INSTITUTO FEDERAL
Catarinense
Campus Blumenau

# Algoritmos

Bacharelado em Ciência da Computação
Prof. Dr. Paulo César Rodacki Gomes

## Lista de exercícios -Alg-04
## Exercícios sobre Laços de repetição

Estes exercícios devem ser entregues no Google Classroom. Para cada um dos exercícios, crie um arquivo fonte Python com o respectivo nome de acordo com a seguinte regra: `SUASINICIAIS-Alg-04-Ex-num.py`. Por exemplo, se o professor resolvesse o exercício número 3, o nome do arquivo seria `PCRG-Alg-04-Ex-03.py`.

## Introdução

Todos os exercícios desta lista devem ser realizados com o uso de laços de repetição. Em alguns casos, o enunciado do exercício especifica qual tipo de laço você deve utilizar. Quando não estiver especificado, você mesmo deve decidir se prefere usar laço com comando **for** ou com comando **while**. Talvez você perceba que alguns exercícios tendem a ficar mais bem resolvidos com um tipo de laço do que com outro. Além disso, alguns exercícios podem requerer múltiplos laços (lembre-se que podemos aninhar um laço dentro de outro). Seja cuidadoso e criterioso ao escolher qual tipo de laço você vai usar - **isso é importante** - e, à medida que você vai ficando mais experiente, essa escolha vai ficando mais clara para você.

**Questões:**

1. **Média aritmética.** Escreva um programa Python que calcula a média aritmética de um conjunto de valores fornecidos pelo usuário. O usuário deve entrar com o valor 0 indicando que não serão mais fornecidos novos valores. Seu programa deve exibir uma mensagem de erro se o primeiro valor fornecido pelo usuário for 0.

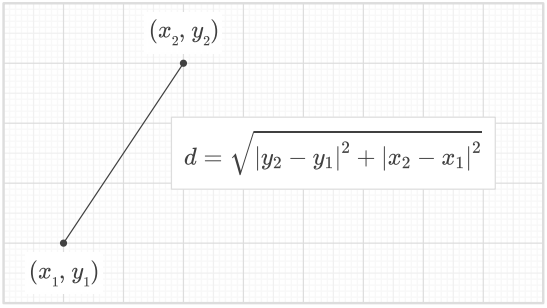
   **Dica**: o número 0 não deve ser incluído no cálculo da média, pois ele só serve para sinalizar o final da entrada de dados.

2. **Tabela de descontos**. Uma loja está oferecendo uma liquidação com descontos de 60% em uma variedade de produtos em final de estoque. O vendedor gostaria de ajudar seus clientes a determinar o preço reduzido (com desconto) de seus produtos. Ele quer criar uma tabela que mostra os preços originais e os preços após o desconto ser aplicado. Escreva um programa Python usando laço de repetição que gere esta tabela mostrando o preço original, o valor de desconto e o novo valor com desconto aplicado para produtos com os seguintes valores: R$ 4.95, R$ 9.95, R$ 14.95, R$ 19.95 e R$ 24.95. Certifique-se que todos os valores são mostrados com duas casas decimais.

3. **Tabela de conversão de temperaturas**. Escreva um programa Python que mostre uma tabela de conversão de temperaturas em graus Celsius e graus Fahrenheit. A tabela deve incluir em suas linhas todas as temperaturas entre 0 e 100 graus Celsius que sejam múltiplas de 10 graus Celsius. Inclua os cabeçalhos apropriados e tabulações para suas colunas. Pesquise na internet sobre a fórmula de conversão de temperaturas Celsius para Fahrenheit.

   **Dica**: dentro das strings na função print você pode usar o caracter especial “\t”, que insere um espaço de tabulação na string (equivalente a usar a tecla “tab” quando digitamos um texto).

4. **Perímetro de um polígono**. Crie um programa Python para calcular o perímetro de um polígono sendo fornecidas as coordenadas x e y de cada um de seus vértices. Inicie lendo x e y do primeiro vértice. Depois disso continue lendo x e y dos próximos vértices até que o usuário entre com uma linha em branco para o valor da coordenada x (ou seja, quando ele digitar “Enter" ou “Return" sem fornecer um valor). Cada vez que você ler as coordenadas de um novo vértice, você deve calcular a distância em relação ao vértice anterior e acrescentá-la ao valor do perímetro. A figura abaixo ilustra como se calcula a distância entre dois pontos sendo dadas suas coordenadas x e y.




   Quando o usuário entrar com a linha em branco na coordenada x, seu programa deve adicionar ao perímetro a distância do último ponto até o primeiro. Depois disso, deve exibir o valor do perímetro. Um exemplo de entrada é mostrado abaixo (o valores digitados pelo usuário estão em negrito.

```
Digite a coordenada x de um ponto: 0
Digite a coordenada y de um ponto: 0
Digite a coordenada x de um ponto (enter para sair): 1
Digite a coordenada y de um ponto: 0
Digite a coordenada x de um ponto (enter para sair): 0
Digite a coordenada y de um ponto: 1
Digite a coordenada x de um ponto (enter para sair):
O perímetro deste polígono é igual a 3.41421356237309
```

5. **Valor das entradas**. Um determinado zoológico estipula o valor da entrada baseado na idade do visitante. Visitantes com até dois anos de idade não precisam pagar. Crianças entre 3 e 12 anos de idade pagam R$ 14.00. Idosos com 65 anos ou mais pagam R$ 18.00. Todos os demais pagam R$ 23.00. Crie um programa que inicia lendo as idades, uma por uma, de um grupo de pessoas. O usuário deve entrar uma linha em branco para indicar que não há mais pessoas no grupo. Depois disso, seu programa deve exibir uma mensagem informando o preço total de todas as entradas para o grupo. O valor deve ser exibido com duas casas decimais.

6. **Bits de paridade**. Um bit de paridade é um mecanismo para detecção de erros em dados transmitidos por uma conexão não confiável, como linha telefônica por exemplo. A idéia básica é que, a cada grupo de 8 bits, seja acrescentado um bit adicional de forma que erros em bits individuais possam ser detectados.

   Os bit de paridade podem ser computados para paridade par ou paridade ímpar. Se for usada paridade par, então o bit de paridade a ser transmitido deve ser tal que o número total de bits “1” transmitidos (8 bits de dados + 1 bit de paridade) é par. Se for utilizada paridade ímpar, o número total de bits “1” transmitidos deve ser ímpar.

   Escreva um programa Python que compute o bit de paridade para grupos de 8 bits fornecidos pelo usuário utilizando paridade par. Seu programa deve ler strings contendo 8 bits (portanto as strings vai ser sequencias de 8 caracteres 0 ou 1) até que o usuário entre com uma linha em branco. Logo após o usuário fornecer cada string, seu programa deve exibir uma

mensagem informando se o bit de paridade deve ser 0 ou 1. O programa também deve exibir uma mensagem de erro caso o usuário entre com algo que não seja a sequência de 8 bits

> **Dica**: você deve ler o impute do usuário como uma string. Então, você pode usar o método **count** do tipo string para determinar a quantidade de zeros e uns na string. Você pode encontrar informação sobre o método **count** na internet.

7. **Aproximação do valor de $\pi$**. O valor aproximado de  pode ser calculado pela série infinita apresentada abaixo:

$$\pi \approx 3 + \frac{4}{2 \times 3 \times 4} - \frac{4}{4 \times 5 \times 6} + \frac{4}{6 \times 7 \times 8} - \frac{4}{8 \times 9 \times 10} + \frac{4}{10 \times 11 \times 12} - \ldots$$

   Escreva um programa Python que exiba 15 aproximações de $\pi$. A primeira aproximação deve ter apenas o primeiro termo da série (ou seja, o valor resultante vai ser somente 3). Cada nova aproximação de  mostrada pelo seu programa deve incluir mais um termo da série, sendo cada vez uma aproximação mais precisa do que a anterior.

8. **Cifra de César**. Um dos primeiros exemplos conhecidos de criptografia foi utilizado pelo imperador romano Julio César. César precisava fornecer instruções por escrito para seus generais, mas não queria que seus inimigos descobrissem suas estratégias caso as mensagens com as instruções fossem extraviadas. Com isso, ele acabou desenvolvendo o que mais tarde ficou conhecido como a “cifra de César”.

   A idéia por trás desta cifra é simples (e portanto não oferece proteção contra as técnicas modernas de quebra de códigos). Cada letra da mensagem original é deslocada em 3 posições. Com isso, a letra A se torna letra D, B se torna E, C se torna F, e assim por diante. A ultimas 3 letras do alfabeto são transformadas nas primeiras. X se torna A, Y se torna B e Z se torna C. Caracteres que não são letras não são convertidos.

   Escreva um program Python que implemente a cifra de César. Permita que o usuário forneça a mensagem e a distância de deslocamento de letras (portanto não será o valor fixo de deslocamento de 3 letras no alfabeto). Certifique-se que seu programa codifique corretamente tanto letras maiúsculas quanto minúsculas. Seu programa também deve suportar valores negativos de deslocamento de letras, assim ele pode ser usado tanto para codificar quanto para decodificar mensagens.

> **Dica**: você pode usar as funções ord(c) e chr(n) do Python. A função ord(c) retorna um número inteiro que representa o caracter Unicode c. A função chr(n) retorna o caracter Unicode representado pelo número inteiro n.

9. **Raiz quadrada.** Escreva um programa Python que implemente o método de Newton para calcular e exibir o valor da raiz quadrada de um número. O método de Newton é descrito pelo pseudo-código abaixo:

   Leia o valor de x do usuário
   Inicialize raiz = x/2
   **Enquanto** raiz não é boa o suficiente, **faça**
       Atualize raiz para receber a média entre raiz e raiz/2

   Quando o algoritmo chega ao fim, raiz contém um valor aproximado da raiz quadrada de x. A qualidade desta aproximação depende de como você define “boa o suficiente”. Podemos, por exemplo, considerar a solução boa o suficiente quando o valor absoluto da diferença entre raiz * raiz e x é menor que $10^{-12}$.

10. **Palíndromo**. Uma string é considerada um palíndromo se, de trás para frente, ela for idêntica à string original. Por exemplo: “arara”, “osso”, “radar”. Escreva um programa Python usando laço de repetição para determinar se uma palavra fornecida pelo usuário é ou não é um palíndromo. Seu programa deve exibir uma mensagem informando o resultado.

11. **Palíndromos com múltiplas palavras**. O conceito de palíndromo também pode ser aplicado a frases, por exemplo: “A base do teto desaba”. Faça um novo programa Python que modifique o programa do exercício anterior para verificar se frases são palíndromos. Seu programa vai precisar ignorar os espaços em brando das frases. Como desafio adicional, amplie sua solução para que também ignore sinais de pontuação e trate letras maiúsculas e minúsculas como equivalentes.

12. **Tabela de multiplicação**. Neste exercício você deve criar uma tabela de multiplicação mostrando os produtos de todos os inteiros de 1 vezes 1 até 10 vezes 10. Sua tabela deve incluir uma linha de cabeçalho com números de 1 a 10, e também uma coluna com os mesmos números. A saída esperada do programa deve ser semelhante ao mostrado abaixo:

```
    1   2   3   4   5   6   7   8   9  10
1   1   2   3   4   5   6   7   8   9  10
2   2   4   6   8  10  12  14  16  18  20
3   3   6   9  12  15  18  21  24  27  30
4   4   8  12  16  20  24  28  32  36  40
5   5  10  15  20  25  30  35  40  45  50
6   6  12  18  24  30  36  42  48  54  60
7   7  14  21  28  35  42  49  56  63  70
8   8  16  24  32  40  48  56  64  72  80
9   9  18  27  36  45  54  63  72  81  90
10 10  20  30  40  50  60  70  80  90 100
```

    Ao desenvolver seu programa, talvez você ache conveniente exibir um valor com print sem mover para a próxima linha. Isso pode ser feito adicionando-se **end=""** como último parâmetro da função print (Exemplo: print(x, end=“”)).

13. **Fatoração numérica**. A fatoração de um número inteiro n pode ser feita por meio de números primos de acordo com o procedimento descrito abaixo:

```
Inicialize fator com valor 2
Enquanto fator for menor ou igual a n, faça
       Se n for divisível por fator então
            Concluímos que fator faz parte da fatoração de n
            Faça divisão inteira de n por fator
      Senão
            Incremente fator em uma unidade
```

    Escreva um programa Python que lê um número inteiro do usuário. Se o valor fornecido pelo usuário é menor do que 2, seu programa deve exibir uma mensagem de erro. Caso contrário, seu programa deve exibir os números primos que podem ser multiplicados para formar o número *n*, com um fator exibido em cada linha. Por exemplo:

```
Digite um número inteiro (maior ou igual a 2): 72
2
2
2
3
3
```

14. **Binário para decimal**. Escreva um programa Python que converte um número binário (base 2) para decimal (base 10). Seu programa deve iniciar lendo um número binário como uma string. Então, ele deve computar o número decimal equivalente processando cada dígito do número binário. Finalmente

seu programa deve exibir uma mensagem informando o número decimal calculado.

15. Decimal para binário. Escreva um programa Python que converte um número decimal (base 10) para o correspondente número binário (base 2). Leia o número decimal como um número inteiro fornecido pelo usuário. Depois disso, use o algoritmo de divisão mostrado abaixo para fazer a conversão. Quando o algoritmo terminar, a variável *result* contém a representação binária do número. Ao final exiba uma mensagem informando o valor de *result*.

```
Inicialize result como uma string vazia
Seja q o número decimal a ser convertido
Repita
   r recebe o resto da divisão de q por 2
   converta r para uma string e adicione no início de result
   faça a divisão inteira de q por 2 (descartando o resto) e guarde o resultado em q
Até que q seja igual a zero
```

16. Cara ou coroa. Qual é o menor número de vezes que você precisa sortear uma moeda para ter três resultados consecutivos iguais de cara ou de coroa? Qual é o número máximo de sorteios necessários? Quantos sorteios precisamos em média? Neste exercício vamos explorar estas questões criando um programa que simula várias séries de sorteios de cara ou coroa.

    Crie um programa que utilize o gerador de números randômicos do Python para simular o sorteio de uma moeda várias vezes. A moeda simulada deve ser justa, ou seja, deve ter a mesma probabilidade de gerar cara e coroa. Seu programa deve continuar simulando sorteios até que ocorra uma sequencia de 3 caras ou de 3 coroas consecutivas. Exiba A para cada vez que ocorrer cara e O para cada vez que ocorrer coroa, com todos os resultados sendo exibidos na mesma linha. Então exiba a quantidade de sorteios que levou para chegar a três resultados iguais consecutivos. Quando seu programa executar, ele deve fazer esta simulação 10 vezes e, ao final, mostrar a quantidade média de sorteios necessária. Abaixo segue um exemplo de saída do programa:

```
A O O O (4 sorteios)
A A O O A O A O O A A O A O O A O O O (19 sorteios)
O O O (3 sorteios)
O A A A (4 sorteios)
A A A (3 sorteios)
O A O O A O A A O O A A O A O A A A (18 sorteios)
A O O A A A (6 sorteios)
A O A A A (5 sorteios)
O O A O O A O A O A A A (12 sorteios)
O A O O O (5 sorteios)
Na média, foram necessários 7.9 sorteios.
```