Capítulo 2

Algoritmos

2.1 Conceito de Algortimo

Um algoritmo pode ser definido como **uma sequência finita de passos (instruções) para resolver um determinado problema**. Sempre que desenvolvemos um algoritmo estamos estabelecendo um padrão de comportamento que deverá ser seguido (uma norma de execução de ações) para alcançar o resultado de um problema.

Para o desenvolvimento de um algoritmo eficiente é necessário obedecermos algumas premissas básicas no momento de sua construção:

- ▷ Definir ações simples e sem ambiguidade;
- ▷ Organizar as ações de forma ordenada
- ▷ Estabelecer as ações dentro de uma sequência finita de passos.

O algoritmo 1 é um exemplo simples de algoritmo (sem condições ou repetições) para troca de um pneu.

Os algoritmos são capazes de realizar tarefas como:

1. Ler e escrever dados;
2. Avaliar expressões algébricas, relacionais e lógicas;
3. Tomar decisões com base nos resultados das expressões avaliadas;
4. Repetir um conjunto de ações de acordo com uma condição;

Algoritmo 1 Troca de pneu do carro.

```
1: desligar o carro
2: pegar as ferramentas (chave e macaco)
3: pegar o estepe
4: suspender o carro com o macaco
5: desenroscar os 4 parafusos do pneu furado
6: colocar o estepe
7: enroscar os 4 parafusos
8: baixar o carro com o macaco
9: guardar as ferramentas
```

No algoritmo 2 estão ilustradas as tarefas anteriormente mencionadas. Nas linhas de 2 a 4 pode-se observar a repetição de uma ação enquanto uma dada condição seja verdadeira, neste caso em específico, o algoritmo está repetindo a ação 'esperar ônibus' enquanto a condição 'ônibus não chega' permanecer verdadeira, assim que essa condição se tornar falsa (quando o ônibus chegar) o algoritmo deixará de repetir a ação 'esperar ônibus', e irá executar a linha 5.

Já nas linhas de 7 a 9, é possível observar um exemplo da execução (ou não execução) de uma uma ação com base na avaliação de uma expressão. Nesse trecho, o algoritmo avalia se a expressão 'não tenho passagem' é verdadeira e em caso positivo, executa a acão 'pegar dinheiro'. Caso a expressão 'não tenho passagem' seja falsa (ou seja, a pessoa tem passagem) então o algoritmo irá ignorar a ação 'pegar dinheiro' e irá executar a linha 10.

Estas estruturas de controle serão estudadas em detalhe nos capítulos 6 e 7.

2.2 Partes de Um Algoritmo

Um algortimo quando programado num computador é constituído pelo menos das 3 partes, sendo elas:

1. Entrada de dados;
2. Processamento de dados;
3. Saída de dados;

Na parte de entrada, são fornecidas as informações necessárias para que o algoritmo possa ser executado. Estas informações podem ser fornecidas no momento em que o programa está sendo executado ou podem estar embutidas dentro do mesmo.

Algoritmo 2 Pegar um onibus.

```
1: ir até a parada
2: enquanto ônibus não chega faça
3:         esperar ônibus
4: fim-enquanto
5: subir no ônibus
6: pegar passagem
7: se não há passagem então
8:         pegar dinheiro
9: fim-se
10: pagar o cobrador
11: troco ← dinheiro - passagem
12: enquanto banco não está vazio faça
13:         ir para o próximo
14: fim-enquanto
15: sentar
16: ...
```

Figura 2.1: Partes básicas de um algoritmo.

Na parte do processamento são avaliadas todas as expressões algébricas, relacionais e lógicas, assim como todas as estruturas de controle existentes no algoritmo (condição e/ou repetição).

Na parte de saída, todos os resultados do processamento (ou parte deles) são enviados para um ou mais dispositivos de saída, como: monitor, impressora, ou até mesmo a própria memória do computador.

Por exemplo, considere o algoritmo 3 que tem como objetivo calcular a área de uma circunferência dada por $A = \pi R^2$. Para calcular a área é necessário saber os valores do raio R e do π. Considerando que o valor de π é constante o mesmo poderá ser gravado (definido) dentro do próprio algoritmo, e a entrada para o processamento desse algoritmo consistirá nesse valor juntamente com o valor do raio R (que deve ser informado pelo usuário pelo teclado, por exemplo). O processamento do algoritmo será a realização do cálculo πR^2 e a atribuição do resultado dessa expressão para a variável A. A parte da saída consistirá na escrita do valor de A no monitor.

Algoritmo 3 Calcula Área de uma Circunferência.

1: $\pi \leftarrow 3.14$ {entrada para o processamento}
2: **leia** R {entrada para o processamento}
3: $A \leftarrow \pi * R^2$ {processametno}
4: **escreva** A {saída}

2.3 Representações de um Algoritmo

2.3.1 Fluxograma

Os fluxogramas são uma apresentação do algoritmo em formato gráfico. Cada ação ou situação é representada por uma caixa. Tomadas de decisões são indicadas por caixas especiais, possibilitando ao fluxo de ações tomar caminhos distintos.

A Figura 2.2 representa um algoritmo na forma de um fluxograma. O início e o fim do algoritmo são marcados com uma figura elíptica; as ações a serem executadas estão em retângulos; sendo que as estruturas de controle condicionais estão em losangos e indicam duas possibilidades de proseguimento do algoritmo, uma para o caso da expressão avaliada (condição) ser verdadeira e outra para o caso de ser falsa.

No exemplo da Figura 2.2, a primeira ação é executada ('abrir forno') e então a segunda expressão é avaliada ('fogo aceso?') como verdadeira ou falsa; caso seja verdadeira, o algoritmo prosegue para a ação à esquerda ('botar lenha'); caso seja falsa, o algoritmo executa a ação à direita ('acender fogo'). Em seguida, para qualquer um dos casos, a próxima ação a ser executada é ('assar pão').

2.4 Programas de Computador

2.5 Linguagens

Qualquer tipo de informação que deva ser transferida, processada ou armazenada deve estar na forma de uma linguagem. A linguagem é imprescindível para o processo de comunicação. Duas pessoas que se falam o fazem através de uma linguagem em comum, a linguagem natural. Da mesma forma, duas máquinas trocam informação por uma linguagem, que neste caso mais técnico e restrito, se chama protocolo. Do mesmo modo, um computador armazena suas instruções em código de máquina. Estas diferentes linguagens não podem ser traduzidas diretamente entre sí, pois além de serem representadas de modos diferentes, também referem-se a coisas muito distin-

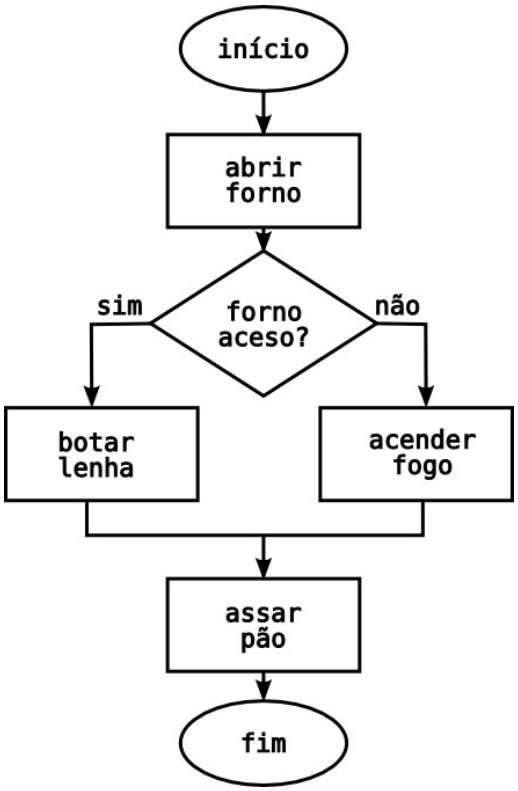

Figura 2.2: Algoritmo representado em forma de um fluxograma.

tas. Para que um ser humano possa programar, armazenar e buscar informações num computador, é necessário que saiba instruí-lo na sua linguagem de máquina ou numa linguagem intermediária (uma linguagem de programação) que possa ser facilmente traduzida para o computador.

2.5.1 Linguagem Natural

A linguagem natural é a maneira como expressamos nosso raciocínio e trocamos informação. Como é a expressão da cultura de uma sociedade, desenvolvida através das gerações e em diferentes situações, raramente constitui um sistema de regras rígidas que possa ser implementada numa máquina ou que possa ser transcrita logicamente. Além da linguagem falada, fazem parte da nossa comunicação gestos e posturas, que não podem ser diretamente adaptados para compreensão de uma máquina. Por fim, toda a comunicação eficiente pressupõe um conhecimento prévio comum entre os interlocutores, por exemplo a mesma língua, a mesma bagagem cultural e assim por diante.

Ao contrário dos seres humanos, as máquinas (dentre elas os computadores) são

projetados para executar tarefas bem determinadas a partir de determinadas instruções. Um computador não é por si só uma máquina inteligente no sentido que não pode aprender com a própria experiência para melhorar seu comportamente futuro[1]. Ao contrário, um computador é somente capaz de realizar estritamente as tarefas que lhe forem delegadas e que façam parte do conjunto daquelas ações que ele pode executar. Neste sentido, é necessário compreender que tipo de instruções podem ser executadas pelos computadores para que possamos programá-los — instruí-los com a sequência de ações necessárias para resolver um determinado problema — de modo que realizem a tarefa do modo desejado.

2.5.2 Linguagem de Máquina e Assembler

Além do fato de o computador necessitar que lhe instruam com ações bem específicas, estas ações devem ser passadas para o computador numa linguagem que ele possa entendê-las, chamada linguagem de máquina. Esta linguagem é composta somente por números, representados de forma binária, que, sob o ponto de vista do computador, representam as operações e os operandos que serão usados no processamento do programa. Para um ser humano, a linguagem de máquina é dificílima de se compreender. Assim, existe uma linguagem representada por comandos mas que reproduz as tarefas que serão executadas dentro do computador, a linguagem de montagem (*assembly*). Entretando, mesmo a linguagem de montagem é difícil de programar e os programas feitos para um determinado processador, por conterem instruções específicas deste, não funcionarão em um processador de outro tipo.

Com ilustração, abaixo é mostrado o início de um programa que escreve a frase "Olá Mundo" no monitor. Na coluna da esquerda está o endereço relativo de memória, na coluna do centro o programa escrito em linguagem de máquina e na coluna da direita a representação em caracteres ASCII. Teoricamente, o programa poderia ser escrito diretamente em linguagem de máquina, como mostrado abaixo, entretando a sintaxe do mesmo é muito pouco compreensível e a probabilidade de erro para o seu desenvolvimento seria muito grande.

```
00000000   7F 45 4C 46  01 01 01 00  00 00 00 00  00 00 00 00  .ELF............
00000010   02 00 03 00  01 00 00 00  D0 82 04 08  34 00 00 00  ............4...
00000020   BC 0C 00 00  00 00 00 00  34 00 20 00  07 00 28 00  ........4. ...(.
```

[1]Diversos esforços vêm sendo despendidos dentro do meio científico para equipar computadores com esta capacidade, o campo de pesquisa que cuida desse tipo de tarefa é conhecido como Inteligência Artifical

Figura 2.3: Compilação: o programa em linguagem de programação é transformado em instruções em linguagem de máquina (que o processador pode executar).

```
00000030   24 00 21 00  06 00 00 00  34 00 00 00  34 80 04 08  ..!.....4...4...
00000040   34 80 04 08  E0 00 00 00  E0 00 00 00  05 00 00 00  4...............
```

2.5.3 Linguagens de Programação

Para facilitar a tarefa de programar um computador, foram criadas várias linguagens de programação. Estas linguagens são um maneira de tentar escrever as tarefas que o computador vai realizar de maneira mais parecida com a linguagem natural. Embora ainda seja muitas vezes complexo em comparação com a linguagem natural, um programa escrito em uma linguagem de programação é muito mais fácil de ser implementado, compreendido e modificado.

As linguagens de programação são um meio termo entre a linguagem de máquina e a linguagem natural. Deste modo são classificadas de acordo com o nível entre a linguagem natural ou de máquina que ocupam. As linguagens muito parecidas com linguagem de máquina são chamadas de linguagens **de baixo nível** e suas instruções parecem-se muito com aquelas que serão executadas pelo processador. As linguagens de **alto-nível** são as que guardam mais semelhanças com a linguagem natural. Exemplo de linguagens de baixo nível é a linguagem de montagem (*assembly*). Exemplos de linguagens de alto-nível são: Pascal, C, Fortran, Java, Perl, Python, Lisp, PHP, entre outras.

Como o processador não pode executar o código numa linguagem de programação, esta deve ser traduzida em código de máquina antes de ser executada. Este processo é chamado de textbfcompilação (representado na Figura 2.3) e é responsável por converter os comandos da lingugem de programação nas instruções em código de máquina que o processador poderá utilizar.

Por exemplo, o código de máquina da seção 2.5.2 foi gerado pelo programa a seguir, escrito na linguagem de programação C. Esse programa, depois de compilado, escreve frase "Olá Mundo" no monitor. A compilação, isto é, a tradução do programa em C para linguagem de máquina, produz algo parecido com o que foi é mostrado na seção 2.5.2, para o caso de um processador da família 80386, usados em PCs.

```
#include <stdio.h>
```

```
int main(){
   printf("Olá Mundo\n");
}
```

A primeira linha (`#include`) inclui algumas bibliotecas de instruções que facilitarão a programação. A linha seguinte indica que esta é a parte principal (`main`) do programa; o que estiver dentro do bloco delimitado por chaves { } será executado. Finalmente, a próxima linha imprime (`printf`) o argumento (“`Olá Mundo`”) no monitor.

Um programa escrito em linguagem de máquina, como contém instruções específicas de um processador, só poderá ser utilizado naquele processador ou em similares. Em contrapartida, uma linguagem de programação, como contém somente instruções abstratas do que fazer, pode ser compilado para qualquer código de máquina. Em resumo, ao invés de escrever um programa em código de máquina para cada família de processdores, escreve-se o mesmo código numa linguagem de programação e está é compilada por um compilador específico daquela arquitetura.

2.5.4 Pseudocódigo

O pseudocódigo é uma maneira intermediária entre a linguagem natural e uma linguagem de programação de representar um algoritmo. Ela utiliza um conjunto restrito de palavras-chave, em geral na língua nativa do programador, que tem equivalentes nas linguagens de programação. Além disso, o pseudocódigo não requer todo a rigidez sintática necessária numa linguagem de programação, permitindo que o aprendiz se detenha na lógica do algoritmos e não no formalismo da sua representação. Na medida que em se obtém mais familiaridade com os algoritmos, então o pseudocódigo pode ser traduzido para uma linguagem de programação.

Algoritmo 4 Exemplo de Pseudocódigo.

leia x, y {Esta linha é um comentário}
se $x > y$ **então**
    escreva "x é maior“
senão-se $x < y$ **então**
    escreva "y é maior"
senão
    escreva "x e y são iguais"
fim-se

Na listagem 4 é mostrado um exemplo de pseudocódigo escrito em português para escrever o maior valor entre, x ou y. As palavras **leia, se, então, senão, senão-se, fim-**

se e escreva são palavras-chave que representam estruturas presentes em todas as linguagens de programação. Entretanto, no pseudocódigo não é necessário se preocupar com detalhes de sintaxe (como ponto-e-vírgula no final de cada expressão) ou em formatos de entrada e saída dos dados. Deste modo, o enfoque no desenvolvimento do algoritmo fica restrito a sua lógica em si, e não na sua sintaxe para representação em determinada linguagem.

Por exemplo, considere o código do programa a seguir que implementa na linguagem de programação C o algoritmo 4. Veja como o mesmo requer uma sintaxe bem mais rígida do que o seu algoritmo correspondente Isso acontece pois para que o compilador C possa entender o programa desenvolvido, é necessário que sejam respeitadas algumas exigências da linguagem, como por exemplo:

- ▷ todo programa em C inicia sua execução na função `main()`, que é obrigatória;
- ▷ para que certas funções sejam acessíveis, é necessário incluir a bilbioteca `stdlib.h`;
- ▷ todas as linhas que contém instruções devem terminar com ponto-e-vírgula;
- ▷ os blocos de instruções são delimitados por chaves;
- ▷ linhas de comentários[2] são iniciadas por duas barras `//`;
- ▷ blocos de comentários são delimitados por `/*` e `*/`;

Este conjunto de regras demonstra como o compilador (nesse caso o compilador C) requer estruturas bem rígidas para poder processar (entender) o programa. Para facilitar o entendimento das estruturas algorítmicas que serão estudadas, os algoritmos apresentados aqui serão escritos em pseudocódigo, sendo fundamental que o estudante consiga entender a a correspondência entre os mesmos e a sua representação em uma linguagem de programação.

```
#include <stdio.h>

int main(){
  int x, y;    // isto é um comentário de linha

  /* isto é um comentário
     de bloco              */
  printf("\ndigite x:");
```

[2]Os textos escritos dentro de linhas e/ou blocos de comentários são ignoradoas pelo compilador e servem para que o programador mantenha o código documentado

```
  scanf("%i",&x);
  printf("\ndigite y:");
  scanf("%i",&y);

  if (x>y) {
    printf("x é maior\n");
  } else if (x<y) {
    printf("y é maior\n");
  } else {
    printf("x e y são iguais\n");
  }
}
```