

Estruturas de Dados e Algoritmos 2002.1

Curso de Ciência da Computação

Dalton Serey
DSC – UFPB

1 Convenções para escrita de algoritmos

Assinaturas Todo algoritmo é expresso em termos de operações. Cada operação tem uma *assinatura*. A assinatura atribui um nome à operação e especifica os parâmetros de entrada. Além dos tipos, a assinatura também determina os nomes pelos quais os parâmetros serão manipulados internamente, na descrição da operação. Em alguns casos, os tipos dos parâmetros podem ser indicados no texto que complementa a assinatura. O tipo de retorno, se houver, também pode ser indicado na assinatura, antecedendo o nome da operação. A notação que usamos para assinaturas é convencional e semelhante à usada em diversas linguagens de programação. Em alguns casos, é conveniente evidenciar que a operação é um método de um objeto. Nesse caso, o parâmetro referente ao objeto é colocado antecedendo o nome da operação (sem o tipo, que deve ficar evidente no texto complementar). Exemplos de assinaturas:

- `int MÍNIMO(int V[1..k])`

Esta assinatura declara a operação de nome *Mínimo* que recebe como entrada um vetor de inteiros denominado localmente de *V*, cujos elementos são indexados de 1 a *k*. O algoritmo retorna um valor inteiro. Por conveniência, consideramos que *k* também é recebido, o que nos permite utilizá-lo livremente no código da operação.

- `C.SORT()`

Esta assinatura declara uma operação denominada *Sort*. Observe que a notação usada é ao estilo OO. Logo, trata-se de um método do objeto *C*, cujo tipo é declarado em texto complementar. Observe que o algoritmo não tem valor de retorno.

- `boolean PRECEDE(string S1, string S2)`

Esta assinatura declara a operação denominada *Precede* que tem dois parâmetros de entrada do tipo *string*, denominados *S₁* e *S₂*. A operação retorna um valor booleano (ou um valor-verdade).

Passagem de parâmetros Toda passagem de parâmetros é *por valor*. Contudo, todos os objetos são sempre tratados *por referência*. Ou seja, se passamos um objeto a um procedimento, então sua *referência será copiada*. Logo, a operação pode alterar os atributos do objeto, mas não a referência original.

Variáveis e atribuições Todas as variáveis utilizadas em uma operação são locais. Não é necessário que sejam declaradas, mas devem ser inicializadas e seu tipo não pode mudar. Não há variáveis globais. Contudo, observe que é possível alterar os campos de um objeto se uma variável for uma referência para ele. Em tempo: a alteração e a leitura dos campos de um objeto pode

ser feita diretamente sem o uso de métodos acessores e modificadores a partir de qualquer operação. Isto simplifica dramaticamente o código. Denotamos atribuições pelo símbolo $\leftarrow$. Isto nos permite restringir o uso do símbolo de igualdade “=” para a operação de comparação.

Condicional A única estrutura condicional usada é o **if ... then ... else**. Quando múltiplos condicionais são usados (ao estilo **else if**), a indentação é mantida no mesmo nível.

Repetição As estruturas de repetição que usaremos são: **for**, **while**, e **do ... while**. A notação específica para cada uma delas é dada pelos exemplos a seguir:

for $i = 0$ to 100 do	while $i < 100$ do	do
$j \leftarrow j + i$	$i \leftarrow f(i)$	$i \leftarrow f(i)$
		while $i < 100$

Naturalmente, também usaremos a recursividade como forma de repetição.

Blocos Blocos de instruções são denotados exclusivamente por indentação. Não usamos chaves, parênteses ou marcadores para denotar o início e/ou o fim de um bloco. Os blocos do **if ... then ... else** também são demarcados por indentação. Novamente, esta convenção tem o objetivo de tornar o algoritmo muito mais legível sem introduzir possibilidades de ambigüidade.

Estrutura dos objetos A estrutura dos objetos consiste na declaração de todos os seus atributos e respectivos tipos. Em geral, usaremos uma notação abstrata semelhante à usada na maioria das linguagens de programação. Métodos construtores e destrutores não precisam ser descritos explicitamente.

Tipos de dados Podemos assumir como tipos elementares os tipos geralmente encontrados nas linguagens de programação convencionais: inteiros, booleanos, representações com ponto flutuante, caracteres e cadeias de caracteres. Também assumimos que dispomos dos mecanismos básicos para compor tipos: estruturas, classes, vetores e matrizes e enumerados. Tipos abstratos de dados podem ser usados, desde que sejam declarados e especificados claramente.

Retornos e erros Operações devem retornar um único valor do tipo indicado na assinatura da operação (veja a descrição acima). Se nenhum tipo de retorno foi indicado, então nenhum valor deve ser retornado. Erros podem ser declarados em situações em que a operação não tem como proceder adequadamente.