# Unidade 0 - Nivelamento - Exercícios de Revisão


Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação

# Exercício

- Faça um método que receba um *array* de inteiros e um número inteiro *x* e retorne um valor booleano informando se o elemento *x* está contido no *array*

- Repita o exercício acima considerando que os elementos do *array* estão ordenados de forma crescente. Uma sugestão seria começar a pesquisa pelo meio do *array*

# Exercício

- Faça um método que receba um *array* de inteiros e mostre na tela o maior e o menor elementos do *array*.

- Repita o exercício acima fazendo menos comparações com os elementos do *array*

# Exercício

- O que o código abaixo faz?

```
boolean doidao (char c){
      boolean resp= false;
      int v = (int) c;
      if (v == 65 || v == 69 || v == 73 || v == 79 || v == 85 || v == 97 || v == 101 || v ==105 ||
          v == 111 || v == 117){
            resp = true;
      }
      return resp;
}
```

A função verifica se o valor numérico (código ASCII) do caractere c corresponde a uma das vogais maiúsculas (A, E, I, O, U) ou vogais minúsculas (a, e, i, o, u). Se o valor numérico do caractere corresponder a alguma dessas vogais, a função retorna true, caso contrário, ela retorna false.

# Exercício

- O que o código abaixo faz?

```
boolean doidao (char c){
      boolean resp= false;
      int v = (int) c;
      if (v == 65 || v == 69 || v == 73 || v == 79 || v == 85 || v == 97 || v == 101 || v ==105 ||
          v == 111 || v == 117){
            resp = true;
      }
      return resp;
}

char toUpper(char c){
      return (c >= 'a' && c <=  'z') ? ((char) (c – 32)) : c ;
}

boolean isVogal (char c){
      c = toUpper(c);
      return (c =='A' || c =='E' ||  c =='I' ||  c =='O' || c =='U');
}
```

Fazendo o isVogal, fica mais fácil ...

# Exercício

- Outras opções

```
boolean isLetra (char c){
      return (c >= 'A' && c <='Z' ||  c >='a' &&  c <='Z');
}

boolean isConsoante (char c){
      return (isLetra(c) == true &&  isVogal(c) == false);
}
```

# Exercício Resolvido

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n){
    boolean resp= true;
    if (n!=s.length()){
      if (s.charAt(n)<'0' || s.charAt(n)>'9'){
        if (s.charAt(n)=='A' || s.charAt(n)=='E' || s.charAt(n)=='I' || s.charAt(n)=='O' ||
          s.charAt(n)=='U' || s.charAt(n)=='a' || s.charAt(n)=='e' || s.charAt(n)=='i' ||
          s.charAt(n)=='o' || s.charAt(n)=='u'){
          resp= false;
        } else{
          n++;
          resp=isConsoante(s, n);
        }
      } else {
        resp=false;
      }
    }
    return resp;
  }
```

# Exercício Resolvido

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n){
    boolean resp= true;
    if (n!=s.length()){
      if (s.charAt(n)<'0' || s.charAt(n)>'9'){
        if (s.charAt(n)=='A' || s.charAt(n)=='E' || s.charAt(n)=='I' || s.charAt(n)=='O' ||
          s.charAt(n)=='U' || s.charAt(n)=='a' || s.charAt(n)=='e' || s.charAt(n)=='i' ||
          s.charAt(n)=='o' || s.charAt(n)=='u'){
          resp= false;
        } else{
          n++;
          resp=isConsoante(s, n);
        }
      } else {
        resp=false;
      }
    }
    return resp;
  }
```

1º passo: Indentação

# Exercício Resolvido

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n){
	boolean resp= true;
	if (n != s.length()){
		if (s.charAt(n)<'0' || s.charAt(n)>'9'){
			if (s.charAt(n)=='A' || s.charAt(n)=='E' || s.charAt(n)=='I' || s.charAt(n)=='O' ||
			  s.charAt(n)=='U' || s.charAt(n)=='a' || s.charAt(n)=='e' || s.charAt(n)=='i' ||
			  s.charAt(n)=='o' || s.charAt(n)=='u'){
				resp= false;
			} else {
				n++;
				resp=isConsoante(s, n);
			}
		} else {
			resp=false;
		}
	}
	return resp;
}
```

# Exercício Resolvido

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n) {    boolean resp = true;        if (n != s.length()) {        if
(s.charAt(n) < '0' || s.charAt(n) > '9') {            if (s.charAt(n) == 'A' || s.charAt(n) == 'E' || s.charAt(n)
== 'I' || s.charAt(n) == 'O' || s.charAt(n) == 'U' ||                s.charAt(n) == 'a' || s.charAt(n) == 'e' ||
s.charAt(n) == 'i' || s.charAt(n) == 'o' || s.charAt(n) == 'u') {                resp = false;            }        }
else {            resp = isConsoante(s, n + 1);        }    } else {        resp = false;    }        return resp;}
```

# Exercício Resolvido

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n){
        boolean resp= true;
        if (n != s.length()){
                if (s.charAt(n)<'0' || s.charAt(n)>'9'){
                        if (s.charAt(n)=='A' || s.charAt(n)=='E' || s.charAt(n)=='I' || s.charAt(n)=='O' ||
                          s.charAt(n)=='U' || s.charAt(n)=='a' || s.charAt(n)=='e' || s.charAt(n)=='i' ||
                          s.charAt(n)=='o' || s.charAt(n)=='u'){
                                resp= false;
                        } else {
                                n++;
                                resp=isConsoante(s, n);
                        }
                } else {
                        resp=false;
                }
        }
        return resp;
}
```

# Exercício Resolvido

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n){
      boolean resp= true;
      if (n != s.length()){
            if (s.charAt(n)<'0' || s.charAt(n)>'9'){
                  if (s.charAt(n)=='A' || s.charAt(n)=='E' || s.charAt(n)=='I' || s.charAt(n)=='O' ||
                    s.charAt(n)=='U' || s.charAt(n)=='a' || s.charAt(n)=='e' || s.charAt(n)=='i' ||
                    s.charAt(n)=='o' || s.charAt(n)=='u'){
                        resp= false;
                  } else {
                        n++;
                        resp=isConsoante(s, n);
                  }
            } else {
                  resp=false;
            }
      }
      return resp;
}
```

2º passo: Simplificação

# Exercício Resolvido

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n){
      boolean resp= true;
      if (n != s.length()){
            if (s.charAt(n)<'0' || s.charAt(n)>'9'){
                  if (s.charAt(n)=='A' || s.charAt(n)=='E' || s.charAt(n)=='I' || s.charAt(n)=='O' ||
                    s.charAt(n)=='U' || s.charAt(n)=='a' || s.charAt(n)=='e' || s.charAt(n)=='i' ||
                    s.charAt(n)=='o' || s.charAt(n)=='u'){
                        resp= false;
                  } else {
                        n++;
                        resp=isConsoante(s, n);
                  }
            } else {
                  resp=false;
            }
      }
      return resp;
}
```

2º passo: Simplificação

# Exercício Resolvido

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n){
      boolean resp= true;
      if (n != s.length()){
            if (s.charAt(n)<'0' || s.charAt(n)>'9'){
                  if (isVogal(s.charAt(n))  ==  true){
                        resp= false;
                  } else {
                        resp=isConsoante(s, n + 1);
                  }
            } else {
                  resp=false;
            }
      }
      return resp;
}
```

2º passo: Simplificação

# Exercício Resolvido

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n){
      boolean resp= true;
      if (n != s.length()){
            if (s.charAt(n)<'0' || s.charAt(n)>'9'){
                  if (isVogal(s.charAt(n))  ==  true){
                        resp= false;
                  } else {
                        resp=isConsoante(s, n + 1);
                  }
            } else {
                  resp=false;
            }
      }
      return resp;
}
```

O código está correto?

# Exercício Resolvido

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int i){
	boolean resp= true;

	if (i == s.length()){
		resp = true;
	} else if (isConsoante(s.charAt(i)) == false){
		resp = false;
	} else {
		resp = isConsoante(s, i + 1);
	}

	return resp;
}
```

# Exercício

- Qual das duas versões é mais fácil de entender?

```
boolean isConsoante(String s, int i){
        boolean resp= true;

        if (i == s.length()){
                resp = true;
        } else if (isConsoante(s.charAt(i)) == false){
                resp = false;
        } else {
                resp = isConsoante(s, i + 1);
        }

        return resp;
}
```

```
boolean isConsoante(String s, int i){
        boolean resp= true;

        if (i < s.length()){
                if (!isConsoante(s.charAt(i))){
                        resp = false;
                } else {
                        resp = isConsoante(s, i + 1);
                }
        } else {
                resp = true;
        }

        return resp;
}
```

O código mais fácil de entender é o primeiro.

# Exercício

- Qual é a sua opinião sobre o código **REAL** abaixo?

```
Unidade recuperarUnidadeComCodigoDeUCI(Unidade unidadeFilha) {
	Unidade retorno = null;

	if (unidadeFilha.getCodUci() != null && !unidadeFilha.getCodUci().isEmpty()) {
		retorno = unidadeFilha;
	} else {
		retorno = unidadeFilha.getUnidadeSuperior();
	}

	while (retorno == null || retorno.getCodUci() == null || retorno.getCodUci().isEmpty()) {
		retorno = retorno.getUnidadeSuperior();
	}

	return retorno;
}
```

O código acima possui variáveis com nomes desnecessariamente longos que poderiam ser otimizados.

# Exercício

- Qual é a diferença entre os dois métodos abaixo?

```
int m1(int i){
      return i--;
}

int m2(int i){
      return --i;
}
```

O método m1 retorna o valor original de i e, em seguida, decrementa i em uma unidade.O segundo método m2 decrementa i e, em seguida, retorna o valor decrementado, novamente, em uma unidade.

# Exercício

- O que o programa abaixo mostra na tela?

```
byte b = 0; short s = 0; int i = 0; long l = 0;

while (true){
      b++;    s++;    i++;    l++;
      System.out.println(b + " " + s + " " + i + " " + l);
}
```

O programa acima declara as variáveis b, s, i e l. Enquanto a condição true se encontra verdadeira, as variáveis são incrementadas em uma unidade. Caso contrário, nada acontece (com base no que se pode afirmar pelo código fornecido).

# Exercício Resolvido

- Por que o código abaixo imprime [46 - 11]?

```
int x = 23, y = 23;
x = x << 1;
y = y >> 1;
System.out.println("[" + x + " - " + y + "]");
```

A operação efetuada x = x << 1 realiza um deslocamento de bits para a esquerda em x. O valor inicial de x é 23, que, em binário, é representado como 10111. Ao deslocar os bits uma posição para a esquerda, obtemos 101110, que em decimal é 46. Desse modo, x é atualizado para 46. O mesmo ocorre com o y, com a pequena diferença que o deslocamento de bits é para direita - fazendo assim com que o valor designado a y passe a ser 11 ao invés de 23, como definido anteriormente.

# Exercício Resolvido

- Por que o código abaixo imprime [46 - 11]?

```
int x = 23, y = 23;
x = x << 1;
y = y >> 1;
System.out.println("[" + x + " - " + y + "]");
```

Os operadores *shift right* e *left* (>> e <<) deslocam os bits para direita e esquerda e inserem um zero na posição vazia

Na prática, temos, uma divisão ou multiplicação por dois

# Exercício Resolvido

- Por que o código abaixo imprime [46 - 11]?

```
int x = 23, y = 23;
x = x << 1;
y = y >> 1;
System.out.println("[" + x + " - " + y + "]");
```