir um subalgoritmo para isso
* Se o delta resultar zero a equação admite o mesmo valor para as duas raízes, então exibir os valores encontrados (para delta, x1 e x2), em seguida a mensagem **“As raízes x1 e x2 tem o mesmo valor”.** Construir um subalgoritmo para isso
* Se o delta resultar um valor positivo (como no exemplo acima) as raízes se dão por duas soluções distintas para x1 e x2, então exibir os valores encontrados (para delta, x1 e x2), em seguida a mensagem **“As raízes x1 e x2 tem valores distintos”.** Construir um subalgoritmo para isso

Agora que lembramos como se calcula a equação do segundo grau por Bhaskara, vamos construir uma solução em Python implementando Subalgoritmos (Funções e procedimentos com ou sem passagem de parâmetros) onde for possível e que resolva este problema (quando possível) a partir de 3 valores fornecidos pelo usuário. Exibir as frases em negrito descritas acima.

Uma ajuda:

Para calcular a raiz quadrada de um número, podemos utilizar a função sqrt(), veja uma aplicação:

# Acrescente esta biblioteca no início do Arquivo  
import math  
  
# y = math.sqrt(x) - calcula a raiz quadrada de x e atribui a y  
raiz = math.sqrt(25) # a variável raiz valerá 5

Ufa, verificamos todas as possibilidades e eventualmente calculamos...

NÃO, NÃO ACABOU!

Ao final da execução do cálculo (OU NÃO) das raízes, perguntar ao usuário: **“Continuar executando o programa? [S]im ou [N]ão:”**. Considere “S” (ou “s” minúsculo) para continuar executando ou “N” (ou “n” minúsculo) para terminar a execução do programa. CASO O USUÁRIO **NÃO DIGITE** UMA DESTAS DUAS LETRAS (em maiúsculo ou minúsculo), adverti-lo com a mensagem **“Letra inválida! Digite [S]im ou [N]ão:”**, até que ele digite a letra correta.

Considerações:

* Você deve aplicar Subalgoritmos onde for possível.
* Cada Subalgoritmo deve resolver apenas UM problema.
* Exceto a função (método) sqrt, nenhuma outra função proprietária do Python pode ser utilizada.

**TESTES:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PLANO DE TESTE | | | | | | | |
| **Teste** | **a** | b | c | delta | x1 | x2 | Mensagem |
| ***1*** | 1 | 2 | 3 | -8 | - | - | Delta = -8.0.  Não é possível calcular x1 e x2 porque o delta é negativo. |
| ***2*** | -1 | 2 | 3 | 16 | -1 | 3 | Delta = 16.0 x1 = -1.0 x2 = 3.0 Equação do segundo grau COMPLETA As raízes x1 e x2 tem valores distintos |
| ***3*** | 0 | - | - | - | - | - | Esta equação não é do segundo grau, sim do primeiro. |
| ***4*** | -1 | 0 | 2 | 8 | -1,41 | 1,41 | Delta = 8.0 x1 = -1.4142135623730951 x2 = 1.4142135623730951 Equação do segundo grau INCOMPLETA As raízes x1 e x2 tem valores distintos |
| ***5*** | -1 | 2 | 0 | 4 | 0 | 2 | Delta = 4.0 x1 = -0.0 x2 = 2.0 Equação do segundo grau INCOMPLETA As raízes x1 e x2 tem valores distintos |
| ***6*** | 1 | 2 | 1 | 0 | -1 | -1 | Delta = 0.0 x1 = -1.0 x2 = -1.0 Equação do segundo grau COMPLETA As raízes x1 e x2 tem o mesmo valor |
| **TESTAR A OBRIGATORIEDADE DE "S" OU "N" PARA CONTINUAR OU NÃO A EXECUÇÃO** | | | | | | | |

No TEAMS, dentro do canal “Fase 4”, tem uma conversa aberta chamada “Discussão sobre a Atividade não avaliativa de CTWP”. Postem lá o seu arquivo .py, baixem as versões dos colegas e discutam a solução que cada fez.

O professor também postará um gabarito.

Minhas crianças, desejo que tenham aprendido o conceito de Subalgoritmos e curtido este capítulo.

Até o próximo!