Procedimentos e funções - Exercícios

Algoritmos e Estruturas de Dados

Prof. Darlon Vasata

Tipos, tamanhos e literais

As diferentes linguagens de programação podem possuir em sua implementação vários tipos de dados primitivos, e os valores que podem ser armazenados em variáveis de cada tipo depende da quantidade de memória que cada tipo utiliza.

Tipos

Para inteiros:

Tipo de dado	Tamanho	valor mínimo	valor máximo
byte	1 byte	-128	127
short	2 bytes	-32768	32767
int	4 bytes	-2147483648	2147483647
long	8 bytes	-9223372036854775808	9223372036854775807 [1]

Para real:

Tipo de dado	Tamanho
float	4 bytes
double	8 bytes

para lógico:

[1] nove quintilhões, duzentos e vinte e três quatrilhões, trezentos e setenta e dois trilhões, trinta e seis bilhões ,oitocentos e cinquenta e quatro milhões, setecentos e setenta e cinco mil, oitocentos e sete.

Tipo de dado	Tamanho	valores
boolean	1 bit	`true` / `false`

para caractere:

Tipo de dado	Tamanho	descrição
char	2 bytes	armazena um único caractere

Fonte: Oracle. Java Language Specification. Types, Values and Variables: 4.2.1 Integral Types and Values.

Mínimos e máximos para inteiro

```
public static void mostrarValoresMinimosInteiro() {

    byte var_byte = Byte.MIN_VALUE;
    short var_short = Short.MIN_VALUE;
    int var_int = Integer.MIN_VALUE;
    long var_long = Long.MIN_VALUE;

    System.out.println("Valores mínimos para inteiros: ");
    System.out.println("    min byte  : " + var_byte);
    System.out.println("    min short : " + var_short);
    System.out.println("    min int   : " + var_int);
    System.out.println("    min long  : " + var_long);

}

public static void mostrarValoresMaximosInteiro() {

    byte var_byte = Byte.MAX_VALUE;
    short var_short = Short.MAX_VALUE;
    int var_int = Integer.MAX_VALUE;
    long var_long = Long.MAX_VALUE;

    System.out.println("Valores máximos para inteiros: ");
    System.out.println("    max byte  : " + var_byte);
    System.out.println("    max short : " + var_short);
```

```
    System.out.println("    max int   : " + var_int);
    System.out.println("    max long  : " + var_long);

}
```

Saída:

```
Valores mínimos para inteiros:
    min byte  : -128
    min short : -32768
    min int   : -2147483648
    min long  : -9223372036854775808
Valores máximos para inteiros:
    max byte  : 127
    max short : 32767
    max int   : 2147483647
    max long  : 9223372036854775807
```

Mínimos e máximos para real

```
public static void mostrarValoresMinimosReal() {
    float var_float = Float.MIN_VALUE;
    double var_double = Double.MIN_VALUE;

    System.out.println("Valores mínimos para real: ");
    System.out.println("    min float  : " + var_float);
    System.out.println("    min double : " + var_double);
}

public static void mostrarValoresMaximosReal() {
    float var_float = Float.MAX_VALUE;
    double var_double = Double.MAX_VALUE;

    System.out.println("Valores máximos para real: ");
    System.out.println("    max float  : " + var_float);
    System.out.println("    max double : " + var_double);
}
```

Saída:

```
Valores mínimos para real:
    min float  : 1.4E-45
    min double : 4.9E-324
Valores máximos para real:
    max float  : 3.4028235E38
    max double : 1.7976931348623157E308
```

Observe
O que o valor mínimo significa aqui?

Overflow

Dizemos que existe um *overflow* quando o valor a ser armazenado extrapola (para mais ou para menos) o valor possível de ser armazenado utilizando determinado tipo de dado.

Quando o valor extrapola o valor máximo dizemos que existe um **overflow positivo**, e quando extrapola o valor mínimo dizemos que ocorre um **overflow negativo** (STALLINGS 2010 p.270).

```java
public static void mostrarOverflowInteiro() {

    byte var_byte = Byte.MAX_VALUE;
    short var_short = Short.MAX_VALUE;
    int var_int = Integer.MAX_VALUE;
    long var_long = Long.MAX_VALUE;

    var_byte++;
    var_short++;
    var_int++;
    var_long++;

    System.out.println("Valores máximos +1 para inteiros: ");
    System.out.println("    max byte  : " + var_byte);
    System.out.println("    max short : " + var_short);
    System.out.println("    max int   : " + var_int);
    System.out.println("    max long  : " + var_long);

}
```

Saída:

```
Valores máximos +1 para inteiros:
    max byte  : -128
    max short : -32768
    max int   : -2147483648
    max long  : -9223372036854775808
```

> Atenção
>
> O *overflow* é um erro nos valores, e portanto deve-se evitar que ele ocorra. Por isso, deve-se tomar o cuidado ao escolher o tipo de dado a ser utilizado.

Underflow

Para os valores reais, quando um valor é pequeno o suficiente que não pode ser armazenado dizemos que houve um **underflow**.

Literais

Chamamos de literais aos valores informados diretamente no código, de maneira explícita.

inteiro

Decimal

```java
public static void mostrarLiteraisInteiroDecimal() {
   int var_int;
   long var_long;

   // Decimal
   var_int = 10100;
   var_long = 10100l; // l ou L para long

   System.out.println("Valores em decimal: ");
   System.out.println("    10100  : " + var_int);
   System.out.println("    10100  : " + var_long);

   }
```

Saída:

```
Valores em decimal:
    10100  : 10100
    10100  : 10100
```

Hexadecimal

```java
public static void mostrarLiteraisInteiroHexadecimal() {
    int var_int;
    long var_long;

    // Hexadecimal
    var_int = 0x10100;
    var_long = 0x10100l; // l ou L para long

    System.out.println("Valores em hexadecimal: ");
    System.out.println("    0x10100  : " + var_int);
    System.out.println("    0x10100  : " + var_long);

}
```

Saída:

```
Valores em hexadecimal:
    0x10100  : 65792
    0x10100  : 65792
```

Binário

```java
public static void mostrarLiteraisInteiroBinario() {
    int var_int;
    long var_long;

    // Binário
    var_int = 0b10100;
    var_long = 0b10100l; // l ou L para long

    System.out.println("Valores em binário: ");
    System.out.println("    0b10100  : " + var_int);
    System.out.println("    0b10100  : " + var_long);

    }
```

Saída:

```
Valores em binário:
    0b10100  : 20
    0b10100  : 20
```

real

Decimal

```java
public static void mostrarLiteraisRealDecimal() {
    float var_float;
    double var_double;

    // Decimais
    var_float = 10100f;  // f ou F para float
    var_double = 10100d; // d ou D para double

    System.out.println("Valores em decimal: ");
    System.out.println("    10100f  : " + var_float);
    System.out.println("    10100d  : " + var_double);

}
```

Saída:

```
Valores em decimal:
    10100f  : 10100.0
    10100d  : 10100.0
```

Hexadecimal

```java
public static void mostrarLiteraisRealHexadecimal() {
    float var_float;
    double var_double;

    // Hexadecimal
    // 0x <val hex> p <exp de 2>
    var_float = 0x10.0P2f; // 0x10.0 * 2^2 = 16 * 4 = 64
    var_double = 0x10.0P3d; // 0x10.0 * 2^3 = 16 * 8 = 128
```

```
    System.out.println("Valores em hexadecimal: ");
    System.out.println("    0x10.0P2f  : " + var_float);
    System.out.println("    0x10.0P3d  : " + var_double);

}
```

Saída:

```
Valores em hexadecimal:
    0x1p1f  : 64.0
    0x1p1f  : 128.0
```

Referências

Oracle. Java Language Specification. Types, Values and Variables: 4.2.1 Integral Types and Values.

STALLINGS, William. Organização e Arquitetura de Computadores. 8. ed. São Paulo: Pearson Practice Hall, 2010.