Roteiro 4 - Estruturas de Repetição

Certos tipos de problemas podem ser resolvidos com sequências de instruções executadas apenas uma vez, porém outros algoritmos requerem a execução de determinados trechos de código várias vezes, por isso o surgimento das estruturas de repetição.

Uma estrutura de repetição permite que uma sequência de instruções seja executada várias vezes até que uma condição seja satisfeita. Por isso, para saber quando utilizar é necessário analisar se uma mesma sequência de instruções necessita ser executada várias vezes no programa.

Exemplo: Imagine por exemplo um algoritmo que tenha que escrever os 1000 primeiros números positivos sem utilizar estrutura de repetição.

```
package javaapplication23;
public class JavaApplication23 {
    public static void main(String[] args) {
        System.out.println(1);
        System.out.println(2);
        System.out.println(3);
        System.out.println(4);
        System.out.println(5);
        ...
        System.out.println(1000);
    }

}
```

O código exemplificado acima é uma solução não prática para resolução do problema de imprimir os 1000 primeiros números positivos, percebe-se a repetição da tarefa de impressão dos números. Nesse caso é viável a utilização de estruturas de repetição.

Estrutura de repetição while

O **while** é uma estrutura de repetição que pode ser utilizada quando o número de repetições necessárias não é fixo. Os comandos serão repetidos até uma **condição** assumir o valor **falso**. Nesse tipo de estrutura, o **teste condicional** ocorre no **início** do bloco de comandos. Isto significa que existe a possibilidade da repetição **não ser executada**, ou seja, quando a condição assumir valor falso logo na primeira verificação. A sintaxe geral do comando é dada como se segue:

```
while (condição)
    {
      Bloco de comandos;
    }
```


Elaboração Prof. Bruno Nonato Gomes

Estrutura de repetição do - while

O **do-while** é uma estrutura de repetição que pode ser utilizada quando o número de repetições necessárias não é fixo. Os comandos serão repetidos até uma **condição** assumir o valor **falso**. Nesse tipo de estrutura, o **teste condicional** ocorre no **fim** do bloco de comandos. Isto significa que o bloco de comandos da repetição será executado pelo menos uma vez, ou seja, quando o bloco for executado e, ao final, a condição assumir o valor falso. A sintaxe geral do comando é dada como se segue:

```
do{
      Bloco de comandos;
  } while (condição);
```

Estrutura de repetição for

Essa estrutura de repetição é utilizada quando se conhece o número de vezes que o bloco de comandos deve ser repetido, ou seja, o número de repetições é definido. A sintaxe geral do comando **for** é composta por 3 fases: inicialização da variável de controle; teste da condição de parada; atualização ou incremento da variável de controle.

```
for (i = valor inicial; condição; incremento ou decremento de i)
     {
       Bloco de comandos;
     }
```

Exemplos

Exemplo 1: Digite e compile o exemplo abaixo que imprime uma sequência de números enquanto o valor de **x** for menor que o valor de **y**.

```
1   package javaapplication23;
2   import java.util.Scanner;
3   public class JavaApplication23 {
4       public static void main(String[] args) {
5           int x, y;
6           Scanner in = new Scanner(System.in);
7           x = in.nextInt();
8           y = in.nextInt();
9           while(x<y){
10              System.out.println(x+" "+y);
11              x+=2;
12              y+=1;
13          }
14      }
15  }
```

Faça testes com o código do exemplo acima (mude valores dos incrementos de x e y; altere o teste condicional; retire os comandos de impressão para fora do bloco de repetições) e veja o que acontece.

Exemplo 2: O exemplo 2 é similar ao exemplo 1, porém utiliza a estrutura de repetição **do-while**.

```
1   package javaapplication23;
2   import java.util.Scanner;
3   public class JavaApplication23 {
4       public static void main(String[] args) {
5           int x, y;
6           Scanner in = new Scanner(System.in);
7           x = in.nextInt();
8           y = in.nextInt();
9           do{
10              System.out.println(x+" "+y);
11              x+=2;
12              y+=1;
13          }while(x<y);
14      }
15  }
```

Responda: Os códigos sempre retornam o mesmo resultado? Justifique.
Faça os mesmos testes do exemplo 1. Digite os valores x = 7 e y = 5 nos dois exemplos. Veja que o resultado não será o mesmo.

Exemplo 3: Digite e compile o código abaixo que solicita a digitação de números ao usuário até que seja digitado um número menor ou igual à zero. Utilização da estrutura **do-while**.

```
1   package javaapplication23;
2   import java.util.Scanner;
3   public class JavaApplication23 {
4       public static void main(String[] args) {
5           int num;
6           Scanner in = new Scanner(System.in);
7           do{
8               num = in.nextInt();
9               System.out.println("Número digitado "+num);
10          }while(num>0);
11      }
12  }
```

Exemplo 4: Digite e compile o código abaixo que tem o objetivo de receber **N** valores digitados pelo usuário, imprimindo ao final a quantidade de valores digitados e a soma dos mesmos.

```
1   package javaapplication23;
2   import java.util.Scanner;
3   public class JavaApplication23 {
4       public static void main(String[] args) {
5           float soma = 0, numero;
6           int N;
7           Scanner in = new Scanner(System.in);
8           System.out.println("Digite o quantos valores deseja somar:");
9           N = in.nextInt();
10          for(int i=1; i<=N; i=i+1){
11              System.out.println("Digite o "+i+"° número:");
12              numero = in.nextFloat();
13              soma += numero;
14          }
15          System.out.println("Foram digitados "+N+" valores!");
16          System.out.println("A soma dos valores digitados é: "+soma);
17      }
18  }
```

Exemplo 5: O exemplo 5 mostra a utilização da estrutura **for** com múltiplas inicializações, testes e incrementos/decrementos. Execute o código e veja qual será o resultado exibido.

```
1   package javaapplication23;
2   import java.util.Scanner;
3   public class JavaApplication23 {
4       public static void main(String[] args) {
5           for(int i=1, j=10; i<=10 && j>=1; i++, j--){
6               System.out.println(i+" "+j);
7           }
8       }
9   }
```

Exemplo 6: O exemplo 5 mostra a utilização de estruturas de repetição aninhadas, neste caso a estrutura **for**. Execute o código e veja qual será o resultado exibido.

```
1   package javaapplication23;
2   public class JavaApplication23 {
3       public static void main(String[] args) {
4           for(int i=1; i<=10; i++){
5               for(int j=1; j<=5; j++){
6                   System.out.println(i+" "+j);
7               }
8           }
9       }
10  }
```

Exercícios

1. Implemente um algoritmo que imprima todos os números inteiros do intervalo fechado de 1 a 100.

2. Escreva um programa que imprima os números pares de 0 à 50. Utilize a estrutura **for.**

3. Escreva um algoritmo que imprima os números de 100 à 1 **(ordem decrescente).**

4. Escreva um algoritmo que receba números do usuário e imprima o triplo de cada número. O algoritmo deve encerrar ao ser digitado o número –999. **Obs.: O triplo de -999 não deve ser exibido.**

5. Faça um programa que calcule e imprima a soma e a média dos 10 primeiros números positivos. **Soma = 1 + 2 + 3 + ... + 10**

6. Faça um programa que calcule e imprima a soma e a média dos N primeiros números positivos. **Soma = 1 + 2 + 3 + ... + N**

7. Faça um algoritmo que imprima a soma da sequência apresentada: H = 1/1 + 1/2 + 1/3 + ... + 1/N. O valor de N deve ser positivo e fornecido pelo usuário.

8. Faça um algoritmo que imprima a soma da sequência apresentada: H = 1 - 1/2 + 1/3 – 1/4 + 1/5... 1/N. O valor de N deve ser positivo e fornecido pelo usuário.

9. Faça um programa que calcule e imprima a soma e a média de **N** números digitados pelo usuário. O valor de **N** deve ser fornecido pelo usuário.

10. Implemente um algoritmo que receba de entrada **N** valores digitados pelo usuário. Dentre os valores digitados, seu programa deve encontrar o menor e o maior dos valores fornecidos.

11. Faça um programa que leia a nota e o nome de **N** alunos na prova de algoritmos e imprima a maior e a menor nota computada e qual aluno tirou tais notas. Além disso, calcule e imprima também a soma e a média de todas as notas.

12. No último ano foi realizada um estudo estatístico sobre acidentes de trânsito em 5 cidades brasileiras. Para isso os seguintes dados foram coletados:
    a) Nome da cidade (String ou char).
    b) Número de veículos
    c) Número de acidentes de trânsito

    Com esses dados deseja-se saber:
    a) O maior e o menor índice de acidentes e o nome da cidade a que pertencem
    b) A razão entre quantidade de acidentes por quantidade de veículos nas 5 cidades analisadas
    c) A média de veículos nas cinco cidades
    d) A média de acidentes de trânsito nas cidades com menos de 200 veículos

13. Faça um algoritmo para identificar se um determinado número fornecido pelo usuário é primo ou não. Lembrando que um número primo só é divisível por 1 e por ele mesmo, ou seja, possui somente 2 divisores. Exemplos: 2, 5, 7, etc.

14. Faça um programa que calcule e imprima o fatorial do valor **N.** O valor de N será fornecido de entrada pelo usuário. O fatorial de um número positivo **N** qualquer (representado por **N!**) é o produto de todos os inteiros positivos menores ou iguais à **N,** ou seja, **Fatorial (N) = 1 * 2 * 3 * ... * N.** Por exemplo, o fatorial de **5 = 1*2*3*4*5 = 120.**

15. Faça um programa em JAVA que imprima os **N** primeiros termos da serie de Fibonacci. Sabe-se que N é fornecido pelo usuário. Fibonacci = 1, 1, 2, 3, 5, 8, 13, 21, ...

16. Elabore um algoritmo que faça a conversão de um número binário digitado pelo usuário para o número na base octal, base decimal e base hexadecimal. **Valide se o valor informado na entrada só possui 0's ou 1's, ou seja, se é mesmo um número binário**. **Não utilize funções pré-definidas no JAVA.**

17. Escreva um algoritmo que calcule o m.d.c. (máximo divisor comum) entre A e B (número inteiros e positivos). Esses dois valores são passados pelo usuário através do teclado. **Utilize a lógica do algoritmo de Euclides.**

18. Faça um algoritmo que simule o funcionamento de uma calculadora que contenha as operações aritméticas básicas com dois números digitados pelo usuário. O programa implementado deve mostrar seguinte menu ao usuário. Não se esqueça de verificar se as operações podem ser realizadas.

```
====================
Calculadora de Fulano
====================
  Opções:
        1 - Soma
        2 - Subtração
        3 - Multiplicação
        4 - Divisão
        5 – Sair
====================
```

19. Faça um programa que deve solicitar números para o usuário até que seja digitado -1. Quando o usuário digitar -1, o programa termina e imprime a média de todos os números positivos digitados.

20. Construa um algoritmo para calcular a média de valores PARES e ÍMPARES, que serão digitados pelo usuário. Ao final o algoritmo deve mostrar estas duas médias bem como o maior número PAR e o menor número ÍMPAR digitado. O algoritmo finaliza quando o usuário digitar um valor negativo.

21. Implemente um programa que receba de entrada um número inteiro qualquer, após isso verifique se o número inserido é ou não um PALÍNDROMO, ou seja, o número é o mesmo tanto de visto da direita para esquerda quanto da esquerda para a direita. Ex: 121, 1441, 34643, etc. **Não utilize funções pré-definidas no JAVA.**

22. Sabe-se que um país A possui 500000 habitantes e uma taxa de natalidade de 3% ao ano, já o país B possui 700000 habitantes e uma taxa de natalidade de 2% ao ano. Escreva um algoritmo, sabendo que estamos no ano de 2015, que calcule em que ano a população do país A ultrapassará a população de B.

23. Considere uma linha ferroviária entre São Paulo e Curitiba. Suponha que uma locomotiva (trem) A parte de São Paulo para Curitiba com velocidade de 30 m/s enquanto que uma outra locomotiva B parte de Curitiba para São Paulo no mesmo instante com velocidade de 40 m/s. Considere a distância entre São Paulo e Curitiba de 400 Km. Considere também que as linhas férreas são paralelas. Implemente um algoritmo que calcule iterativamente o tempo em que as locomotivas se cruzarão no caminho. O algoritmo deve calcular também a distância que as locomotivas devem percorrer até o momento do cruzamento.

24. Uma Empresa de fornecimento de energia elétrica faz a leitura mensal dos medidores de consumo. Para cada consumidor, são digitados os seguintes dados:
    - Número do consumidor;
    - Quantidade de kWh consumidos durante o mês;
    - Tipo (código) do consumidor.
        1 – residencial, preço em reais por kWh = 0,3
        2 – comercial, preço em reais por kWh = 0,5
        3 – industrial, preço em reais por kWh = 0,7

    Os dados devem ser lidos até que seja encontrado um consumidor com Número 0 (zero). Escreva um programa que calcule e imprima:
    - O custo total para cada consumidor;
    - O total de consumo para os três tipos de consumidor;
    - A média de consumo dos tipos 1 e 2.

25. O valor aproximado de PI pode ser calculado usando os 51 primeiros termos da seguinte série:

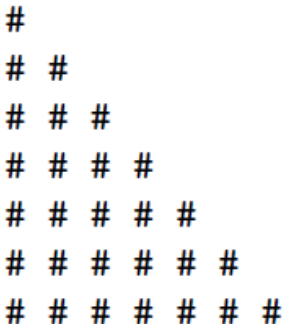
$$S = 1 - \frac{1}{3^3} + \frac{1}{5^3} - \frac{1}{7^3} + \frac{1}{9^3} \cdots$$

    Sendo PI = $\sqrt[3]{S \times 32}$. Sabendo disso, implemente um algoritmo que calcule e imprima o valor de PI utilizando a série apresentada.

26. Implementar um algoritmo para calcular o sen(X). Sabe-se que o valor de X deverá ser fornecido pelo usuário em graus. O valor do seno de X será calculado pela soma dos **15 primeiros termos** da série a seguir:

$$\text{sen}(X) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \ldots$$

27. Escreva um programa em JAVA que imprima o triângulo abaixo, em que a altura do triângulo (número de linhas) é fornecido pelo usuário.



28. Implemente um programa que solicite um valor inteiro positivo (**N**) ao usuário. Após isso imprima a tabuada dos números de 1 à N.

29. Suponha que JAVA possua **somente as operações de soma e subtração**. Dados dois números inteiros positivos A e B, determine o quociente e o resto da divisão de A por B.

30. A companhia de teatro do IFMG Sabará deseja realizar uma série de espetáculos. A direção calcula que a despesa fixa do espetáculo é de R$200,00. Além disso, sabe-se que com os ingressos ao preço de R$5,00 serão vendidos 120 entradas. Em uma pesquisa de público estimou-se que a cada R$0,50 de diminuição no valor do ingresso espera-se um aumento de 26 ingressos nas vendas. Diante dessas informações, implemente um programa que calcule e imprima uma tabela contendo os valores dos lucros esperados em função do valor dos ingressos, fazendo uma variação nos ingressos de R$5,00 à R$1,00 de R$0,50 em R$0,50. Escreva ainda o lucro máximo esperado, o preço do ingresso e quantidade de ingressos vendidos para obtenção desse lucro máximo estimado.

31. Em uma eleição presidencial existem quatro candidatos. Os votos são informados através de códigos. Os dados utilizados para a contagem dos votos obedecem à seguinte codificação:
    - 1, 2, 3, 4: voto para os respectivos candidatos;
    - 5: voto em branco;
    - Outros valores: voto nulo.

    Elabore um algoritmo que leia o código do candidato em um voto. Calcule e escreva:
    - total de votos para cada candidato;
    - total de votos nulos;
    - total de votos em branco;

    Como finalizador do conjunto de votos, tem-se o valor de código igual à 0.

32. Escrever um algoritmo que gera e escreve os 5 primeiros números perfeitos. Um número perfeito é aquele que é igual a soma dos seus divisores. (Ex.: 6 = 1+2+3; 28= 1+2+4+7+14 etc).

33. O número de inscrição no CPF é composto de onze dígitos decimais, sendo os oito primeiros aleatoriamente designados no momento da inscrição. Já o nono (antepenúltimo) dígito indica a região fiscal responsável pela inscrição (MG é a região 6, portanto em todos CPF's emitidos em MG o nono digito é 6). Por fim, o décimo e o décimo primeiro são dígitos verificadores calculados de acordo com um algoritmo definido pela Receita Federal e publicamente conhecido. Referências abaixo:
    a. http://www.gerardocumentos.com.br/?pg=entenda-a-formula-do-cpf
    b. https://bityli.com/CI4gC

Assim, implemente um algoritmo que receba de entrada 11 dígitos de um CPF qualquer (uma única variável inteira) e verifique se o valor inserido é ou não um CPF válido. Utilize as referências citadas anteriormente para realizar os cálculos para validação. **Utilize estrutura de repetição.**