

# Introdução

Nesta parte, vamos explorar a noção de “algoritmo”. Antes que consigamos executar um programa em um computador, o programa deve ser codificado em uma linguagem de programação da escolha do programador. E, antes de codificar o programa, o programador deve ter escolhido (inventado, descoberto) um *algoritmo* para resolução do problema.

Portanto, a criação de um algoritmo *precede* a própria escrita do programa na linguagem de programação. Antes de inventar um algoritmo para resolver o problema, ou escolher um algoritmo já pronto, não há como codificá-lo numa linguagem de programação, muito menos como executá-lo em um computador.

A noção de algoritmo é uma das noções mais básicas em computação. Compreender essa noção, e seus limites, é muito importante. Sem essa noção, ou com uma idéia confusa a seu respeito, a tarefa de se codificar programas e, através deles, usar um computador para se obter respostas para problemas práticos, fica bastante prejudicada e bem mais árida. Uma compreensão inadequada dessas noções oferece o risco de se escolher algoritmos inadequados para um certo problema, desperdiçando-se recursos, tais como tempo de execução e espaço de memória, quando da execução do programa correspondente. Ou pior ainda, como veremos, pode-se buscar soluções para problemas para os quais simplesmente não existem algoritmos capazes de resolvê-los. Nesse último caso, simplesmente não existirá um programa de computador capaz de sempre obter respostas corretas para o problema.

Vamos abordar a noção de algoritmo, dividindo a tarefa em três grandes partes:

- O que são algoritmos e o que os caracteriza.
- Como algoritmos se transformam em programas de computador úteis.
- Quais os limites dessa noção de algoritmo.

Sendo esse um texto voltado à iniciação em programação de computadores, não vamos usar de formalismos matemáticos para descrever a noção de algoritmos de maneira totalmente precisa. Isso seria necessário em outros contextos, onde se deseja realmente *provar matematicamente* propriedades de algoritmos e dos programas a eles associados. Não temos necessidade disso. A abordagem será, portanto, mais discursiva. Porém, as noções e idéias básicas poderão, mesmo assim, ser perfeitamente compreendidas, num nível de detalhe suficiente para suportar plenamente a tarefa subsequente de programação.

A primeira parte descreve o que são algoritmos e quais as suas principais características, ou seja, que condições deve uma idéia, ou plano de execução, satisfazer para que possa ser encarada como um algoritmo legítimo. No fundo, o algoritmo nada mais é que uma “receita” que permite resolver um dado problema.

A segunda parte tece uma conexão entre um algoritmo e os possíveis programas que os descrevem. Resumidamente, os programas são uma descrição precisa dos algoritmos, em uma “linguagem” que o computador possa “entender”. Assim, conseguiremos dirigir o comportamento do computador de tal forma que se obtenha respostas para o problema cujo algoritmo está codificado no programa em execução.

A terceira parte explora quais as fronteiras de problemas que podem ser resolvidos algoritmicamente, ou seja, automaticamente com o auxílio de computadores. Queremos saber se há algoritmos para resolver problemas práticos. Ou talvez alguns problemas sejam insolúveis de forma automática? E quanto às necessidades de recursos computacionais para resolver problemas de forma algorítmica? Por exemplo, será que existem problemas importantes para os quais o tempo de execução seja sempre muito longo? Ou o uso de memória do computador seja demasiado? Se afirmativo, então esses problemas podem nunca serem resolvidos de forma adequada em computadores! Quão longo seria o tempo mínimo necessário, ou quanto de memória seria necessário, para um algoritmo eficiente tratar o problema, caso existam tais algoritmos eficientes?