Unidade00b_exercícios

Aluno (a): Thaís Ferreira da Silva
Curso: Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II
Turno: Manhã **Período:** 2º
Professor: Max do Val Machado

Exercício Página 4

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;
}
```

O método converte o caractere recebido por parâmetro em um inteiro (utilizando a tabela ascii) e verifica se o caractere em questão é uma vogal, maiúscula ou minúscula.

Exercício Página 5

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 a verificação de se é ou não vogal ficar mais fácil.

Exercício Página 7

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;
}
```

```
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 Página 16

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;
}
```

Para mim a mais fácil de se compreender inicialmente é a primeira versão

Exercício Página 17

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;
}
```

A minha opinião é que o código é confuso e redundante

Exercício Página 18

Qual é a diferença entre os dois métodos abaixo?

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

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

A diferença está na ordem dos processos. O m1 é decrescido após a operação, enquanto o m2 é decrescido antes da operação.

Exercício Página 19

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 mostra diversos tipos de dados, entre eles:

short: consome 2 bytes, range: 32.767
int: consome 4 bytes, range: 2,14*10a9
long: consome 8 bytes, range: 9.23*10a18
byte: consome 1 byte, range (valor máximo e mínimo): 127

Exercício Página 20

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

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

O programa imprime [46-11] pelo fato de usar o operador shift (<< e >>). Através dele os bits são deslocados para a direita ou para a esquerda. Desta forma:
O número 23 = 10111 em binário, se tornou 101110 = 46 (Deslocou uma casa para a esquerda). Depois o número 23 = 10111 em binário, se tornou 1011 = 11 (Deslocou uma casa para a direita).