

Algoritmos e Estruturas de Dados II – Unidade 0 – Nivelamento

Pedro Henrique Amorim Sá – 742626

1)

```
public static boolean inArray(int x, int[] array)
{
    boolean contains = false;
    int n = array.length;
    int index = 0;

    while (!contains && index < n)
    {
        contains = (x == array[index]);
        index++;
    }

    return contains;
}
```

2)

```
public static boolean inAscArray(int x, int[] array)
{
    boolean contains = false;
    int n = array.length;
    int index = n / 2;
    n--;

    if (x == array[index])
    {
        contains = true;
    }
    else if (x < array[index])
    {
        while (!contains && index > 0)
        {
            index--;
            contains = (x == array[index]);
        }
    }
    else
    {
        while (!contains && index < n)
        {
            index++;
            contains = (x == array[index]);
        }
    }

    return contains;
}
```

3)

```
public static void getMaxMin(int[] array)
{
    int max = array[0];
    int min = array[0];
    int n = array.length;

    for (int i = 0; i < n; i++)
    {
        if (array[i] > max)
            max = array[i];
        if (array[i] < min)
            min = array[i];
    }

    System.out.println("max = " + max);
    System.out.println("min = " + min);
}
```

4)

```
public static void getMaxMin2(int[] array)
{
    int max = array[0];
    int min = array[0];
    int n = array.length;

    for (int i = 0; i < n; i++)
    {
        if (array[i] > max)
            max = array[i];
        else if (array[i] < min)
            min = array[i];
    }

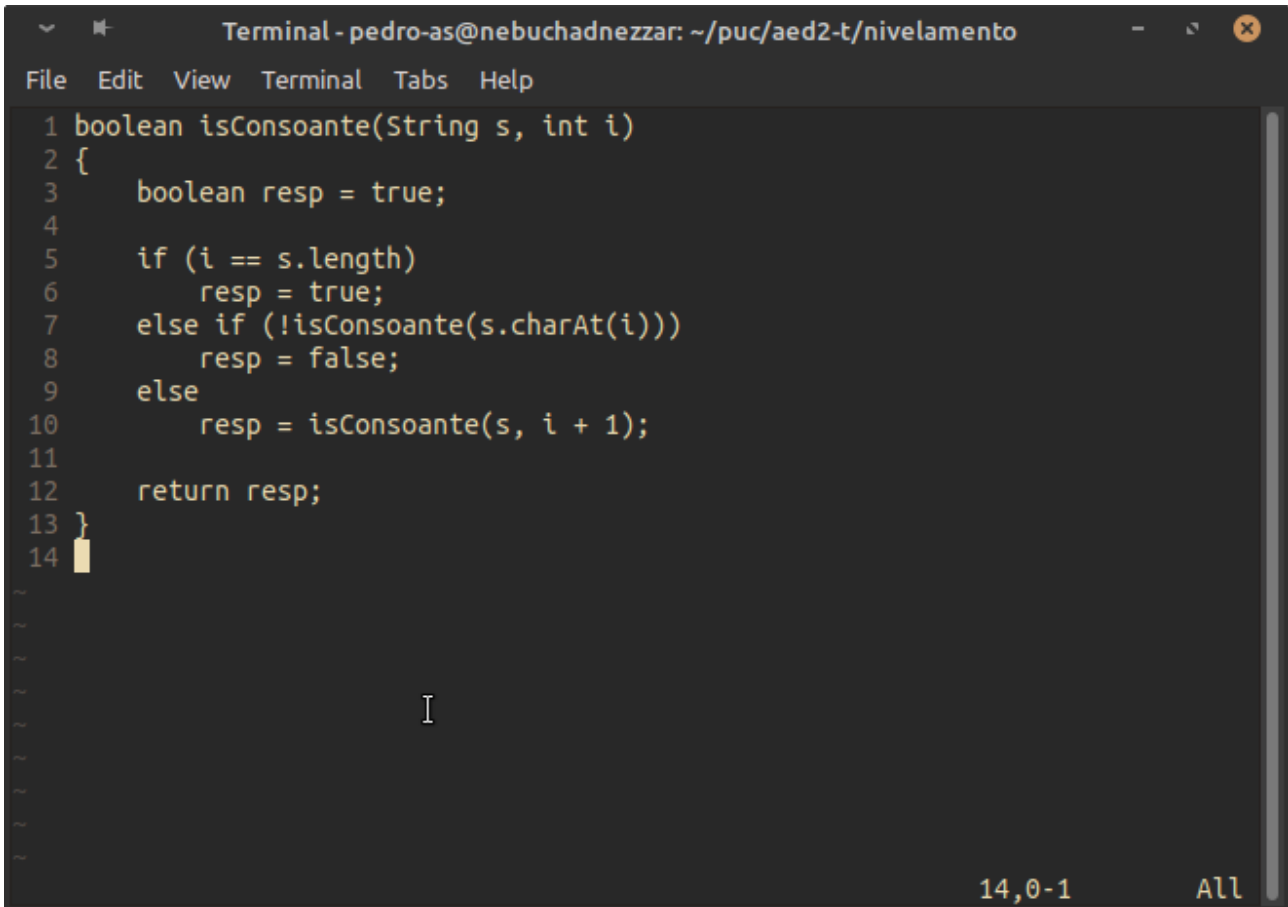
    System.out.println("max = " + max);
    System.out.println("min = " + min);
}
```

5) Verifica se o caractere recebido como parâmetro é uma vogal (*case insensitive*) e retorna *true* ou *false*.

6) O método `isVogal()` primeiramente chama o método `toUpper()`, que testa se o caractere recebido é minúsculo. Em caso positivo, converte-o para maiúsculo e o retorna; em caso negativo, retorna o próprio caractere. Com isso, o `isVogal()` verifica se o caractere é uma vogal de modo *case insensitive*.

7) `isLetra()` testa se o caractere é de fato uma letra e `isConsoante()` verifica se é uma consoante, caso o teste de `isLetra()` seja *true* e o de `isVogal()` seja *false*.

8) Correção do código:



```
Terminal - pedro-as@nebuchadnezzar: ~/puc/aed2-t/nivelamento
File Edit View Terminal Tabs Help
1 boolean isConsoante(String s, int i)
2 {
3     boolean resp = true;
4
5     if (i == s.length)
6         resp = true;
7     else if (!isConsoante(s.charAt(i)))
8         resp = false;
9     else
10         resp = isConsoante(s, i + 1);
11
12     return resp;
13 }
14
```

9) A primeira versão me parece mais fácil de entender por utilizar a estrutura *if-else if-else*, diminuindo a ramificação das condicionais.

10) O código poderia ser simplificado para tratar todas as situações que envolvem um retorno nulo ou em branco em uma condicional única.

11) O primeiro método retorna o próprio valor `i`, antes de decrementá-lo. O segundo decrementa o valor antes de retorná-lo, retornando efetivamente `i - 1`.

12) "1 1 1 1", "2 2 2 2", "3 3 3 3", "4 4 4 4", ...

13) $23_{(10)} = 0001\ 0111_{(2)}$

`<< 1` desloca 1 bit para a esquerda: $0010\ 1110_{(2)} = 46_{(10)}$

`>> 1` desloca 1 bit para a direita: $0000\ 1011_{(2)} = 11_{(10)}$