Ariel da Silva Dias

Algoritmos e programação I: introdução à programação com C#

Senac

Dados Internacionais de Catalogação na Publicação (CIP)
(Simone M. P. Vieira – CRB 8ª/4771)

Dias, Ariel da Silva
Algoritmos e programação I : introdução à programação com C# / Ariel da Silva Dias. – São Paulo : Editora Senac São Paulo, 2022. (Série Universitária)

Bibliografia.
e-ISBN 978-85-396-3471-2 (ePub/2022)
e-ISBN 978-85-396-3472-9 (PDF/2022)

1. Desenvolvimento de sistemas 2. Linguagem de programação 3. Algoritmos – Conceitos 4. Pensamento computacional 5. Programação estruturada I. Título. II. Série

22-1583t

CDD – 005.13
003
BISAC COM051300
COM051230

Índice para catálogo sistemático:
1. Linguagem de programação : Algoritmos 005.13
2. Desenvolvimento de sistemas 003

ALGORITMOS E PROGRAMAÇÃO I: INTRODUÇÃO À PROGRAMAÇÃO COM C#

Ariel da Silva Dias

Senac

Administração Regional do Senac no Estado de São Paulo

Presidente do Conselho Regional
Abram Szajman

Diretor do Departamento Regional
Luiz Francisco de A. Salgado

Superintendente Universitário e de Desenvolvimento
Luiz Carlos Dourado

Editora Senac São Paulo

Conselho Editorial
Luiz Francisco de A. Salgado
Luiz Carlos Dourado
Darcio Sayad Maia
Lucila Mara Sbrana Sciotti
Luís Américo Tousi Botelho

Gerente/Publisher
Luís Américo Tousi Botelho

Coordenação Editorial/Prospecção
Dolores Crisci Manzano
Ricardo Diana

Administrativo
grupoedsadministrativo@sp.senac.br

Comercial
comercial@editorasenacsp.com.br

Acompanhamento Pedagógico
Mônica Rodrigues dos Santos

Designer Educacional
Estenio Azevedo

Revisão Técnica
Celso Vital Crivelaro

Preparação e Revisão de Texto
Karen Daikuzono

Projeto Gráfico
Alexandre Lemes da Silva
Emília Corrêa Abreu

Capa
Antonio Carlos De Angelis

Editoração Eletrônica
Marcella Rigazzo Maiolino

Ilustrações
Marcella Rigazzo Maiolino

Imagens
Adobe Stock Photos

E-book
Rodolfo Santana

Editora Senac São Paulo
Rua 24 de Maio, 208 – 3º andar
Centro – CEP 01041-000 – São Paulo – SP
Caixa Postal 1120 – CEP 01032-970 – São Paulo – SP
Tel. (11) 2187-4450 – Fax (11) 2187-4486
E-mail: editora@sp.senac.br
Home page: http://www.livrariasenac.com.br

Sumário

Capítulo 1
Introdução ao pensamento computacional, 7

1 Habilidades, contribuições e pilares do pensamento computacional, 9

2 Resolução de um problema usando pensamento computacional, 13

Considerações finais, 16

Referências, 16

Capítulo 2
Introdução aos algoritmos e programação por blocos, 17

1 Introdução e conceitos de algoritmos, 18

2 Descrição narrativa de um algoritmo e implementação por blocos, 23

Considerações finais, 29

Referências, 30

Capítulo 3
Variáveis e programação por blocos, 31

1 Programação por blocos: variáveis e entrada de dados, 32

2 Programação por blocos: operações aritméticas com as variáveis, 42

Considerações finais, 44

Referências, 45

Capítulo 4
Tomando decisões e repetindo instruções, 47

1 Expressões lógicas e fluxo de controle, 48

2 Programação por blocos: *se* (*if*) e *se/senão* (*if/else*), 52

3 Programação por blocos: repetição, 55

Considerações finais, 57

Referências, 57

Capítulo 5
Linguagem de programação imperativa, 59

1 Programação imperativa, 61

2 Variáveis e constantes, 64

3 Entrada e saída de dados, 67

Considerações finais, 71

Referências, 71

Capítulo 6
Estruturas condicionais, 73

1 Programação imperativa: estrutura de controle linear e condicional, 75

2 Expressões relacionais e lógicas, 78

3 Condicional simples (*if*) e composta (*if/else*), 81

Considerações finais, 88

Referências, 89

Capítulo 7
Estruturas de repetição, 91

1 Programação imperativa: estrutura de repetição (*while* e *do/while*), 92

2 Programação imperativa: estrutura de repetição (*for*), 97

3 Exemplo de um caso com laço de repetição, 99

Considerações finais, 104

Referências, 104

Capítulo 8
Programação estruturada, 105

1 Conceito de programação estruturada (funções), 106

2 Exemplos e aplicações da utilização de funções, 113

Considerações finais, 121

Referências, 122

Sobre o autor, 125

Capítulo 1

Introdução ao pensamento computacional

Antes de iniciarmos a nossa ação de programar, a primeira etapa a ser considerada é ter o pensamento computacional. Trata-se de um processo responsável por decompor um determinado problema em etapas simples o suficiente para que até um computador possa entender (lembre-se que um computador só entende zeros e uns, ou seja, quanto mais simples para ele, melhor).

Os computadores, como deve ser de seu conhecimento, seguem literalmente as instruções que damos e, às vezes, os resultados são inesperados. Afinal, se não fornecermos aos computadores instruções precisas

e detalhadas, seu algoritmo pode não ter o mesmo comportamento e as ações vitais que a maioria das pessoas considera natural.

Vejamos um exemplo de uma atividade rotineira como escovar os dentes. Observe que, a princípio, parece uma tarefa um tanto quanto simples, mas que, na verdade, engloba muitas etapas. Primeiro, precisamos de uma escova e creme dental, além de uma pia com água corrente. O próximo passo é colocar o creme dental na escova. É importante não nos esquecermos de abrir a água e passar a escova com o creme dental por baixo. E assim prosseguimos até a finalização do processo, que já é de conhecimento geral. Enfim, como pode-se observar, uma atividade simples envolve muitas etapas, se perdermos uma etapa ou colocarmos uma fora de ordem, poderá acabar em confusão.

Assim, o computador (não limitando apenas a ele, mas também aos dispositivos móveis como smartphones e tablets) é fortemente aplicado na resolução de problemas em diversas áreas: educação, saúde, transporte, financeiro, entre outras. Por exemplo, as planilhas eletrônicas, os aplicativos de pedir refeição ou de solicitar um transporte, todos eles possuem uma lógica que, por trás, nos ajudam a resolver problemas. No entanto, antes que um problema possa ser resolvido, o próprio problema e as maneiras pelas quais ele pode ser resolvido precisam ser compreendidos.

O pensamento computacional ajuda as pessoas a desenvolverem habilidades que sejam atraentes para futuras oportunidades de emprego (mesmo se você não almejar ser um programador ou programadora). Afinal, embora as habilidades de tecnologia sejam muito importantes, são as habilidades mais suaves de raciocínio (como organização de itens em uma lista de compras ou planejar o melhor caminho para se deslocar de casa para o trabalho) e a solução de problemas (como montar um checklist de tópicos para uma reunião) que os empregadores realmente consideram atraentes. Desse modo, ter habilidade para resolver problemas é o princípio essencial para compreender o pensamento computacional.

1 Habilidades, contribuições e pilares do pensamento computacional

O pensamento computacional pode ser dividido em quatro habilidades essenciais: decomposição, reconhecimento de padrões, abstração de padrão e projeto de algoritmos (FORBELLONE; EBERSPACHER, 2005). A seguir, veremos cada uma dessas habilidades, que são pilares fundamentais do pensamento computacional.

1.1 Decomposição

O primeiro pilar é a decomposição. A decomposição é a habilidade de dividir problemas complexos em pedaços menores e que possam ser mais bem gerenciados.

Qualquer atividade, das mais simples, como escovar os dentes, preparar o café da manhã ou ler um livro, até as mais complexas, como o processo de logística de entrega de produtos por uma empresa, funcionará. Precisamos apenas dividir a tarefa em pequenos passos simples. Quanto mais pudermos decompor um problema, mais fácil será resolvê-lo. Essa habilidade permite compreender como é importante dar instruções que sejam exatas.

Um ótimo exercício de decomposição é decompor os componentes de uma bicicleta. Se tentarmos, observaremos que, em um primeiro momento, podemos decompor uma bicicleta em quadro, rodas, guidom e engrenagens. Mas é só isso? Não! Podemos ainda quebrar cada componente um pouco mais, por exemplo, uma roda é composta por raios, aro, pneu e válvula. Logo, podemos quebrar a roda nestes componentes. Podemos fazer o mesmo com o quadro, guidom e engrenagens. Esse estágio também permite desenvolver uma melhor compreensão do problema que você enfrenta, identificando todos os componentes em detalhes.

Importante salientar que a decomposição nos permite avaliar um problema e descobrir todas as etapas necessárias para que a tarefa aconteça. Então, a decomposição é uma habilidade importante para a vida no futuro, quando precisarmos assumir tarefas maiores, sejam elas de programação ou de qualquer outra área em que estejamos atuando. Com isso, seremos capazes de delegar melhor tarefas para uma equipe ou grupo de trabalho, bem como saber gerenciar melhor o nosso tempo.

NA PRÁTICA

Quando vamos a um restaurante, o problema principal a ser resolvido é a fome. Logo, para que a fome seja aliviada, colocamos no prato pequenas porções das opções do buffet, como arroz, feijão e uma proteína. Observe que o problema foi decomposto, afinal, cada um desses elementos, quando forem digeridos, serão responsáveis por saciar a fome.

1.2 Reconhecimento de padrões

O segundo pilar é o reconhecimento de padrões, semelhanças e tendências dentro de um determinado problema. Essa habilidade é essencial, pois, se alguns problemas são de natureza semelhante (tanto no problema atual que está sendo enfrentado quanto nos problemas anteriores), há uma boa chance de que eles possam ser resolvidos usando técnicas semelhantes ou repetidas.

Certamente, você já viajou para algum lugar que nunca tinha ido anteriormente. Já percebeu como, em sua primeira ida, o caminho pareceu mais longo que nas idas posteriores? Isso ocorre porque você não reconhecia padrões na primeira viagem. No entanto, após já conhecer o caminho, identificou padrões em suas viagens seguintes.

Podemos notar essa habilidade nos atletas. Quando eles se preparam para uma corrida de 50 km, por exemplo, a primeira tentativa nunca é muito bem-sucedida: ou começou muito rápido e no meio do caminho

já estava cansado, ou começou muito devagar e acabou não conseguindo completar o trajeto em um bom tempo.

Em uma próxima tentativa, nos recordamos da experiência anterior e certamente isso nos ajuda a ter um desempenho diferente da primeira vez. Essa, então, é a habilidade essencial para criar soluções eficientes e economizar tempo no longo prazo. Durante a corrida, o atleta pode ajustar seus passos mediante a memória da primeira tentativa.

Também é possível notar essa habilidade no dia a dia. Para quem anda de bicicleta, por exemplo, mesmo depois de ficar longo tempo sem praticar, quando retoma a atividade, lembra-se de como fazer. Observe que a pessoa não regride ao estado inicial de quando ainda estava aprendendo a andar.

O reconhecimento de padrões é uma habilidade-chave quando se trata de criar soluções eficientes e eficazes para determinados problemas. Essencialmente, é dar uma resposta para a seguinte pergunta: o que aprendemos no passado que pode nos ajudar a resolver esse problema?

NA PRÁTICA

Já observou a agilidade das crianças em interagir com um smartphone? Elas rapidamente acessam os aplicativos e interagem com eles. O mais interessante é que muitas delas não sabem ler (em razão da pouca idade), mas conseguem acessar plataformas de vídeo, como o YouTube ou YouTube Kids e os seus jogos favoritos. Isso ocorre pelo reconhecimento de padrões. Elas sabem que, toda vez que clicarem em um ícone vermelho em formato de seta, será aberto um local cheio de vídeos. Assim, rapidamente encontram os seus personagens favoritos para assisti-los.

1.3 Abstração

Qual a função de um arquiteto? Colocar no papel uma casa ou um edifício. Observe que não podemos morar em um papel ou explorar cada canto daquele desenho, caminhando internamente por aquela construção (no papel). Essa é a terceira habilidade do pensamento computacional que diz que, em vez de olhar para detalhes específicos, devemos ter a capacidade de filtrar os elementos desnecessários de um problema para que você se concentre apenas nos elementos importantes. Em outras palavras, esqueça o fato de ser apenas um desenho, abstraia o fato de estar apenas em um papel.

Qual a vantagem da abstração? Bem, nesse exemplo do arquiteto, a planta desenhada por ele incluirá todos os principais elementos de design que serão incluídos na construção final. Quaisquer ajustes ou acréscimos podem ser discutidos com os clientes nessa fase, antes do início do trabalho de construção. Com isso, economiza-se dinheiro e tempo de obra.

Voltando ao exemplo anterior, um corredor quando treina abstrai alguns elementos como temperatura, clima (chuva ou sol), vento, se existe algum atleta mais veloz que ele, entre outros. Todo o foco dele está na corrida, ou seja, em desempenhar o seu papel de corredor, ignorando assim qualquer outro elemento.

De modo semelhante, no exemplo da bicicleta, se uma pessoa aprendeu a andar em uma bicicleta pequena, não terá dificuldade de andar em uma bicicleta de tamanho maior. Também abstrai outros fatores como espessura dos pneus, se a bicicleta possui ou não marcha, entre outros.

Identificar as informações cruciais em um problema e desconsiderar as informações irrelevantes é uma das partes mais difíceis do pensamento computacional.

NA PRÁTICA

Montar um projeto utilizando peças Lego é uma ótima maneira de resolver problemas. As lojas vendem kits com algumas dezenas ou até mesmo com milhares de peças. No entanto, se seu objetivo é construir um carro com Lego, você certamente não utilizará todas as mil peças de um kit. Existem muitas peças que você poderá ignorar, ou seja, poderá abstrair aquelas menos importantes e que não contribuem com a solução do seu problema.

1.4 Algoritmos

O pensamento computacional envolve o desenvolvimento de soluções para um problema e, entre essas soluções, temos os algoritmos. Especificamente, o pensamento algorítmico cria regras sequenciais a serem seguidas para resolver um problema.

Um bom algoritmo é aquele que pode ser passado para outra pessoa seguir sem a necessidade de nenhuma explicação extra (FORBELLONE; EBERSPACHER, 2005). O mundo está cheio de algoritmos: receitas culinárias, montagem de móveis, trocar o pneu de um carro, escovar os dentes, comprar um produto em um mercado, entre outros.

Quanto mais pensamos sobre algoritmos, mais começamos a perceber que seguimos muitos conjuntos de instruções todos os dias. Os exemplos apresentados são claramente sinalizados, enquanto outros são mais inatos, como manter o limite de velocidade ao dirigir um carro, subir ou descer uma escada ou seguir um código de conduta em uma empresa.

2 Resolução de um problema usando pensamento computacional

Anteriormente, foi possível compreender que o pensamento computacional é dividido em quatro habilidades essenciais: decomposição,

reconhecimento de padrões, abstração de padrão e projeto de algoritmos. Agora, vamos pegar um exemplo do cotidiano, trocar o pneu de um carro, e, de posse dessas quatro habilidades, resolvê-lo usando o pensamento computacional.

2.1 Exemplo: troca de pneu de um carro

Em uma viagem, o pneu do carro furou e, estando em um local totalmente deserto, sem borracharia, será necessário realizar a sua troca. Observe que temos um problema: trocar o pneu furado.

Primeiramente, vamos decompor esse problema em: abrir o porta-malas, retirar o estepe, colocar o macaco embaixo do carro, levantar o carro, remover os parafusos, retirar o pneu furado, colocar o estepe, abaixar o carro e guardar o macaco.

Observe que, nessa decomposição, muita coisa ficou faltando. Primeiramente, precisamos verificar se o estepe está vazio ou cheio. Afinal, se estiver vazio, a melhor saída é contactar a seguradora para fazer o reboque (o que pode levar horas). No entanto, se estiver cheio, podemos começar a trocar.

Um passo importante é levantar o carro, porém, do lado onde está o pneu furado. Parece óbvio, mas lembre-se que o problema deve ser decomposto em instruções simples para que o computador possa entender (ou, neste caso, que um amador possa entender a troca de pneu).

Outro passo importante é a remoção dos parafusos. Precisamos desaparafusar as rodas, porém, os veículos em sua maioria possuem quatro ou cinco parafusos na roda. Logo, precisamos colocar a chave no primeiro parafuso, rotacionar a chave e desaparafusar. Em seguida, precisamos colocar a chave no segundo parafuso, rotacionar a chave e desaparafusar. Observe que serão quatro (ou cinco) instruções iguais, as quais seguem um padrão que é colocar a chave no parafuso, rotacionar a chave e desaparafusar.

Além disso, abstraia o fato de um parafuso demorar mais tempo do que o outro para ser removido, isso não afetará a resolução do problema. Acrescenta-se ainda que, mesmo existindo diferentes tipos de carros, o processo de troca de pneu é o mesmo, ou seja, segue a mesma sequência lógica de instruções.

Após remover todos os parafusos, chega a hora de remover a roda que possui o pneu furado. Remova e coloque-a de lado. Em seguida, vamos colocar o estepe. Observe que agora colocaremos os parafusos. Apesar de ser uma instrução diferente (afinal, antes nós removemos os parafusos), a ação é semelhante, pois precisamos colocar a chave e parafusar quatro ou cinco vezes. Novamente nos deparamos com instruções-padrão que devem ser executadas.

Agora que o estepe está no lugar do pneu furado, podemos remover o macaco, para isso, é necessário primeiramente abaixar o carro e, somente depois, podemos retirar o equipamento.

Por fim, será possível guardar as ferramentas e seguir na estrada, afinal, como temos as habilidades do pensamento computacional e como seguimos os quatro pilares fundamentais, a troca do pneu foi realizada com sucesso.

IMPORTANTE

Todo algoritmo é uma sequência finita de instruções. Observe que, na troca de pneu, tivemos um início, que foi verificar se o estepe estava cheio ou não, e o fim, que foi sair com o carro. Se estivesse cheio, o fim do algoritmo seria sair com o carro, e, se estivesse vazio, teríamos o mesmo fim (sair com o carro), porém levado pelo guincho.

Considerações finais

Neste capítulo, pudemos compreender o conceito de pensamento computacional. Inicialmente, observamos que pensar computacionalmente não significa pensar igual a um computador (rápido e com precisão), mas, sim, dividir um problema em partes menores e, a partir de cada uma dessas partes menores, resolvê-lo.

Em seguida, conhecemos os quatro pilares do pensamento computacional e quais habilidades estão inerentes a esses pilares. Por exemplo, vimos que, ao decompor um problema, podemos resolvê-lo de modo mais rápido e direto. Aprendemos que o reconhecimento de padrões auxilia na tomada de decisão com base em um conhecimento prévio, e abstrair padrões nos permite focar no problema, eliminando distrações. Além disso, vimos que um algoritmo é uma sequência de instruções que, quando bem escrita, permite levar à solução de um dado problema.

Por fim, estudamos um caso prático, a troca de pneu, e pudemos colocar em prática cada uma das habilidades intrínsecas do pensamento computacional.

Referências

FEIJÓ, Bruno; CLUA, Esteban; SILVA, Flávio S. C. da. **Introdução à ciência da computação com jogos**: aprendendo a programar com entretenimento. Rio de Janeiro: Campus, 2009.

FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico. **Lógica de programação**: a construção de algoritmos e estruturas de dados. 3. ed. São Paulo: Pearson, 2005.