Cheat Sheet Java - Parte 1

Técnicas de Programação - 2025/02

Última atualização: 10/08/2025 11:02

Algoritmos, Variáveis e Atribuições

Toda variável e valor de retorno deverá ter seu tipo declarado de maneira explícita. Veja abaixo uma tabela com as equivalências de tipos.

Pseudo código	Java
int	long
float	double
character	char
string	String

Atenção

1. Divisão de **int** resultará sempre em um **int** (como a operação // em *Python*).
2. Operações que misturam **int** e **float** fazem conversão para **float**.

A parte inicial de todo algoritmo descreve seu nome, propósito, entradas e saídas. Veja abaixo um algoritmo que recebe dois inteiros e devolve sua soma. Note que declaramos os tipos das variáveis em **Input** e o tipo da saída em **Output** e que descrevemos em **Result** o que o algoritmo faz.

Algorithm 1: Exemplo1

Result: Soma as variáveis a e b
Input : int a, int b
Output: int

int $c \leftarrow a + b$
return c

```java
public static long Exemplo1(long a, long b) {
  long c = a + b;
  return c;
}
```

Condicionais e Loops

Indentação representa o conteúdo de um loop ou condicional. Para facilitar, estamos usando a linha vertical para indicar onde cada bloco acaba. Veja abaixo.

Algorithm 2: Exemplo2

Result: Calcula $\sum_i^n 2^i(-1)^i$
Input : int i, int n
Output: int

int $total \leftarrow 0$
int $temp \leftarrow 2^i$
while $i <= n$ **do**
  if $i\%2 = 0$ **then**
    $total \leftarrow total + temp$
  else
    $total \leftarrow total - temp$
  end
  $i \leftarrow i + 1$
  $temp \leftarrow temp \times 2$
end
return $total$

```java
public static long Exemplo2(long i, long n) {
  long total = 0;
  long temp = Math.pow(2, i);

  while (i <= n) {
    if (i % 2 == 0) {
      total += temp;
    } else {
      total -= temp;
    }
    i++;
    temp *= 2;
  }
  return total;
}
```

Loops `for` serão sempre **exclusivos** em nossos algoritmos. Veja o trecho abaixo.

Algorithm 3: Exemplo de for

```
for int i ← 5 to 10 do
  | ....
end
```

```java
// exemplo de for
for (long i = 5; i < 10; i++) {
  ...
}
```

Arrays

Um *Array* `A` é um tipo composto de um número **fixo** de elementos, todos do mesmo tipo. Usamos `A.length` para representar seu tamanho. O primeiro elemento tem índice `0` e o elemento `i` é acessado com `A[i]`. Veja um exemplo de uso abaixo.

Algorithm 4: Exemplo2

Result: Calcula média de um array
Input : array float A
Output: float

```
float total ← 0
for i = 0 to A.length do
  | total ← total + A[i]
end
return total / A.length
```

```java
public static double Exemplo2(double[] A) {
  double total = 0;
  for (int i = 0; i < A.length; i++) {
    total += A[i];
  }
  return total / A.length;
}
```

Strings

Java tem muitas funções de conveniência com Strings. Neste disciplina iremos tratá-las como um array do tipo especial `char`, que representa um único caractere. Veja abaixo como passar por todos os caracteres de uma `String` (em Java).

Algorithm 5: Acesso aos carateres de uma string

```
string S = ....
for int i ← 0 to S.length do
  | Faz algo com S[i]
end
```

```java
String S = ....;
for (long i = 0; i < S.length(); i++) {
  // faz algo com
  S.charAt(i);
}
```

Cuidado com `char` em Java

- `'a'` representa um único caractere (tipo `char`)
- `"a"` representa uma string de tamanho 1 (tipo `String`)

Essas duas coisas são diferentes em Java e costumam dar erros de compilação complicados.

Referências

Todo array é recebido *por referência* em um algoritmo. Ou seja, o comportamento padrão é que modificar elementos de um array recebido como parâmetro em um algoritmo modifica também o array "original" passado.

Strings são sempre imutáveis. Ou seja, não é possível fazer uma atribuição e modificar o caractere no índice i.

Um valor para "vazio"

O valor especial `NIL` representa a ideia de *vazio* ou *inválido*. Ele pode ser usado tanto para *string* quanto para *array*s. Em Java, ele é representado pela constante `null`.