



Algoritmos e Linguagem de Programação

Capitulo 1

Introdução Algoritmos

Prof. Me. Renato Carioca Duarte



O que é Programação ?

- O computador não tem nenhuma inteligência; apenas faz as coisas de maneira muito rápida. Cabe aos programadores dar essa inteligência aos computadores.
- A única forma que existe para fazer isso é desenvolvendo programas que guiem nosso computador na busca de soluções de problemas.
- Para programar o computador é necessário 2 atividades:
 - Saber resolver um problema.
 - Saber escrever o programa em uma linguagem de programação.



Aprender a Programar

- A aprender a programar é o mesmo que estudar os processos computacionais.
- Processos computacionais são como seres abstratos que habitam os computadores.
- Esses seres abstratos agem manipulando outras abstrações, que chamamos de dados.
- Programar é direcionar este processo para obter um resultado.
- Para programar bem, você deve coordenar ações e manipular abstrações.



Linguagem de Programação

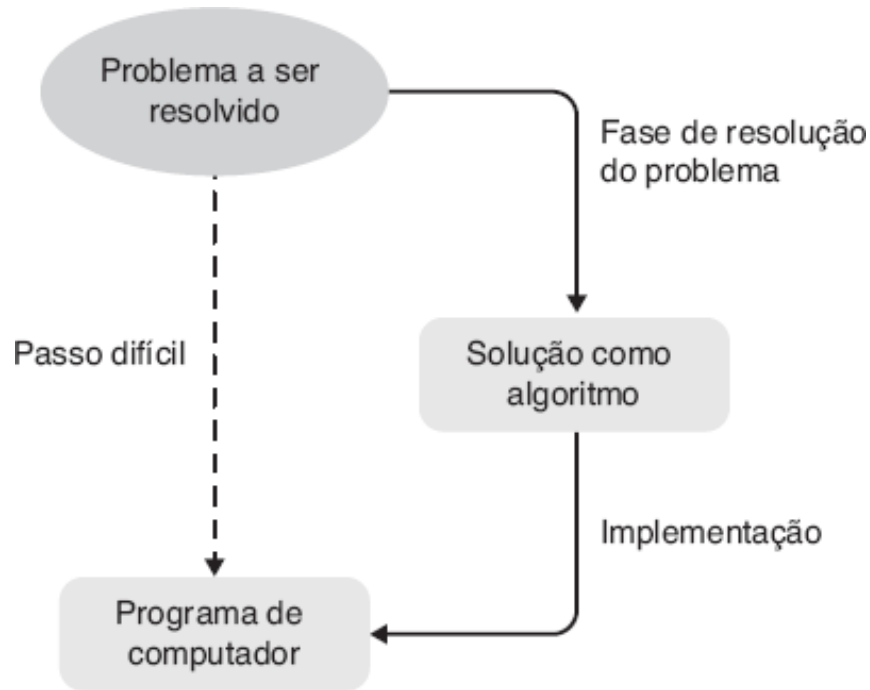
- **Para programar, é óbvio, você precisa aprender uma linguagem de programação.**
- Os conceitos de programação serão exemplificados na linguagem C#.
- Importante: os conceitos de programação têm prioridade sobre C#. Conhecendo esses conceitos, você será capaz de reconhecê-los em outras linguagens.
- C# é uma linguagem poderosa que não vai servir apenas para você aprender a programar. Muitos programas de computador úteis, de uso profissional, podem ser feitos por meio de C#.



Algoritmos

- Antes de programar com uma linguagem específica é necessário entender como trabalhar em um nível mais abstrato.
- Para isto é necessário **dominar o conceito de algoritmos** e como **usar algoritmos** na programação de computadores.
- **Algoritmo é um conjunto das regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito e ordenado de passos.**
- Cada passo deve contribuir para se chegar mais perto da solução final de um problema, ou seja, cada passo deve avançar em direção à solução.
- Outra consequência dessa definição é que a solução de um problema deve vir por “passos”, ou seja, para problemas reais não existe atalho.
- Não adianta tentar resolver um problema computacional indo direto à solução como um programa.

Algoritmos



- Antes de programar, você deve pensar em uma solução na forma de algoritmo, sem as amarras que a linguagem de programação impõe.
- Na maioria das vezes, para problemas complexos, uma solução assim terá muitos erros e não irá atender aos requisitos da solução desejada.
- Somente problemas muito simples podem ser implementados diretamente em uma linguagem sem uma reflexão sobre a solução.



Algoritmos

- Podemos, desta forma, elencar algumas propriedades comuns aos algoritmos computacionais:
 1. Cada operação deve ser bem definida. Não deve haver dúvida sobre o seu significado, isto é, a operação não pode, em hipótese alguma, conter ambiguidades.
 2. Cada operação deve ser efetiva: deve contribuir para a solução do problema. A retirada de uma operação efetiva prejudica a solução.
 3. Teoricamente, uma pessoa munida de papel e lápis deve poder seguir os passos do algoritmo. Popularmente chamamos isso de “fazer o chinês”, quer dizer, se tivermos tempo e paciência suficientes, devemos ser capazes de anotar em um papel cada passo de um algoritmo e encontrar a solução para o problema.
 4. O algoritmo deve terminar em um tempo finito.

Algoritmos

- A partir de alguma **entrada de informações**, um algoritmo deve **processar essas informações** e, em um tempo finito, fornecer uma **saída como solução** de algum problema.



- Computadores são máquinas para processar informações, mas precisam ser programados para tal. Possuem uma linguagem própria (C#, Java, Python, C, PHP, por exemplo) por meio da qual podemos passar instruções sobre o que deve ser feito.



Algoritmos

- **Análise do Problema:**
 1. Ler atentamente o enunciado do problema até entendê-lo bem.
 2. Identificar os dados de entrada.
 3. Identificar as saídas (resultados esperados).
 4. O que o programa deve fazer (seu objetivo), isto é, como transformar as entradas em saídas.
 5. Identificar se existem valores ou dados intermediários, necessários para transformar entradas em saídas





Algoritmos

- É útil fazer um paralelo entre algoritmos computacionais e não computacionais. Nós lidamos com algoritmos não computacionais o tempo todo em nosso cotidiano. **Uma receita culinária é uma espécie de algoritmo; no caso, um algoritmo não computacional.** Digamos que você queira fazer um pão. Podemos ser bem genéricos e vagos na receita:

- 1.coloque meio quilo de farinha em um recipiente adequado;*
- 2.adicione 10 gramas de sal;*
- 3.adicione 10 gramas de fermento;*
- 4.adicione 350 ml de água;*
- 5.misture o conjunto e sove a massa até que fique bem elástica;*
- 6.deixe crescer por 2 horas;*
- 7.rove mais um pouco a massa e faça o formato de pão;*
- 8.coloque em uma forma adequada, deixando crescer por 1 hora;*
- 9.asse no forno a 220 °C por 30 minutos.*



Algoritmos

- A receita do pão funciona, **mas é bem superficial**. Diversos detalhes foram deixados de lado. O que é um recipiente adequado? O que significa “bem elástica”?
- As instruções, ou seja, o algoritmo da receita de pão, é adequado para um padeiro experiente, mas não seria para um padeiro novato. Um padeiro novato precisaria de mais detalhes para fazer um pão corretamente.
- E se fôssemos criar um robô padeiro, uma máquina que fizesse pão?
- Neste caso, deveríamos criar um algoritmo mais detalhado. Nossas instruções teriam de ser particularizadas de acordo com o nível de entendimento da máquina, isto é, com seu nível de inteligência.
- Certamente, um computador seria o guia dessa máquina de fazer pão. Como os computadores têm uma inteligência limitada, o programa precisaria ser bastante extenso.



Um Algoritmo Computacional Simples

- Pensar algoritmicamente é o mesmo que pensar com as limitações de um computador.
- Imagine que você tenha que procurar o significado da palavra **“pneumoultramicroscopicossilicovulcanoconítico”** em um dicionário.
- Vamos começar com uma solução ruim, porém extremamente simples, para ensinar a um computador como achar uma palavra em um dicionário.
- A maneira mais simples e menos inteligente de fazer esta busca é olhar palavra por palavra, desde a primeira até encontrarmos a palavra procurada.
- Se chegarmos ao final do dicionário é porque este não contém uma definição para a palavra, que talvez não exista em nossa língua.



Um Algoritmo Computacional Simples

- A solução poderia ser:

Leia cada palavra do dicionário até encontrar a palavra procurada ou não restarem mais palavras a serem lidas.

Se encontrar a palavra, imprima seu significado.

- Esta primeira solução é apenas um aquecimento. Serve para você pensar em como resolver seu problema, sem entrar em detalhes.
- Neste ponto você se preocupa apenas em esboçar uma solução.
- Esta solução ainda não é um algoritmo computacional; não tem detalhes suficientes para poder ser traduzida em uma linguagem de programação. Não tente resolver o problema de uma vez.



Um Algoritmo Computacional Simples

- Pode parecer perda de tempo começar de maneira tão abstrata e depois ir refinando a solução, com mais detalhes.
- Vamos agora dividir em passos. Algoritmo que busca a definição de uma palavra em um dicionário:

Passo 1: Leia a primeira palavra do dicionário.

Passo 2: Se for a palavra procurada, imprima a definição e termine.

Passo 3: Se for a última palavra, imprima “Palavra não existe” e termine.

Passo 4: Leia a próxima palavra.

Passo 5: Volte para o passo 2.

- Este algoritmo seria a forma de solução a que intuitivamente poderíamos chegar. Apesar de correta, não é a melhor forma de solução computacional.



Um Algoritmo Computacional Simples

- Quando um computador tiver de repetir uma mesma tarefa, é melhor indicarmos isso de imediato, antes dos comandos que serão repetidos.
- No algoritmo apresentado, apenas no passo 5 aparece uma ordem de repetição, na forma de um desvio do fluxo de execução do algoritmo.
- Nos primórdios da Computação era comum usarmos comandos como este do passo 5. **Com o aumento da complexidade dos programas, notou-se que comandos que desviam o fluxo de execução criavam um código difícil de corrigir, pois não podemos ter certeza do local em que o programa está executando em cada momento.**
- Em nosso exemplo simples, isto não é evidente, mas imagine um programa com milhares de linhas de código e centenas de desvios. Criou-se até um termo pejorativo para este tipo de código: código espaguete.
- **A solução encontrada foi usar blocos de comandos que são repetidos de acordo com uma condição.**



Um Algoritmo Computacional Simples

- Com isto controlamos melhor o fluxo de execução de um algoritmo e sempre sabemos qual parte do código está sendo executada. É a chamada **Programação Estruturada**. Vamos modificar nosso algoritmo, antecipando o comando de repetição e detalhando melhor cada passo.
- Algoritmo que busca a definição de uma palavra em um dicionário:
 - Passo 1: Leia a primeira palavra do dicionário.*
 - Passo 2: Repita:*
 - Passo 2.1: Se a palavra lida for a palavra procurada, faça:*
 - Passo 2.1.1: Imprima a definição*
 - Passo 2.1.2: Termine a execução do algoritmo.*
 - Passo 2.2: Se for a última palavra:*
 - Passo 2.2.1: Imprima “Palavra não existe”*
 - Passo 2.2.2: Termine a execução do algoritmo.*
 - Passo 2.2: Leia a próxima palavra.*



Um Algoritmo Computacional Simples

- Esta forma de apresentar um algoritmo ainda é visualmente confusa.
- Para melhorar seu aspecto, foi convenicionado que subpassos como 2.1 ou 2.1.1 seriam **identados**, ou seja, vamos deslocar seu início na linha para que visualmente possamos identificar os blocos de execução.
- Vejamos como ficaria:

Algoritmo que busca a definição de uma palavra em um dicionário:

Leia a primeira palavra do dicionário.

Repita:

Se a palavra lida for a palavra procurada:

Imprima a definição

Termine a execução do algoritmo.

Se for a última palavra:

Imprima "Palavra não existe"

Termine a execução do algoritmo.

Leia a próxima palavra.



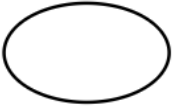
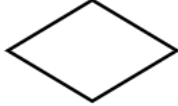
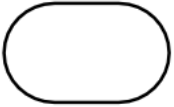
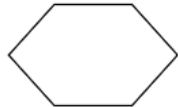
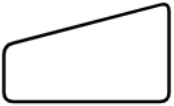


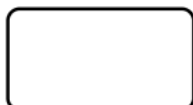
Um Algoritmo Computacional Simples

- Cada vez que um passo do algoritmo é deslocado na coluna da linha, quer dizer que este comando é um subpasso do passo anterior que começa em uma coluna menor.
- Esta é a forma usual de apresentar algoritmos.
- Desta maneira, não precisamos numerar cada passo e visualmente identificamos os blocos de execução.



Fluxograma

- Um fluxograma é baseado em símbolos que representam os passos de um algoritmo. Cada símbolo representa um tipo de ação a ser executada.
- Símbolos usados em fluxogramas:

 Início	 Decisão
 Fim	 Preparação
 Entrada de dados	 Entrada ou Saída de dados
 Saída de dados	 Processamento



Fluxograma

- Nos primórdios da Computação usava-se muito este tipo de recurso.
- Com o advento de linguagens estruturadas e o aumento da complexidade dos programas, notou-se que a forma gráfica acabava complicando o entendimento do processo de solução de problemas.
- Quando os algoritmos se tornam mais complexos, os fluxogramas simplesmente se tornam confusos, dificultando o entendimento da lógica da solução.
- Outro problema marcante é a sua dificuldade de manutenção. Até mesmo pequenas modificações podem levar a um grande trabalho de redesenho.
- Os fluxogramas ainda são usados, porém para demonstrar a lógica de pequenos trechos ou ilustrar uma visão geral de uma solução.



Fluxograma

Um fluxograma simples,
representando o algoritmo da
busca de palavra no dicionário

