capítulo 15

Aplicações

> *As ideias são como as nozes, e, até hoje, não descobri melhor processo para saber o que há dentro de umas e outras, senão quebrá-las.*
>
> Machado de Assis

A informação é a matéria-prima da qual se extrai o conhecimento, por meio do qual adquirimos sabedoria. Por isso, podemos afirmar que, enquanto o conhecimento e a sabedoria nos permitem ter novas ideias, o experimento é a forma mais indicada de testá-las. Experimentar é "quebrar as nozes".

OBJETIVOS DO CAPÍTULO

Ao final da leitura deste capítulo, o leitor deverá ser capaz de:

- entender a importância do conhecimento obtido por meio dos conceitos explicitados nos capítulos anteriores;
- projetar e implementar algoritmos e programas, tendo esses conceitos como base, para solucionar problemas computacionais;
- pensar e criar aplicações com base nos exemplos e exercícios sugeridos em cada capítulo.

Como você sabe, muitos dos problemas computacionais resolvidos por meio da construção de algoritmos/programas resultam de ideias ou mesmo de abstrações de situações do mundo real. Com base em ideias, nós construímos modelos mentais lógicos para, em seguida, aplicando os conceitos e teorias sobre construções lógicas de algoritmos, implementar modelos de algoritmos que serão executados pelo computador para solucionar problemas.

Até este momento, diversos assuntos novos – ao menos para alguns leitores – foram apresentados e estudados, e isso nos possibilitou conhecer, ou até mesmo reforçar, um número razoável de conceitos e teorias sobre a construção de algoritmos/programas. Agora só nos resta experimentar isso tudo, abstraindo, projetando, desenvolvendo e implementando sistemas, ou *aplicações*, como muitos profissionais denominam os sistemas.

Como afirma o professor José Augusto Fabri, da Universidade Federal Tecnológica do paraná, "O sistema está ligado a alguma coisa nova, a uma nova teoria, a uma invenção, a um novo paradigma".

Para começar

Ao longo de seus estudos, principalmente nas disciplinas Algoritmos e Programação, quantas vezes você se debruçou sobre os problemas sugeridos e tentou encontrar soluções para eles?

Certamente, inúmeras vezes e sempre recorrendo a teorias, conceitos e exemplos que adquiriu – e vem adquirindo – nesses anos de estudos.

Na construção de algoritmos e programas de computador, quanto mais aplicamos aquilo que aprendemos, mais sedimentados ficam os conceitos, as sintaxes, as estruturas lógicas, os modelos etc. Portanto, o desenvolvimento de novos projetos, sistemas ou aplicações nos ajuda a entender e a solucionar problemas, possibilitando a integração da teoria com a prática.

Vamos experimentar?

Conhecendo a teoria para programar

Até o momento, você estudou vários assuntos que foram distribuídos entre os seguintes capítulos:

1. Tipos abstratos de dados (TAD).
2. Mapa de memória de um processo.
3. Recursão.
4. Métodos de busca.
5. Métodos de ordenação.
6. Alocação dinâmica.
7. Listas ligadas lineares.
8. Outros tipos de lista.
9. Filas.
10. Pilhas
11. Árvores
12. Árvores *N*-árias.
13. Árvores balanceadas.
14. Grafos.

Os conteúdos desses capítulos serão fundamentais durante a construção de algoritmos e programas para solucionar problemas computacionais com maior nível de complexidade lógica em relação àqueles que você estava acostumado a desenvolver.

Na construção de programas de computadores, recomendamos recapitular o conteúdo do "Para começar" do último capítulo do livro *Algoritmos e programação de computadores* (PIVA, D. *et al.*, 2012). Nele estão pontuadas algumas propriedades de um bom sistema, também válidas para um programa de computador, que reproduzimos e enumeramos a seguir:

- ***ser* eficiente e eficaz** - dar respostas ao usuário em tempo adequado, executando de forma correta e lógica suas instruções;

- **ser portável** - poder ser executado em diversas plataformas computacionais, por computadores que utilizam sistemas operacionais diferentes;
- **ser seguro e tolerante a falhas** - não permitir sua utilização por qualquer usuário, nem deixar as conhecidas *backdoor*, que permitem a entrada e a instalação de vírus;
- **confiável e disponível** - processar dados e informações de forma consistente; quando em execução, estar disponível o tempo todo que o computador permanecer ligado, seja onde estiver sendo executado;
- **modular e escalável** - permitir modificações sempre que ocorrerem mudanças ambientais, por meio da incorporação de novos processos ou funções, sem afetar seu desempenho.

Em "Vamos programar", a seguir, sugerimos novos exemplos de aplicações que abrangem os conceitos apresentados em cada capítulo. Você, certamente, poderá desenvolver alguns deles ou, se preferir, todos. De qualquer forma, esperamos que você os desenvolva considerando – e aplicando – algumas dessas características – ou todas elas. Para tanto, é recomendável a escolha de uma boa linguagem de programação.

Opa! E quais são as características de uma boa linguagem de programação?

Existem inúmeras, porém, para propósitos gerais, podemos considerar que uma boa linguagem de programação é aquela que:

- possui extensiva capacidade de gerenciamento de arquivos e bancos de dados com rapidez, integridade e segurança;
- trata arquivos independentemente da quantidade de registros, sem perder *performance*;
- tem compatibilidade com outras bases de dados;
- tem portabilidade, ou seja, pode ser executada em qualquer plataforma de máquina, sem necessidade de modificações ou recompilações;
- pode ser utilizada em vários ambientes computacionais e por diferentes IDE (*Integrated Development Environment*);
- transmite confiabilidade e mantém sua continuidade (disponibilidade de novas versões);
- proporciona, por intermédio do fabricante, bom suporte técnico, e dispõe de vasta e boa documentação técnica e literária nos mais diversos suportes midiáticos;
- tem uma interface gráfica amigável (GUI).

Enfim, uma boa linguagem de programação é aquela que satisfaz melhor as necessidades do programador de computador e está adequada para a solução dos problemas que se apresentam.

Por fim, o mais importante é pensar em todos os problemas sugeridos como um novo desafio para sua criatividade.

Considerando que são os desafios que nos movem, vamos em frente!

Vamos programar

Um dos principais objetivos da construção de algoritmos e da programação de computadores é desenvolver a capacidade de raciocínio lógico, abstração, interpretação e análise crítica de dados e informações, criando e implementando soluções computacionais para problemas previamente identificados.

Para a implementação dos algoritmos exemplificados em cada capítulo, utilizamos três diferentes linguagens de programação. Todavia, a construção de modelos lógicos para a solução de problemas computacionais independe da linguagem de programação utilizada, pois, para escrever algoritmos, basta dispor de lápis e papel.

Sugerimos, a seguir, alguns problemas do cotidiano, propondo algumas estratégias para seu desenvolvimento. Fica a seu critério a escolha da linguagem de programação para sua implementação. Todavia, a escolha do TAD ou estrutura de dados, bem como da estrutura lógica mais adequada para a solução do problema, dependerá dos conhecimentos que você obteve nos capítulos anteriores. Vamos lá!

1. Uma estrutura de dados muito utilizada na área da ciência da computação é a fila. Dentre as muitas aplicações dos conceitos e teorias sobre filas, destacam-se aqueles utilizados na construção de sistemas operacionais. Como sabemos, o sistema operacional tem várias funções, que, *grosso modo*, podem ser divididas em duas categorias: máquina virtual e gerenciador de recursos. Enquanto máquina virtual, ele fornece a base sobre a qual os programas de aplicação são escritos e executados, e, como gerenciador de recursos, sua função primordial é controlar ou gerenciar a utilização de todos os recursos fornecidos pelo *hardware* do computador e, principalmente, sua distribuição entre os diversos programas (processos) que competem (disputam) por ele. Como você pode perceber, estamos falando de controlar prioridades.

Então, sugerimos que você elabore uma aplicação que controle as tarefas (processos) que esperam por um recurso de *hardware* escasso, que são as impressoras. Isso mesmo: esse aplicativo deve controlar os pedidos de uso de impressora por processos, na ordem em que forem colocados em execução, ou seja, obedecendo a uma ordem prioritária.

Você pode pensar em implementar o seguinte:

a. listar todos os processos que estão aguardando liberação da impressora;
b. informar o tempo de espera de cada processo na fila de impressão;
c. permitir mudar a prioridade de execução de um processo;
d. permitir retirar (cancelar) um processo da fila.

Se abstrair um pouco mais sobre esse exercício, talvez você descubra mais alguma funcionalidade para esse aplicativo.

2. Uma empresa agropecuária dispõe, entre outros produtos que vende, daqueles que têm prazo de validade. Se esses prazos não forem controlados, a empresa poderá perder muitos produtos com prazo de validade vencido ou até mesmo sofrer processos judiciais, caso venda um produto nessa condição. Portanto, um aplicativo que controla prazos de validade é estratégico nesse tipo de negócio, assim como em outros similares.

Gostaria de ajudar essa empresa a resolver tal problema?

Se a resposta for SIM, desenvolva um aplicativo para controlar o prazo de validade de produtos que entram no estoque de produtos acabados de uma empresa.

Sugestão:

Imagine que a empresa precisa que, além de controlar o prazo de validade, esse aplicativo forneça informação importante, como: quantidade de produtos em estoque; valor total do estoque de produtos com prazo de validade a expirar em determinado tempo (daqui a 7 dias, um mês ou outro qualquer, informado previamente). E então? Você imaginou que terá de trabalhar com uma estrutura de dados heterogênea (registro) ligada a outras estruturas homogêneas?

Tem mais: a empresa também poderia pedir que essa aplicação permitisse visualizar os produtos em estoque por descrição, por valor e, obviamente, por prazo de validade. Nesses casos, você estaria aplicando os métodos de ordenação, aquilo que aprendeu no Capítulo 5 deste livro.

3. Uma cidade costuma ser dividida em regiões considerando-se os pontos cardeais, ou seja, norte, sul, leste e oeste. Olhe o mapa da cidade onde você mora e procure identificar essa divisão. Em seguida, localize as instituições públicas, ou mesmo privadas (entidades), que são de interesse da população que fazem parte de cada região: hospitais, delegacias, corpo de bombeiros, escolas, creches, etc.

Agora, que tal construir uma aplicação que contemple, por região, uma relação de entidades, com seus endereços e telefones de emergência?

Vamos lá!

Sugestões:

a. possibilitar inserção e exclusão de entidades;
b. possibilitar a alteração de dados sobre cada entidade;
c. possibilitar a busca de informações sobre essas entidades;
d. possibilitar a ordenação dos dados sobre as entidades, por exemplo, por bairro, logradouro, nome, telefone, etc.

4. A principal atividade de um carteiro é entregar correspondências postais. Você pode notar que todos eles carregam uma sacola com pilhas de correspondências. Quando um carteiro chega a determinado endereço, ele retira e entrega a correspondência que está no topo da pilha. Considerando isso, elabore um programa para simular os processos executados por um carteiro. Ao pensar esse programa, você certamente estará aplicando os conceitos sobre alocação dinâmica de memória (Capítulo 6) e pilhas (Capítulo 10). Lembre-se de que a pilha é uma estrutura de dados com a característica de acessibilidade somente ao elemento que está no topo.

Vamos lá!

Você pode pensar nos seguintes processos e torná-los computacionais:

a. inserir correspondência na sacola (empilhar);
b. remover correspondência da sacola (desempilhar);
c. verificar se a sacola está cheia ou vazia;
d. verificar a próxima correspondência a ser entregue;
e. informar quantas correspondências existem na sacola;

Feito isso, que tal ajudar o carteiro a traçar o caminho mínimo para a entrega das correspondências – afinal, todas as correspondências têm um endereço/destino. Lembra-se dos conceitos relativos a caminhamento mínimo em um grafo? Este é o momento de aplicá-lo.

5. Uma aplicação interessante para o exercício dos conceitos relacionados a pilhas é a compilação de uma expressão aritmética. Quando você estudou operadores aritméticos, aprendeu sobre a prioridade desses operadores dentro de uma expressão, não é mesmo? Então, que tal escrever um programa que compile uma expressão aritmética, como:

```
a - b / e + c * d
```

Critérios:

a) os operadores da expressão aritmética são representados pelos caracteres: *, /, + e -;
b) os operandos são representados por letras de *a* até *j*;
c) os resultados das operações são representados (armazenados) por letras, de trás para a frente, iniciando-se em *z*;

A tabela a seguir representa o resultado da compilação da expressão modelo.

Operador	Operando 1	Operando 2	Resultado
/	*b*	*e*	*Z*
*	*c*	*d*	*Y*
-	*a*	*z*	*x*
+	*x*	*y*	*v*

Procedimentos:

a) o programa deve receber os caracteres e as letras especificados na tabela;
b) o programa deve validar esses caracteres, ou seja, se o caractere for um espaço em branco, você deverá ignorá-los; todavia, se não for nenhum dos caracteres da tabela, envie mensagem de erro e encerre o programa;
c) se o caractere for um operando, coloque-o na pilha de operandos;
d) se o caractere for um operador, então compare sua precedência (prioridade) com o operador que está no topo da pilha de operadores;
e) se o operador em questão (atual) tem prioridade maior do que aquele que está no topo da pilha, ou a pilha for NULL, então o operador atual deve ir para o topo da pilha;
f) se o operador atual tem a mesma prioridade ou prioridade menor, então o operador do topo da pilha é o próximo a ser avaliado, da seguinte forma: retira-se o operador da pilha de operadores, com um par de operandos, que está na pilha de operandos, e monta-se uma nova linha na tabela de saída. O caractere escolhido para armazenar o resultado da operação dessa linha deve ser colocado na pilha de operandos;
g) Após isso, o operador atual deve ser novamente comparado com aquele que ficou no topo da pilha de operadores. Repita esse processo até que a pilha de operadores seja NULL.

Após desenvolver e testar essa aplicação, que tal você pensar em modificá-la para implementar parênteses nas expressões aritméticas?

Lembre-se de que, em expressões, os parênteses são utilizados para modificar a precedência (ou prioridade) dos operadores. Temos aí um novo desafio para você. Aceita?

Bem, até aqui você avançou muito nas aplicações sobre estrutura de dados ou TAD, listas, filas, pilhas. Portanto, as próximas sugestões de aplicações relacionam-se com árvores, que, conforme estudamos nos Capítulos 11, 12 e 13, são estruturas de dados baseadas em listas encadeadas que têm um nó superior (também chamado de raiz) que aponta para outros nós (chamados de nós filhos), que também podem ser pais de outros nós.

Então, vamos sugerir uma aplicação para esses conceitos.

6. Você já ouvir falar em árvore genealógica de uma família?

Conceitualmente: "Uma árvore genealógica é um histórico de certa parte dos antepassados de um indivíduo ou família. Mais especificamente, trata-se de uma representação gráfica genealógica para mostrar as conexões familiares entre indivíduos, trazendo seus nomes e, algumas vezes, datas e lugares de nascimento, casamento, fotos e falecimento." (Wikipedia)

Que tal construir a árvore genealógica da sua família iniciando-a pelo seu bisavô. Essa é uma boa aplicação dos conceitos relativos a árvores e a listas, pois você poderá armazenar dados sobre as pessoas da família e, portanto, aplicar os conceitos estudados nos Capítulos 7 e 8.

Após montar a árvore genealógica de sua família, desenvolva uma aplicação para implementá-la computacionalmente.

Você pode pensar nos seguintes processos e torná-los computacionais:

a. inserir um novo nó filho nessa árvore quando alguém nascer;
b. excluir um nó dessa árvore quando alguém falecer;
c. inserir nós quando ocorrer um casamento;
d. efetuar buscas na árvore a partir de determinado nó (família ou pessoas).

Em todos os casos, você vai aplicar os conceitos referentes à inserção de nós à esquerda e à direita, e também ao percurso em uma árvore.

Você terá que rotular os nós e criar listas ligadas com mais informações sobre cada família ou mesmo integrantes da família.

E então, está pronto para pensar esse problema?

Nossas próximas sugestões de aplicações referem-se ao assunto *grafos*.

Já vimos que grafos armazenam e facilitam a manipulação de estruturas de dados com características de ligações entre pontos (nós ou vértices).

Quando elaboramos algoritmos aplicando conceitos sobre grafos, certamente utilizamos estruturas de dados, como vetores, listas encadeadas ou matrizes. O uso de listas geralmente proporciona acesso mais rápido aos dados da estrutura; todavia, na maioria dos casos, é mais fácil utilizar vetores. Podemos verificar isso na literatura, cuja maioria dos exemplos utiliza-se de vetores.

Vamos para mais uma aplicação?

7. Imagine que a prefeitura de sua cidade está projetando um novo bairro, em um local de relevo bem plano e que permite ótimo planejamento urbano, principalmente das quadras e ruas. Entretanto, um dos desafios da prefeitura é planejar esse bairro de forma que ele tenha o maior número possível de ruas com sentido único.

Sugestões:

a. procure saber quais ruas podem ter mão-única e para que sentido ela apontará, mantendo todas as ruas alcançáveis de qualquer outro ponto na cidade;
b. elabore e implemente um grafo orientado e rotulado, de forma que você possa, partindo de determinado ponto, chegar a um destino através de um caminho mínimo e um caminho máximo.

8. Um dos problemas enfrentados por uma pessoa numa grande cidade é localizar um restaurante próximo do ponto onde ela se encontra, para almoçar ou jantar. Atualmente, esse tipo de aplicação está disponível em dispositivos móveis (plataforma *Mobile*), que combinam informações de GPS com dados e informações sobre esses estabelecimentos, contidos em guias turísticos ou mesmo na lista telefônica.

Sugerimos que você construa uma aplicação que auxilie uma pessoa a encontrar um restaurante próximo do local onde ela está.

Sugestão:

Delimite uma área de cobertura (abrangência) para seu aplicativo. Considerando essa área, mapeie nela os principais restaurantes em funcionamento e estabeleça a distância entre eles. Com essas informações, você já pode pensar em construir um grafo e sua matriz de adjacências ou lista de incidências. Agora deve preocupar-se em escolher o método de caminhamento nesse grafo, a partir de um ponto (nó) inicial.

Ah! lembre-se de coletar e registrar outras informações sobre cada estabelecimento, tais como: telefone, e-mail, tipos de cartões aceitos, horários de funcionamento, etc. Armazene tudo isso em estruturas de dados, para que possam ser acessadas pelos usuários do aplicativo.

Outra sugestão de importante aplicação da teoria de grafos é encontrada na biologia e está relacionada com o sequenciamento da cadeia de

DNA. A cadeia de DNA é composta pelas bases A (adenina), C (citosina), G (guanina) e T (timina). Uma das aplicações de computadores nessa área (bioinformática) é o estabelecimento de medidas para similaridade entre sequências biológicas.

9. Considere como exemplos as sequências CATT e GAT. O possível alinhamento dessas sequências é dado a seguir.

```
G A - T T - C A - T
```

(-) representa uma falha.

Cada membro de uma sequência é emparelhado com um membro de outra sequência ou com uma falha, sendo atribuído um peso (valor) a cada emparelhamento.

(1) representa o emparelhamento de dois membros iguais.

(-1) representa o emparelhamento de dois membros diferentes.

(-2) representa o emparelhamento de um membro e uma falha.

A "qualidade" do alinhamento é dada pela soma dos valores (pesos) de todos os seus emparelhamentos.

No exemplo acima, o valor será: -2-1-2-2 + 1 = **-6**.

Agora, no alinhamento a seguir:

```
G A T T C A T -
```

temos os valores: -1 + 1 + 1-2 = **-1.**

Portanto, conforme esse critério, o segundo sequenciamento é de melhor qualidade. A evolução das técnicas de clonagem de moléculas, aliada aos avanços das tecnologias da informação e à evolução das técnicas de armazenamento e recuperação de dados e informações (bancos de dados), a partir de grandes bases de dados, vem exigindo cada vez mais aplicações (programas) de métodos de comparação de sequências biológicas.

Esse problema, e outros, pode ser resolvido aplicando-se os conceitos sobre grafos. Lembra-se da matriz de adjacências? Então, você pode construir um dígrafo correspondente ao problema de alinhamento das sequências acima, montar uma matriz de adjacências e, a partir daí, calcular o melhor alinhamento (o de valor mais alto) das sequências. Para isso, você deve achar o caminho mais longo (máximo), em que seu comprimento seja a soma dos comprimentos dos arcos no caminho.

Tente fazer isso!

Para saber mais...

Dos salões paroquiais às salas de aulas, destas para as salas de reuniões nas empresas, das salas de reuniões aos encontros sociais nos fins de semana, das redes privativas às redes sociais, sem dúvida nenhuma uma revolução está acontecendo, apoiada por tecnologias da informação. Temos hoje o conceito de "um para um" indo em direção ao de "um para muitos", em um ambiente social, público e compartilhado. Isso é, bem superficialmente, o conceito das redes sociais, de que você certamente faz parte.

Atualmente, não somente as pessoas, mas também as empresas estão investindo em modelos de sistemas que permitam entender melhor os relacionamentos que ocorrem nessas redes e, principalmente, as informações que circulam nelas. Investir em aplicativos que auxiliem na análise de redes sociais deixou de ser uma necessidade dos departamentos de Marketing e Vendas para se tornar um assunto relacionado com estratégias empresariais. Isso ficou nítido com a criação do cargo de Analista de Redes Sociais.

Nesse contexto, podemos inferir sobre a importância dos assuntos abordados neste livro, notadamente aqueles relativos a árvores e grafos, e como estes podem ser utilizados para fazer a análise e o estudo das relações em redes, auxiliando pessoas e empresas a tirar proveito delas.

Para saber mais sobre isso, sugerimos que você acesse o link http://labspace.open.ac.uk/course/view.php?id=4951&topic=all, onde você vai encontrar textos sobre redes sociais e sobre a teoria de grafos aplicada às redes sociais. Obviamente, centenas de outros materiais sobre isso estão disponíveis em diversos suportes midiáticos, entretanto, este é de fácil leitura e entendimento.

Vamos ler?

Navegar é preciso

Percebeu quantas aplicações podemos pensar e solucionar usando algoritmos e, consequentemente, a programação de computadores?

Isso mesmo! Quanto mais abstraímos o mundo em que vivemos, seja muito próximo ou mesmo distante de nós, mais descobrimos problemas que podem ser solucionados usando o computador.

Você já deve ter ouvido falar, lido sobre o assunto ou até mesmo participado das Maratonas de Programação. A cada edição dessa competição,

seja em nível regional, nacional, seja em nível internacional, novos desafios (problemas computacionais) são lançados e as equipes participantes devem construir programas para vencê-los/solucioná-los. Portanto, sugerimos que você acesse o link http://uva.onlinejudge.org/, em que poderá ter acesso a vários problemas dados nessas maratonas.

No link http://maratona.ime.usp.br/preparacao13.html você encontrará outros sublinks importantes com problemas sugeridos nas maratonas da Sociedade Brasileira de Computação e também na Olimpíada Brasileira de Informática.

Vale a pena navegar nessa página!

A área de computação vem, ao longo da sua evolução, criando termos e terminologias que, muitas vezes, guardam em si ou provocam o desenvolvimento de aplicações complexas. Atualmente, ouve-se falar muito em *big data*. O que seria isso?

Em tecnologia da informação, *big data* é um conceito no qual os focos são o grande armazenamento de dados e a maior velocidade na recuperação desses dados. Podemos dizer que o *big data* baseia-se em 5V: *velocidade*, *volume*, *variedade*, *veracidade* e *valor*.

Se antes as empresas pecavam pela falta de dados e informações, hoje pecam pelo excesso, ou seja, potentes *data warehouse* armazenam *exabytes* de dados e informações, e sabe-se que o conhecimento tão necessário para a condução dos negócios das empresas está contido nas relações entre dados e informações. Entretanto, o maior problema enfrentado pelas empresas atualmente é como extrair o conhecimento de um *big data*.

Se você quer saber um pouco mais sobre como os conceitos de estruturas de dados vêm sendo aplicados nesse assunto ou mesmo em bioinformática, acesse os links a seguir:

http://www.inf.ufpr.br/jbdorr/texto-quali-jbdorr.pdf

http://www.bibliotecadigital.unicamp.br/document/?code=000201497

Para saber mais sobre *big data*, acesse:

https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/voce_realmente_sabe_o_que_e_big_data?lang=en

No link http://mitpress.mit.edu/sites/default/files/titles/content/9780262033848_Solutions_to_Exercises_and_Problems.pdf você encontrará solução para exercícios contidos no livro *Algoritmos: teoria e prática* (Cormen, 2012).

Glossário

Bioinformática: área da ciência da computação que visa analisar dados e resolver problemas de aplicações biológicas por meio do desenvolvimento de ferramentas. O avanço tecnológico dos computadores possibilitou à bioinformática desenvolver softwares para processar grande volume de dados gerados pelos projetos da biologia.

(http://www.ic.unicamp.br/~zanoni/orientacoes/mestrado/mangelica/PropostaMariaAngelica.pdf)

GPS (G*lobal Positioning System*): sistema de navegação por satélite que fornece a um aparelho receptor móvel sua posição, assim como informação horária, sob todas as condições atmosféricas, a qualquer momento e em qualquer lugar na Terra, desde que o receptor se encontre no campo de visão de quatro satélites GPS. (Wikipedia)

IDE (*Integrated Development Environment)*: programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar esse processo. Geralmente, os IDE facilitam a técnica de RAD (*Rapid Application Development*, ou Desenvolvimento Rápido de Aplicativos), que visa a maior produtividade dos desenvolvedores.Um IDE pode incluir vários utilitários, todavia, todos implementam pelo menos editor de texto, compilador, *link-editor* ou montador, depurador e *run-time* (executor). (Wikipedia)

Mídias sociais: sistemas online que permitem a interação a partir da criação e do compartilhamento de conteúdo nos mais diversos formatos. Esses sistemas possibilitam a publicação de conteúdo por qualquer pessoa/empresa, baixando para quase zero os custos com produção e distribuição de algum material de divulgação.(Wikipedia)

Processo: sequência sistemática de operações que visam a produzir um resultado específico. Em computação, é um conjunto de instruções sequenciado que pode ser utilizado em um ou mais pontos, em um ou mais programas de computador, e que geralmente tem um ou mais parâmetros de entrada, produzindo um ou mais parâmetros de saída.

Sistema: conjunto de elementos que se conectam, harmonicamente, com o objetivo de formar um todo unificado. (Ludwig von Bertalanffy)

Software aplicativo (**aplicativo ou aplicação**): programa de computador que tem por objetivo ajudar seu usuário a desempenhar uma tarefa específica, em geral ligada ao processamento de dados. (Wikipedia)

Referências bibliográficas

BATES, B.; SIERRA, K. *Use a cabeça: Java*. Rio de Janeiro: Alta Books, 2007.

CORMEN, T. H. et al. *Algoritmos: teoria e prática*. Rio de Janeiro: Elsevier, 2012.

LAFORE, R. *Estrutura de dados & algoritmos em Java*. Trad. Eveline Vieira Machado. Rio de Janeiro: Ciência Moderna, 2004.

LISKOV, B.; ZILLES, S. Programmingwith Abstract Data Types. *ACM SIGPLAN Notices*, ano 9 (4) 50-59, 1974.

MOKARZEL, F. C.; SOMA., N. Y. *Introdução à ciência da computação*. Rio de Janeiro: Elsevier, 2008.

PIVA, D. *et al. Algoritmos e programação de computadores*. Rio de Janeiro: Campus, 2012.

SCHILDT, H. C. *completo e total*. 3. ed. São Paulo: Makron Books, 1996.

TENENBAUM, A. M. *Estruturas de dados usando C*. São Paulo: Makron Books, 1995.

VELOSO, P. et al. *Estruturas de dados*. Rio de Janeiro: Campus, 1983.

WIRTH, N. *Algoritmos e estruturas de dados*. Rio de Janeiro: Prentice-Hall, 1989.

O QUE VEM DEPOIS

Ufa! Chegamos ao final do nosso passeio pelas estruturas de dados.

Se você teve a oportunidade de ler nosso livro *Algoritmos e programação de computadores*, lançado em 2012 pela Editora Elsevier, então, com mais este livro, você certamente avançou muito e agregou conhecimentos sobre o assunto *algoritmos*. Diante disso, acreditamos que você está mais seguro e tem mais habilidades para construir aplicações com maior nível de complexidade de algoritmos. Aliás, a complexidade de algoritmos é o próximo assunto que sugerimos que você pesquise e estude, pois, dessa forma, avançará ainda mais na programação de computadores, encontrando soluções para problemas cada vez mais intricados.

O poeta francês Jean Cocteau escreveu a seguinte frase: "*Não sabendo que era impossível, foi lá e fez*". A um hábil construtor de algoritmos e programas, caberia essa frase com uma pequena modificação: "Sabendo que era impossível, foi lá e fez". Assim deve agir um bom programador de computadores.

Sucesso!