Conceitos Básicos de Programação - Algoritmos

Prof. Francisco Rapchan
www.geocities.com/chicorapchan
rapchan@terra.com.br

O objetivo desta pequena apostila é servir de material de apoio para os cursos introdutórios de programação. Deve ficar claro que não há aqui nenhuma pretensão em substituir os livros texto da disciplina. Ao contrário, os livros devem ser adquiridos e estudados com afinco pois trazem informações valiosas e de uma forma muito mais abrangente e completa do que esta pequena apostila.

Este material é baseado nos seguintes textos:

Notas de Aula da Disciplina Linguagem de Programação do professor Edilson Luís do Nascimento;
Notas de Aula da Disciplina Linguagem de Programação da professora Monalessa Perini Barcellos;
Manuais de Referência da ferramenta VisuAlg (www.apoioinformatica.inf.br/visualg);
Livro Fundamentos da Programação de Computadores, A.F.G. Ascencio e E.A.V. de Campos.

Introdução

O algoritmo é certamente um dos conceitos mais importantes na programação de computadores. Podemos pensar no algoritmo como a descrição, de forma lógica, dos passos a serem executados para executar uma tarefa. É a forma pela qual descrevemos soluções de problemas do nosso mundo, a fim de serem implementadas utilizando os recursos do mundo computacional. Dito de outra forma, um algoritmo é uma seqüência lógica de ações a serem executadas para se realizar uma determinada tarefa.

Algoritmo não é a solução de um problema, pois, se assim fosse, cada problema teria um único algoritmo. Algoritmo é um caminho para a solução de um problema, e em geral, são muitos os caminhos que levam a uma solução.

O algoritmo não é uma etapa na formação de um programa. Não é um rascunho do programa, nem mesmo um programa em uma linguagem informal. O algoritmo é o conjunto de instruções que manipula os dados para obtenção de um resultado.

Há uma relação direta entre algoritmos e programas: os algoritmos são a estrutura lógica dos programas. Um dos grandes teóricos da computação e criador da linguagem de programação Pascal, Niklaus Wirth define que:

Programas = Algoritmos + Estrutura de dados

Por exemplo, suponha o seguinte algoritmo para fazer um bolo (atenção não execute este algoritmo sozinho, peça ajuda de alguém que realmente saiba cozinhar!)

```
1. Pegue os ingredientes: ovos, trigo, sal, leite, açúcar e fermento
2. Bata em uma tigela a manteiga com o açúcar até virar uma pasta branca
3. Acrescente as gemas, o leite, o trigo e o sal.
4. Bata até formar um creme homogêneo.
5. Acrescente o fermento e misture.
6. Coloque o creme na assadeira
7. Coloque para assar por 20 minutos
```

Neste caso temos os dados como os ingrediente (ovos, trigo, sal, leite, açúcar e fermento) e o algoritmo é a própria receita que manipula estes ingredientes.

Um mesmo algoritmo pode ser escrito de formas diferentes. Esta mesma receita escrita de outra forma poderia ser:

```
1. Pegue os ingredientes: ovos, trigo, sal, leite, açúcar e fermento
2. Coloque a manteiga e o açúcar em uma tigela
3. Enquanto não formar um creme branco:
   Bata a manteiga e o açúcar da tigela
4. Acrescente as gemas, o leite, o trigo e o sal.
5. Enquanto não formar um creme homogênio:
   Bata as gemas, o leite, o trigo e o sal.
6. Acrescente o fermento e misture.
7. Coloque o creme na assadeira
8. Coloque no forno
9. Aguarde :
   Assar até passar 20 minutos
```

Pode-se aprender a construir programas fazendo os algoritmos usando uma linguagem de programação como C ou Pascal. Entretanto estas linguagens apresentam uma série de dificuldades extras para quem está começando a programar tais como:

- Formalidade. As linguagens de programação são extremamente formais. O estudante pode se perder na sintaxe da linguagem perdendo o algoritmo de vista.
- Idioma. As linguagens de programação, na sua maioria, usam termos e expressões em Inglês. Essa é outra fonte de confusão para quem inicia a programar e não conhece bem o idioma inglês.

Uma outra forma de aprender a fazer algoritmos é utilizar uma linguagem hipotética e mais tarde, assim que os principais conceitos tenham sido entendidos, mostrar a "tradução" desta linguagem em uma linguagem real.

Neste curso utilizarmos inicialmente uma linguagem parecida com o Pascal, porém em português e muito menos formal. Chamamos uma linguagem de programação não formal de uma "**pseudo-linguagem de programação**". Há várias pseudo-linguagens, algumas muito conhecidas como o **PORTUGOL** e o **Fluxograma**. A primeira é uma mistura de português com Pascal e a segunda é uma forma gráfica de representação de algoritmos.

Exemplo 1
Faça um algoritmo em pseudo linguagem que leia dois números que o usuário do computador digitará no teclado, some-os e mostre o resultado na tela do computador.

Uma solução seria:

```
Leia do teclado dois valores
Some os dois valores
Mostre o resultado da soma na tela
```

Uma solução um pouco mais formal seria:

```
Leia o valor do teclado e armazene na memória A
Leia o valor do teclado e armazene na memória B
Some os valores da memória A e B e coloque o resultado na memória SOMA
Mostre na tela o valor da memória SOMA
```

Nesta solução vemos o uso de 3 memórias: A, B e SOMA. Aqui o conceito de memória é semelhante ao das memórias existentes nas calculadoras. Elas servem para acumular o resultado de cálculos intermediários.

Uma solução ainda mais formal seria:

```
INÍCIO
   Leia (A)
   Leia (B)
   SOMA ← A + B
   Mostre(SOMA)
FIM ALGORITMO
```

Nesta solução, as palavras INÍCIO e FIMALGORITMO foram usadas para delimitar o algoritmo.

Observe ainda que nessa última solução, usamos uma notação bem mais resumida. Embora aqui cada linha corresponda à linha do exemplo anterior, vemos que utilizamos bem menos palavras.

- Na primeira linha por exemplo, ao invés de dizer: "Leia o valor do teclado e armazene na memória A", dissemos apenas Leia (A), significando a mesma coisa.
- Na terceira linha, ao invés de dizer: "Some a memória A com a memória B e coloque o resultado na memória SOMA", dissemos apenas SOMA ← A + B, significando a mesma coisa.

De fato, mesmo em pseudo - linguagens de programação, é necessário algum formalismo, caso contrário teremos muita dificuldade para adaptar (traduzir) os algoritmos construídos na pseudo-linguagem em algoritmos de uma linguagem real de programação como o Pascal.

Outro conceito inicial importante em algoritmos é o de **variáveis**. Hoje em dia qualquer calculadora tem pelo menos uma **memória para armazenar valores intermediários** dos cálculos. As calculadoras um pouco mais sofisticadas têm dezenas destas memórias. Nestas calculadoras, para diferenciar uma memória da outra, normalmente são usadas letras do alfabeto (A, B, C, D, etc.) ou símbolos como M1, M2, M3, etc. representando a memória 1, memória 2 e assim por diante.

No computador a idéia de memória é um pouco mais sofisticada do que em uma calculadora. Quando vamos usar memória em um algoritmo para armazenar algo como um resultado de uma operação ou mesmo um número ou uma palavra, precisamos informar primeiro ao computador que precisaremos desta memória.

Informalmente é como se disséssemos ao computador: "*Reserve memória para que eu possa usar em um cálculo com números inteiros*" ou "*Reserve memória para que eu possa usar em um cálculo com números reais*" e assim por diante.

Alem de reservar a memória, temos que dizer ao computador como iremos referenciar aquela memória. Então, diríamos de alguma forma para o computador: "*Reserve memória para que eu possa usar em um cálculo com números inteiros e chame esta memória de A*" ou então: "*Reserve memória para que eu possa usar em um cálculo com números inteiros e chame esta memória de SOMA*" e assim por diante. Chamamos os endereços de memória respectivamente de A e de SOMA. Poderíamos ter chamado de B, de VALOR, de CONTADOR, de SALÁRIO, de DÍVIDA ou do que quiséssemos (dentro de certas regras, que veremos depois).

Dizemos que estes endereços nomeados de memória são as **variáveis do programa**. Então poderíamos ter dito: "*Crie a variável A do tipo inteiro*" que seria o equivalente a dizer: "*Reserve memória para que eu possa usar em um cálculo com números inteiros e chame esta memória de A*". Uma forma ainda mais simples seria dizer simplesmente "Variável A : inteiro". Se tivermos que usar as variáveis A, B e SOMA faríamos:

```
     Declare as Variáveis a,b,c,j: numérico
ou
     Variáveis
       A, B, C: inteiro
```

Uma variável representa uma localização de memória do computador utilizada para armazenar valores. Uma variável simples pode assumir diversos valores ao longo do tempo, mas em um dado instante ela representa exatamente um. Sempre encontramos na posição de memória representada por uma variável o último valor lá depositado. Eventuais valores anteriores não são mais recuperáveis. Uma vez atribuído um valor a uma variável, o valor anteriormente armazenado se perde.

Exemplo 2

Faça um algoritmo em pseudo-linguagem que leia dois números que o usuário do computador digitará no teclado, some-os e mostre o resultado na tela do computador.

Uma solução poderia ser:

```
Declare as Variáveis
    A, B, C : do tipo numérico
Início
    Leia (A)
    Leia (B)
    SOMA ← A + B
    Mostre(SOMA)
Fim
```

Como estamos representando o algoritmo em uma pseudo-linguagem, não temos um compromisso muito forte com a sintaxe desta representação, ou seja, podemos dizer a mesma coisa de forma um pouco diferente:

```
algoritmo "Soma de dois números"

var
   A, B, Soma: inteiro

inicio
   Leia (A)
   Leia (B)
   Soma ← A + B
   Escreva (Soma)
fimalgoritmo
```

Observe neste exemplo que apresentamos a mesma solução com duas notações ligeiramente diferentes. Na segunda notação definimos um "nome" para o algoritmo: "Soma de dois números" e usamos a expressão *declare* para indicar as variáveis que vamos precisar. Nesta notação também escolhemos o tipo *inteiro* para as variáveis.

Outra diferença é que dissemos Escreva (soma) ao invés de Mostre (soma).

Com isso percebe-se que um mesmo algoritmo pode ser representado com notações ligeiramente diferentes.

Exemplos de Algoritmos

A seguir são apresentados vários exemplos que buscam mostrar, passo a passo, a forma de construir algoritmos usando pseudocódigo. Para isso iremos procurar adotar a mesma notação usada pelo software de domínio público VisuAlg 1.5 desenvolvido por Cláudio Morado de Souza. O VisuAlg é um editor e interpretador de algoritmos.

Exemplo 3. Faça um algoritmo que leia dois números e mostre sua soma.

```
ALGORITMO "Soma dois números"
VAR
   numero1, numero2, resultado: Numerico

INICIO
   Leia (numero1)
   Leia (numero2)
   soma ← numero1 + numero2
   Escreva (soma)
FIMALGORITMO
```

Neste algoritmo, estamos criando **3 variáveis**: *numero1*, *numero2* e *resultado*. Todas elas são do tipo **numérico**, ou seja, só podem receber números. O comando ***Leia*** faz com que o algoritmo pare e fique esperando que alguém digite um valor. Este valor é atribuído (depositado) na variável *numero1*. Depois é feita a mesma coisa com a variável *numero2*. Assim temos os números que o usuário do algoritmo quer somar. A soma é feita e depositada em uma variável chamada de *soma* (nome sugestivo, não?). Embora na matemática tenhamos nos acostumado a usar *x*, *y*, *z* e *w* como nomes de variáveis, em programação procuramos usar **nomes mais significativos**. A idéia é sempre usarmos variáveis com nomes que expressem o que elas contêm ou o que elas fazem.

Exemplo 4. Faça um algoritmo que leia os lados de um retângulo e mostre sua área.

```
algoritmo "Área do retângulo"
var
   lado1, lado2, area: Numerico
inicio
   Leia (lado1)
   Leia (lado2)
   area ← lado1 * lado2
   Escreva (area)
fimalgoritmo
```

A área de um retângulo é o produto dos seus lados. Observe que a multiplicação é representada pelo asterisco (*).

Exemplo 5. Faça um algoritmo que leia os lados de um triângulo e mostre sua área. A fórmula da área do triângulo calculado pelos valores dos lados a, b e c é: raiz quadrada de p(p-a)(p-b)(p-c) onde p = (a+b+c)/2.

```
ALGORITMO "Área do triangulo"
   var a,b,c,p,area: Numerico
INICIO
   Leia (a,b,c)
   p ← (a+b+c)/2
   area ← (p*(p-a)*(p-b)*(p-c))^(1/2)
   Escreva (area)
FIMALGORITMO
```

Usamos o símbolo ^ para representar a **potenciação**. Observe que usamos **^ (1/2)** para representar a **raiz quadrada**. Isso vem da relação em que a raiz n de um número é igual a este número elevado a 1/n.

Exercícios

1. Faça um Programa que mostre a mensagem "Olá Mundo" na tela.
2. Faça um Programa que leia a temperatura em graus Farenheit e mostre a temperatura em graus Celsius. Depois, altere o programa para que leia a temperatura em graus Celsius e mostre em Farenheit.

   F = (9/5)*(Celsius) + 32. C = (5 * (Farenheit −32) / 9).

3. Faça um programa que receba um número positivo e mostre: o número digitado ao quadrado, o número digitado ao cubo, a raiz quadrada do número digitado, a raiz cúbica do número digitado.
4. Faça um algoritmo que leia o valor do raio (r) de um círculo e calcule o valor da sua circunferência e área (Circunferência $= 2 \pi r$, Área $= \pi r^2$ onde $\pi \approx 3{,}14$).
5. Faça um algoritmo que leia o valor de um salário e mostre este salário com um aumento de 20%.
6. Para tornar o algoritmo anterior mais genérico, altere-o para que permita ao usuário digitar o salário e a porcentagem de aumento e então mostre o valor do novo salário.
7. Faça um algoritmo que leia três notas e mostre a soma e a média aritmética delas.
8. Em uma certa escola, a média sé calculada de forma ponderada: a primeira nota tem peso 2, a segunda tem peso 3 e a terceira tem peso 4 (a idéia é que as notas do final do período sejam mais importantes e motive o aluno a estudar mais...).
9. Altere o algoritmo anterior para que ele mostre a média aritmética e a média ponderada das notas.
10. Para tornar mais genérico o algoritmo da média ponderada desenvolvido nos exercícios anteriores, permita que o usuário entre com as notas e os pesos de cada nota e então calcule a média ponderada usando esses pesos.