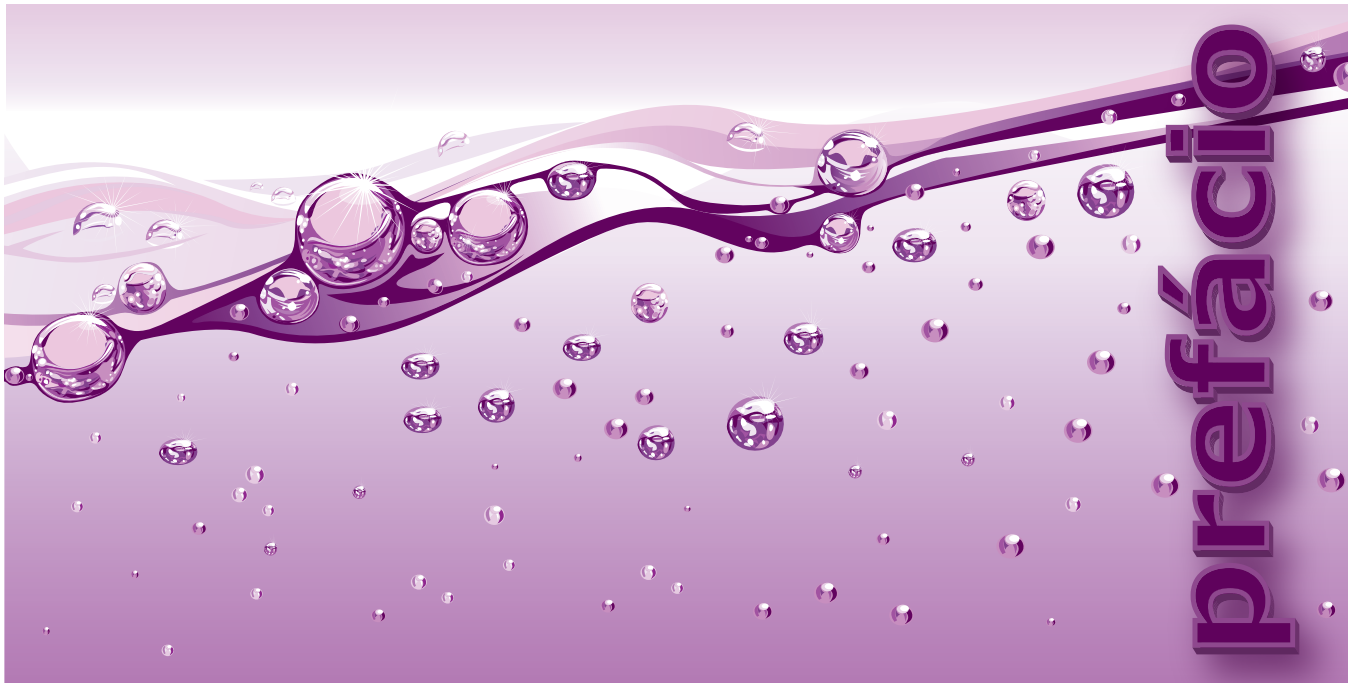

Objetivos e Público-alvo

ALGORITMOS E ESTRUTURAS DE DADOS constituem os alicerces da programação. Assim, para se tornar um bom programador, o aspirante precisa dominar bem essas ferramentas e saber qual delas utilizar no desenvolvimento de cada programa.

Estruturas de Dados é uma disciplina essencialmente de programação e, por isso, deve ser melhor apresentada com a adoção de uma linguagem de programação. Muitos algoritmos apresentados neste livro aparecem em forma de funções da linguagem C, em vez de em pseudolinguagem (como em muitos livros similares). Essa abordagem não apenas é menos formal como (espera-se) motiva o leitor e serve como fonte de referência para o programador.

Este livro destina-se primariamente a alunos de cursos das áreas de computação e informática que estejam cursando uma disciplina de construção de algoritmos e estrutura de dados. Portanto conhecimentos básicos de programação e matemática de nível médio são requeridos.

Organização do Livro

Esta obra é dividida em dois volumes. O **Volume 1** é dedicado às estruturas de dados fundamentais enquanto que o **Volume 2** devota-se ao estudo de busca e ordenação de dados.

Volume 2

O presente volume é dividido em cinco apêndices e três partes principais, que são:

- **Parte 1**, constituída pelos **Capítulos 1** e **2**, discute memórias internas e externas e como o tipo de armazenamento no qual os dados se encontram influencia o desempenho dos algoritmos que os processam.
- **Parte 2** é dedicada à busca e é composta pelos **Capítulos** de **3** a **9**.
- **Parte 3** cobre a área de ordenação de dados. Os **Capítulos** de **10** a **12** compõem essa parte.

Capítulo 1 descreve conceitos e tecnologias relacionados com armazenamento de dados. Esse conhecimento é importante por duas razões principais: (1) ele mostra que a eficiência de algoritmos que processam dados em memória secundária deve ser julgada de modo bem diferente do modo como é avaliada a eficiência de algoritmos que processam dados em memória principal e (2) ele ensina como melhorar o desempenho de programas utilizando conhecimento sobre hierarquias de memória.

Capítulo 2 discute brevemente as facilidade de processamento de arquivos providas pela biblioteca padrão de C e mostra várias maneiras pelas quais um arquivo pode ser processado. Processamento sequencial é usado em virtualmente todo o **Volume 2** a partir do **Capítulo 3** e processamento por acesso direto será necessário a partir do **Capítulo 6**. Logo o estudo desse último tipo de processamento poderá ser adiado para quando ele for de fato necessário.

Capítulo 3 introduz conceitos básicos associados a busca e, em especial, concentra-se numa categoria de busca efetuada em memória principal denominada *busca linear*. Nesse capítulo, as estruturas de dados utilizadas são listas indexadas e encadeadas convencionais bem como as mais recentes listas com salto.

Capítulo 4 lida com tabelas de busca implementadas como árvores binárias em memória principal. A primeira estrutura de dados discutida nesse capítulo é a árvore binária ordinária de busca. A seguir, esse capítulo explora as árvores binárias balanceadas AVL, que, devido às suas complexidades, requerem um esforço maior para que sejam completamente entendidas. A última estrutura de dados a ser discutida neste capítulo é a árvore binária de busca afunilada, que é baseada no conceito de localidade de referência discutido no **Capítulo 1**.

Capítulo 5 apresenta uma ferramenta de análise de algoritmos, denominada *análise amortizada*, que constitui uma alternativa para a análise assintótica discutida em detalhes no **Capítulo 6** do **Volume 1**. Esse novo tipo de análise é conveniente em situações nas quais a análise assintótica convencional, apesar de correta, resulta em avaliações consideradas pessimistas demais. A ideia básica que norteia esse tipo de análise é o exame de operações em conjunto (em vez de individualmente), de modo que as poucos operações dispendiosas quando combinadas com muitas operações menos onerosas resultam numa boa avaliação de desempenho para uma longa sequência de operações.

Capítulo 6 começa apresentando árvores multidirecionais descendentes de busca, que, apesar de não terem nenhuma utilidade prática, são importantes do ponto de vista didático, visto que são bem mais fáceis de entender e implementar do que árvores B e B+, que são discutidas mais adiante nesse capítulo.

Capítulo 7 lida com uma das mais importantes técnicas de implementação de tabelas de busca em memória principal. Essa técnica é denominada *dispersão* (*hashing*, em inglês) e consiste basicamente em associar cada chave a um índice de uma tabela de busca. Por meio dessa técnica, espera-se obter custo temporal constante para as operações básicas de busca, inserção e remoção.

Capítulo 8 discute o uso de dispersão em memória secundária. Em especial, esse capítulo explora duas técnicas frequentemente usadas na prática: dispersão estática e dispersão extensível.

Capítulo 9 discute algoritmos básicos para casamento de strings, que são essenciais em várias aplicações de processamento de documentos, tais como editores de texto, recuperação de informação e mecanismos de busca da internet. Nesse capítulo, a estrutura de dados *trie*, usada principalmente para representar strings, também é examinada.

Capítulo 10 descreve o conceito de lista de prioridade e apresenta diversas maneiras de implementação dessa estrutura de dados. Uma delas, o *heap* binário, merece destaque especial. Assim como arrays, *heaps* são usados apenas como estruturas básicas na implementação de estruturas de dados de maior nível de abstração.

Capítulo 11 ocupa-se de ordenação de dados em memória principal, que é um dos mais antigos e bem estudados problemas de programação. Embora alguns algoritmos de ordenação apresentados nesse capítulo sejam

fáceis de entender e implementar, outros levam um pouco mais de tempo para entender e requerem mais prática para implementar. Tipicamente, os algoritmos de ordenação mais fáceis de entender e implementar têm aplicações limitadas a pequenas quantidades de dados e vice-versa.

Capítulo 12 discute algoritmos de ordenação externa, que são elaborados para lidar com volumes de dados muito grandes que residem numa memória externa mais lenta (usualmente, um HD). Ou seja, esse capítulo lida com situações nas quais deseja-se ordenar um arquivo sem ter que mantê-lo integralmente em memória principal. O último tópico discutido neste livro é o processo de criação de uma árvore B+ usando um enorme arquivo de registros numa única operação. A discussão desse processo, denominado *inserção massiva* (*bulkloading*, em inglês), foi adiada para esse último capítulo porque ele requer que os referidos registros sejam ordenados em memória externa.

Os apêndices do **Volume 2** apresentam os seguintes conteúdos:

- **Apêndice A** apresenta os arquivos de dados usados nos diversos exemplos de busca e ordenação discutidos neste volume.
- **Apêndice B** expõe noções básicas de programação de baixo nível em C. Esse conhecimento é necessário para completo entendimento do texto principal, especialmente o **Capítulo 8**.
- **Apêndice C** descreve várias boas funções de dispersão que têm sido exaustivamente estudadas e são publicamente disponíveis.
- **Apêndice D** exibe as convenções utilizadas na escrita de identificadores que constituem os exemplos de programação deste livro.
- **Apêndice E** apresenta respostas e sugestões para os **Exercícios de Revisão** de cada capítulo deste volume.
- **Apêndice F** discute árvores rubro-negras e é encontrado exclusivamente online no site dedicado a este livro na internet (***www.ulysseso.com/ed2***).

Volume 1

O **Volume 1** desta obra é dividida em três partes principais e quatro apêndices. Essas partes são:

- **Parte 1** é dedicada a uma revisão da linguagem C, que será utilizada como ferramenta de implementação. Essa parte é constituída pelos **Capítulos** de **1** a **4**.
- **Parte 2**, que é composta pelos **Capítulos 5** e **6**, apresenta os conceitos fundamentais de estruturas de dados e as ferramentas matemáticas necessárias para analisá-las.
- **Parte 3** é constituída pelos demais capítulos e discute em detalhes todas as estruturas de dados básicas.

Capítulo 1 apresenta as construções básicas da linguagem C, mas não pretende ser um mini curso dela. Ao contrário, esse e os dois próximos capítulos constituem apenas uma revisão dessa linguagem que deverá ser útil como referência no acompanhamento dos tópicos centrais deste livro.

Capítulo 2 continua com a revisão da linguagem C apresentando tópicos mais avançados de programação nessa linguagem. É recomendável conhecer os tópicos discutidos resumidamente nesse capítulo, pois eles serão considerados pré-requisitos em capítulos posteriores.

Apesar de o **Capítulo 3** ser ainda considerado uma revisão da linguagem C, novamente, o leitor é aconselhado a examiná-lo, pois poderá desconhecer construções dessa linguagem que serão usadas subsequentemente.

A despeito de o **Capítulo 4** estar incluído na parte dedicada à revisão da linguagem C, dificilmente o conhecimento básico sobre programação dará completa ciência de todos os tópicos apresentados nesse capítulo, que lida com recursão e retrocesso. Portanto é recomendado que se inicie nesse capítulo o estudo do tema central.

Capítulo 5 enfoca em conceitos e definições fundamentais de algoritmos e estruturas de dados. Em especial, esse capítulo explora os conceitos de tipos de dados transparentes e opacos (TADs) e mostra como implementá-los em C.

O foco do **Capítulo 6** é a análise assintótica, que é uma ferramenta matemática fundamental para a avaliação de algoritmos. Esse capítulo tem um formato diferente dos demais porque é recheado com muitos exemplos que visam motivar os leitores menos afeitos a formalismos.

Capítulo 7 apresenta lista indexada como a primeira estrutura de dados fundamental. Além de definir conceitualmente essa estrutura de dados, esse capítulo também mostrará como essa abstração pode ser naturalmente implementada por meio de arrays estáticos.

Capítulo 8 lida com duas categorias de contêineres: pilhas e filas, que são estruturas de dados semelhantes às listas, mas que apresentam restrições de acesso. Ele mostra ainda como implementar essas estruturas de dados por meio de arrays estáticos.

No **Capítulo 9**, os conceitos de alocação estática e dinâmica de memória são discutidos. Na prática, ele ensina como alocar e liberar memória dinamicamente por meio de chamadas de funções da biblioteca padrão de C. Listas implementadas como arrays dinâmicos, com e sem ordenação, também são estudadas nesse capítulo.

Capítulo 10 justifica a necessidade de implementação de listas de forma encadeada e mostra por meio de inúmeras ilustrações e linhas de código como os diversos tipos de listas encadeadas podem ser implementados.

Capítulo 11 discute o conceito e aplicações de ponteiros para funções. Esse importante tópico de programação permite a implementação de algoritmos e estruturas de dados genéricos, que constituem o tema central desse capítulo.

Capítulo 12 é dedicado ao estudo introdutório das estruturas de dados hierárquicas mais importantes em programação: as árvores — especialmente as árvores binárias. Como texto introdutório, esse capítulo diverge dos demais, pois ele não descreve árvores como um tipo de dados convencional munido de operações essenciais e complementares. Esse capítulo terá prosseguimento mais aprofundado no **Volume 2**.

Os apêndices do **Volume 1** apresentam os conteúdos descritos a seguir:

- **Apêndice A** contém uma tabela com precedências e associatividades de todos os operadores da linguagem C e representa uma fonte de referência essencial para programadores dessa linguagem.
- **Apêndice B** expõe uma revisão dos tópicos de matemática utilizados nesse volume bem como demonstrações de teoremas enunciados nos **Capítulos 6** e **12**.
- **Apêndice C** exibe as convenções utilizadas na escrita de identificadores que constituem os exemplos de programação desse volume.
- **Apêndice D** apresenta respostas e sugestões para os **Exercícios de Revisão** de cada capítulo desse volume.

Exemplos de Programação

A maioria dos capítulo deste livro contém uma seção intitulada **Exemplos de Programação** na qual são demonstradas aplicações práticas dos conceitos apresentados no respectivo capítulo. Alguns poucos exemplos de programação incluídos nessas seções requerem conhecimentos específicos de uma determinada área de conhecimento e, quando isso ocorre, apresenta-se um preâmbulo cujo objetivo é prover definições e conceitos necessários para o completo entendimento do problema que será resolvido.

Exercícios

Ao final de cada capítulo são incluídos exercícios que servem para reforçar e ajudar a fixar o material exposto no respectivo capítulo. Eles estão divididos em dois grupos:

- **Exercícios de Revisão**. Esses exercícios objetivam a verificação de aprendizagem de cada capítulo. Resolvendo os exercícios de revisão, é possível fazer uma autoavaliação e, então, rever aquilo que ficou mal entendido. As respostas para a maioria desses exercícios são encontradas no **Apêndice D** do **Volume 1** e no **Apêndice E** do **Volume 2**. Exercícios de revisão não requerem a escrita de programas, embora algumas questões possam ser respondidas prontamente desse modo.
- **Exercícios de Programação**. O objetivo desses exercícios é estritamente prático e eles requerem o uso de um computador e um ambiente de desenvolvimento adequados. Deve-se salientar que não há solução única para cada problema e, além disso, pode-se constatar facilmente se uma proposta de solução é correta executando-se o programa (i.e., a solução) e verificando-se se os resultados são compatíveis com o enunciado do respectivo problema.

Organizadores Prévios

No início de cada capítulo é incluído um quadro contendo as competências que o leitor deverá adquirir após o estudo desse capítulo. Esses quadros são denominados **organizadores prévios** e foram criados pelo psicólogo educacional americano David Ausubel (v. **Bibliografia**). Segundo seu criado, um organizador prévio facilita o entendimento do novo material que será apresentado no capítulo.

Material Complementar

Este livro possui um site dedicado a si na internet que pode ser acessado no endereço: ***www.ulysseso.com/ed2***. Nesse site encontram-se os códigos-fonte de todos os exemplos apresentados no texto, além de outros programas não inseridos nele. Esse material é classificado de acordo com os capítulos correspondentes no livro

Recursos Utilizados

Este livro foi diagramado pelo autor usando **Adobe InDesign™**. A maioria das figuras foi criada com **Adobe Illustrator™** e grande parte das fórmulas matemáticas foi concebida com **MathType™**.

Agradecimentos

Muitos exemplos e exercícios propostos neste livro foram inventados pelo próprio autor, mas muitos outros são oriundos de textos referenciados na **Bibliografia** ao final do livro. Outros tantos foram coletados ao longo da extensa carreira do autor e, se não foram devidamente reconhecidos na bibliografia, o autor lamenta tal omissão não intencional.

Este livro é oriundo de notas de aula continuamente refinadas durante vários semestres de ensino das disciplinas **Estruturas de Dados** e **Ordenação e Recuperação de Dados** nos cursos de **Ciência da Computação e de Engenharia da Computação** da **Universidade Federal da Paraíba**. Desse modo, o autor gostaria de agradecer a alunos e monitores do **Departamento de Informática da UFPB** que indicaram falhas em versões anteriores do texto e apresentaram sugestões para sua melhoria.

Em especial, gostaria de agradecer ao amigo arquiteto a quem sou penhoradamente grato Amaro Muniz pelas inúmeras sugestões para melhoria do design gráfico.

Ulysses de Oliveira

Janeiro de 2019