

ALGORITMOS COM QUALIDADE

1. Algoritmos devem ser feitos para serem lidos por seres humanos. Tenha em mente que seus algoritmos deverão ser lidos e entendidos por outras pessoas (e por você mesmo) de tal forma que possam ser corrigidos, receber manutenção e ser modificados.
2. Escreva os comentários no momento em que estiver escrevendo o algoritmo. Como o objetivo de se escrever comentários é facilitar o entendimento do algoritmo, eles devem ser tão bem concebidos quanto o próprio algoritmo. E a melhor maneira de se conseguir isso é escrevê-lo nos momentos de maior intimidade com os detalhes, ou seja, durante a resolução do problema.
3. Os comentários deverão acrescentar alguma coisa; não apenas parafrasear os comandos. O conjunto de comandos nos diz o que está sendo feito; os comentários deverão nos dizer por quê.
4. Use comentários no prólogo. Todo algoritmo ou procedimento deverá ter comentários no seu prólogo para explicar o que ele faz e fornecer instruções para seu uso. Alguns destes comentários seriam:
 - a) Uma descrição do que faz o algoritmo.
 - b) Como utilizá-lo.
 - c) Explicação do significado das variáveis mais importantes.
 - d) Estruturas de dados utilizadas.
 - e) Os nomes de quaisquer métodos especiais utilizados, juntamente com referências nas quais mais informações possam ser encontradas.
 - f) Autor.
 - g) Data de escrita.
5. Utilize espaços em branco para melhorar a legibilidade. Espaços em branco, inclusive linhas em branco, são valiosos para melhorar a aparência de um algoritmo.

Exemplo:

- a) Deixar uma linha em branco entre as declarações e o corpo do algoritmo.
 - b) Deixar uma linha em branco antes e outra depois de um comentário.
 - c) Separar grupos de comandos que executam funções lógicas distintas por uma ou mais linhas em branco.
 - d) Utilizar brancos para indicar precedência de operadores. Ao invés de " $A + B * C$ " é bem mais agradável a forma " $A + B * C$ ".
6. Escolha nomes representativos para suas variáveis. Os nomes das variáveis deverão identificar, o melhor possível, as quantidades que elas representam. Por exemplo, $X = Y + Z$ é muito menos claro que $PRECO = CUSTO + LUCRO$. Uma seleção adequada de nomes de variáveis é o princípio mais importante da legibilidade de algoritmos.
 7. Um comando por linha é suficiente. A utilização de vários comandos por linha é prejudicial por várias razões, dentre as quais destacam-se:
 - a) O algoritmo fica mais ilegível.
 - b) O algoritmo fica mais difícil de ser depurado.

Exemplo:

```
A ← 14.2; I ← 1; enquanto I < 10 faça X ← X + 1; K ← I * K; I ← I + 1; fimenquanto;
```

O mesmo exemplo com cada comando em uma linha:

```
A ← 14.2;
I ← 1;
enquanto I < 10 faça
    X ← X + 1;
    K ← I * K;
    I ← I + 1;
fimenquanto;
```

8. Utilize parênteses para aumentar a legibilidade e prevenir-se contra erros. Exemplo:

Com poucos parênteses

```
A * B * C / ( D * E * F )
A * B / C * D / E * F
A ^ B ^ C
A / B / C / D
X > Y ou Q
A + B < C
```

Com parênteses extra

```
( A * B * C ) / ( D * E * F )
( ( ( ( A * B ) / ( C * D ) / E ) * F
( A ^ B ) ^ C
( ( A / B ) / C ) / D
( X > Y ) ou Q
( A + B ) < C
```

9. Utilize "identação" (tabulação) para mostrar a estrutura lógica do algoritmo. A identação não deve ser feita de forma caótica, mas segundo certos padrões estabelecidos.

10. Lembre-se: toda vez que for feita uma modificação no algoritmo, os comentários associados devem ser alterados, e não apenas os comandos. Antes não comentar do que deixar um comentário errado.

METODOLOGIA DE DESENVOLVIMENTO DE ALGORITMOS

Uma das dificuldades naturais de um iniciante em programação é como começar a desenvolver um algoritmo para resolver um dado problema. Os passos seguintes, se seguidos, podem auxiliar nesta tarefa:

passo 1: leia cuidadosamente a especificação do problema até o final. (fazer anotações)

```
ENTENDEU ← falso; VEZES ← 0;
```

passo 2:

```
enquanto não ENTENDEU faça
  se VEZES ≤ 4 então
    "leia a especificação até o final";
    VEZES ← VEZES + 1;
  senão
    "pergunte ao professor até entender";
    ENTENDEU ← verdadeiro;
  fimse;
fimenquanto;
```

passo 3: Levantar e analisar todas as saídas exigidas na especificação do problema; (impressões)

passo 4: Levantar e analisar todas as entradas citadas na especificação do problema; (leituras)

passo 5: Verificar se é necessário gerar valores internamente ao algoritmo e levantar as variáveis necessárias e os valores iniciais de cada uma; (comentar)

passo 6: Levantar e analisar todas as transformações necessárias para, dadas as entradas e valores gerados internamente, produzir as saídas especificadas; (comentar)

passo 7: Testar cada passo do algoritmo, verificando se as transformações intermediárias executadas estão conduzindo aos objetivos desejados. Utilizar, sempre que possível, valores de teste que permitam prever os resultados a priori; (comentar)

passo 8: Fazer uma reavaliação geral, elaborando o algoritmo através da integração das partes. (rever comentários)

REFERÊNCIA:

GUIMARÃES, Ângelo de Moura. **Algoritmos e estruturas de dados**. 1.ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 1985.