



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática
Trabalho Prático 1

Curso : *Engenharia de Software*
Disciplina : *Algoritmos e Estruturas de Dados II*
Professora : *Eveline Alonso Veloso*

Regras Básicas:

1. Estude bastante cada par de entrada/saída.
2. Todos os programas deverão ser desenvolvidos na linguagem de programação especificada em seu enunciado (C ou Java).
3. Esse trabalho prático poderá ser desenvolvido em grupos de, no máximo, três integrantes.
4. Cópias de trabalho, se existirem, serão encaminhadas ao colegiado de coordenação didática do curso.
5. Neste trabalho prático, vocês não poderão usar os métodos da classe *String*, a não ser os métodos: *char charAt(int)*, *boolean equals(String)* e *int length()*. Outras classes de Java que manipulam *strings* não são permitidas nesse momento.
6. Para contar as letras, consoantes e vogais desconsiderem acentos e cedilha, ou seja, considerem somente os caracteres cujos códigos ASCII estejam entre 'A' e 'Z' ou 'a' e 'z'.
7. Fique atento ao *charset* dos arquivos de entrada e saída. Recomenda-se a utilização dos métodos da classe *MyIO.java* para leitura de dados do teclado e escrita de dados na tela do computador.
8. Para cada exercício, vocês devem submeter apenas um arquivo (.java ou .cpp) por grupo. Essa regra será necessária para a submissão de trabalhos no VERDE e no identificador de plágios utilizado na disciplina.
9. A resolução (código) de cada exercício deverá ser submetida ao VERDE.
10. A correção será realizada automaticamente pelo VERDE e validada por meio de apresentações durante as aulas práticas da disciplina.

Dicas sobre Recursividade:

1. Um método recursivo pode ser utilizado como uma estrutura de repetição e um erro básico ao se utilizar recursividade consiste em confundir tais estruturas com os métodos recursivos. Para evitar esse erro, neste trabalho, vocês não poderão usar estruturas de repetição nos métodos recursivos.
2. Outro erro comum consiste em criar variáveis globais (em Java, atributos) para darem suporte à recursividade. Para evitar esse erro, neste trabalho, vocês não poderão usar variáveis globais.
3. Um terceiro erro comum consiste em passar o valor de retorno como parâmetro para o método recursivo. Para evitar esse erro, neste trabalho, vocês não poderão passar parâmetros extras aos do enunciado, exceto um contador para a recursividade.

O método abaixo apresenta um exemplo desse erro.

```
int contarLetrasMinusculas(String s, int i, int resp){  
  
    int contar;  
  
    if (i == s.length())  
        contar = resp;  
    else if (s.charAt(i) >= 'a' && s.charAt(i) <= 'z')  
        contar = contarLetrasMinusculas (s, i + 1, resp + 1);  
    else  
        contar = contarLetrasMinusculas (s, i + 1, resp) ;  
  
    return contar ;  
}
```

4. Quando criamos um método recursivo, normalmente, passamos um contador como parâmetro. Uma forma elegante de implementar essa solução recursiva consiste em criar dois métodos: um recursivo, privado; e outro, público, que chama o método recursivo pela primeira vez e inicializa o valor do contador, encapsulando, dessa maneira, a utilização da recursividade. Neste trabalho, devemos usar essa técnica em todos os métodos recursivos. Veja o exemplo abaixo:

```
public boolean somenteLetrasMinusculas(String s) {  
    return somenteLetrasMinusculas (s, 0);  
}  
  
private boolean somenteLetrasMinusculas(String s, int i) {  
  
    boolean resp ;  
  
    if (i == s.length())  
        resp = true;  
    else if ((s.charAt(i) >= 'a' && s.charAt(i) <= 'z') == false)  
        resp = false;  
    else  
        resp = somenteLetrasMinusculas (s, i + 1) ;  
  
    return resp;  
}
```

5. Uma boa prática de programação consiste em empregar um único *return* em cada método. Por exemplo, o método anterior é mais indicado que a versão abaixo:

```

public boolean somenteLetrasMinusculas(String s) {
    return somenteLetrasMinusculas (s, 0);
}

private boolean somenteLetrasMinusculas(String s, int i) {

    if (i == s.length())
        return true;
    else if ((s.charAt(i) >= 'a' && s.charAt(i) <= 'z') == false)
        return false;
    else
        return somenteLetrasMinusculas (s, i + 1) ;
}

```

Exercícios:

1. Ciframento de César Iterativo em Java

O Imperador Júlio César foi um dos principais nomes do Império Romano. Entre suas contribuições, temos um algoritmo de criptografia chamado "Ciframento de César". Segundo os historiadores, César utilizava esse algoritmo para criptografar as mensagens que enviava aos seus generais durante as batalhas. A ideia básica é um simples deslocamento de caracteres. Assim, por exemplo, se a chave utilizada para criptografar as mensagens for três, todas as ocorrências do caractere 'a' serão substituídas pelo caractere 'd', as do 'b' por 'e', e assim sucessivamente. Crie um método iterativo que receba uma *string* como parâmetro e retorne outra correspondente à entrada de forma cifrada. Neste exercício, suponha a chave de ciframento três. A entrada termina com a leitura de "FIM". Na saída padrão, para cada linha de entrada, escreva uma linha com a mensagem criptografada.

2. Palíndromo Iterativo em Java

Crie um método iterativo que receba uma *string* como parâmetro e retorne *true* se essa for um palíndromo. A entrada do programa termina com a leitura de "FIM". Na saída padrão, para cada linha de entrada, escreva uma linha de saída com SIM/NAO indicando se a entrada corresponde a um palíndromo ou não. Destaca-se que uma linha de entrada pode conter caracteres não-letas.

3. Alteração Aleatória em Java

Crie um método iterativo que receba uma *string*, sorteie duas letras minúsculas aleatórias (código ASCII \geq 'a' e \leq 'z'), substitua todas as ocorrências da primeira letra na *string* pela segunda e retorne a *string* com as alterações efetuadas. Na saída padrão, para cada linha de entrada, execute o método desenvolvido nesta questão e mostre a *string* retornada como uma linha de saída. Abaixo, observamos um exemplo de entrada supondo que para a primeira linha as letras sorteadas foram o 'a' e o 'q'. Para a segunda linha, foram o 'e' e o 'k'.

EXEMPLO DE ENTRADA:

o rato roeu a roupa do rei de roma
e qwe qwe qwe ewq ewq ewq
FIM

EXEMPLO DE SAÍDA:

o rqto roeu q roupq do rei de romq
k qwk qwk qwk kwq kwq kwq

A classe `Random`, da hierarquia de classes de Java, gera números (ou letras) aleatórios e o exemplo abaixo mostra uma letra minúscula na tela. Em especial, destacamos que:

- i) *seed* é a semente para geração de números aleatórios;
- ii) nesta questão, por causa da correção automática, *seed* deverá ser quatro.

```
1 Random gerador = new Random();  
2 gerador.setSeed(4);  
3 System.out.println((char)('a' + (Math.abs(gerador.nextInt()) % 26)));
```

4. Is Iterativo em Java

Crie um método iterativo que receba uma *string* e retorne *true* se a mesma for composta somente por vogais. Crie outro método iterativo que receba uma *string* e retorne *true* se a mesma for composta somente por consoantes. Crie um terceiro método iterativo que receba uma *string* e retorne *true* se a mesma corresponder a um número inteiro. Crie um quarto método iterativo que receba uma *string* e retorne *true* se a mesma corresponder a um número real. Na saída padrão, para cada linha de entrada, escreva outra de saída da seguinte forma X1 X2 X3 X4 onde cada Xi é um booleano indicando se a entrada é: composta somente por vogais (X1); composta somente por consoantes (X2); um número inteiro (X3); um número real (X4). Se Xi for verdadeiro, seu valor deverá ser SIM; caso contrário, NAO.

5. Álgebra Booleana Iterativa em Java

Crie um método iterativo que receba uma *string* contendo uma expressão booleana e o valor de suas entradas e retorne um número inteiro indicando se a expressão é verdadeira ou falsa. Cada *string* de entrada é composta por um número inteiro *n*, que indica o número de entradas da expressão booleana corrente. Em seguida, a *string* contém *n* valores binários (um para cada entrada) e a expressão booleana. Na saída padrão, para cada linha de entrada, escreva uma linha de saída com 1 ou 0, indicando se a expressão corrente é verdadeira ou falsa.

6. Ciframento de César Recursivo em Java

Refaça a questão Ciframento de César Iterativo em Java de forma recursiva.

7. Palíndromo Recursivo em Java

Refaça a questão Palíndromo Iterativo em Java de forma recursiva.

8. Is Recursivo em Java

Refaça a questão Is Iterativo em Java de forma recursiva.

9. Álgebra Booleana Recursiva em Java

Refaça a questão Álgebra Booleana Iterativa em Java de forma recursiva.

10. Palíndromo Iterativo em C

Refaça a questão Palíndromo Iterativo em Java na linguagem C.

11. Palíndromo Recursivo em C

Refaça a questão Palíndromo Iterativo em C de forma recursiva.