Trabalho Teórico 2
Unidade 0

Iyan Lucas Duarte Marques

12 de agosto de 2020

1 Exercícios de revisão

1.1 Exercício p4

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.

```
boolean doidao (char n){
    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;
    }
}
```

1.2 Exercício p5

```
boolean doidao (char n){
    boolean resp= false;
    int v = (int) c;
    if (v == 65 || v == 69 || v == 73 || v == 79 || v == 85 || v == 97 || v == 101 || v ==
    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');
}
```

1.3 Exercício p7

O código simplificado:

```
import java.util.Scanner;

public class temp {
    public static 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;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String s = input.nextLine();
        System.out.println(isConsoante(s, 0));
    }
}
```

1.4 Exercício p16

A primeira versão é mais fácil de entender pelo fato de estar mais simplificada e concisa.

1.5 Exercício p17

A minha opinião é que o código é confuso e redundante

1.6 Exercício p18

A diferença entre os códigos é que, no m1, i é decrescido depois da operação se realizar. Em m2, i é decrescido antes de realizar a operação de return.

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

```
}

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

1.7 Exercício p19

O programa mostra vários tipos de dados, entre eles:

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

```
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);
}
```

1.8 Exercício p20

O programa imprime [46-11] pelo fato de ao usar o operador *shift*, os bits são deslocados para a direção das aspas francesas pelo número de posições especificadas pela expressão aditiva. As posições de bits que foram liberadas pela operação de deslocamento são preenchidas com zeros. Dessa forma, o número 23 que em binário é 10111 foi deslocado uma casa para a esquerda resultando no número 101110, que é 46. O número 23 foi deslocado para a direita, o que resultou no número 11, que é 1011 em binário.

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

2 Exercícios Resolvidos

2.1 Exercício p4

O código verifica se o caractere é o mesmo que as letras cujos números da tabela ascii correspondem aos no if, ou seja, uma vogal.

2.2 Exercício p14

O primeiro passo ao corrigir o código foi a indentação, após isso, foi simplificado o if else.