Introdução à Ciência da Computação

Hendrik Teixeira Macedo

São Cristóvão/SE
2009

Introdução à Ciência da Computação

Elaboração de Conteúdo
Hendrik Teixeira Macedo

Capa
Hermeson Alves de Menezes

Reimpressão

Ficha catalográfica produzida pela Biblioteca Central
Universidade Federal de Sergipe

N972i Macedo, Hendrik Teixeira
Introdução à ciência da computação / Hendrik Teixeira Macedo -- São Cristóvão: Universidade Federal de Sergipe, CESAD, 2009.

1. Ciência da computação. 2. Tecnologia da computação. 3. Informática. I. Título.

CDU 004

UNIVERSIDADE FEDERAL DE SERGIPE
Cidade Universitária Prof. "José Aloísio de Campos"
Av. Marechal Rondon, s/n - Jardim Rosa Elze
CEP 49100-000 - São Cristóvão - SE
Fone(79) 2105 - 6600 - Fax(79) 2105- 6474

Sumário

I

Programação Básica

1 Introdução à Programação

Onde são apresentados os fundamentos da computação e o porquê da importância de seu estudo para o aprendizado da programação de computadores

Pré-requisito(s):	Objetivos (ao final você deverá ser capaz de):
• Conhecimento de matemática do ensino médio • Manuseio básico do computador • Saber navegar na Internet • Muita disposição para aprender sobre computação e lógica de programação	• Identificar e distinguir elementos de um sistema computacional • Compreender a relação entre programação e organização de um sistema computacional • Identificar problemas resolvíveis via programação • Elaborar um algoritmo para solução de um problema • Compilar e executar programas em Java

Existem pelo menos dois tipos de pessoas que lidam com computadores em seu dia a dia. O primeiro tipo consiste de pessoas que fazem uso dos diversos aplicativos disponíveis que facilitam o trabalho ou propiciam algum tipo de lazer. Essas pessoas também se beneficiam bastante do conhecimento em larga escala disponível na Internet e cujo acesso é facilitado também pelo uso desses aplicativos. O outro tipo de pessoa é o profissional da área, que também é usuário, mas cujo conhecimento sobre os fundamentos da computação extrapola o do usuário comum. Essas pessoas possuem grande conhecimento sobre dispositivos físicos computacionais e ferramentas lógicas para criação da diversidade de aplicativos que o outro tipo de pessoa tanto se beneficia. Este capítulo apresenta com mais detalhes o conceito de sistema computacional e como seus elementos fundamentais estão interligados para propiciar o processamento dos dados que são fornecidos constantemente ao computador pelos usuários e produzir informação de

valor útil como resultado. O capítulo também introduz o conceito de programação lógica de computadores e como um problema pode ser encarado como um problema de solução computacional; solução esta, que é chamada de algoritmo. Em seguida, é mostrado como esses algoritmos podem ser construídos e executados em um computador através do uso de uma linguagem de programação. Finalmente, uma linguagem de programação bastante rica e popularizada, chamada Java, é introduzida juntamente com o procedimento básico para criação de programas e execução dos mesmos.

1.1 Fundamentos da Computação

Informática é a ciência que estuda o **tratamento automático da informação**. Dentre as principais funções da informática (ou da computação) estão:

- Desenvolvimento de novas máquinas;
- Automatização de processos; e
- Melhoria de aplicações existentes.

O **computador** funciona como elemento central na computação. É uma máquina constituída por componentes e circuitos eletrônicos, capaz de receber, armazenar e processar dados e transmitir informações.

A atividade de **processamento de dados** consiste em transformar sinais brutos e sem significado individual (dados) em informação com algum valor real para o usuário final através de diferentes formas de manipulação (figura 1.1). Uma lista de valores decimais, representativos de índices pluviométricos e de temperatura, gerados automaticamente e enviados a uma estação na Terra por um satélite em órbita, por exemplo, não possui em princípio valor algum para um usuário padrão. Entretanto, se submetidos a um processamento adequado, esses valores podem ser transformados em informações úteis do tipo "Chuva forte em todo o litoral sergipano nesta terça-feira" ou "Mínima de 24º e máxima de 28º em Arauá".

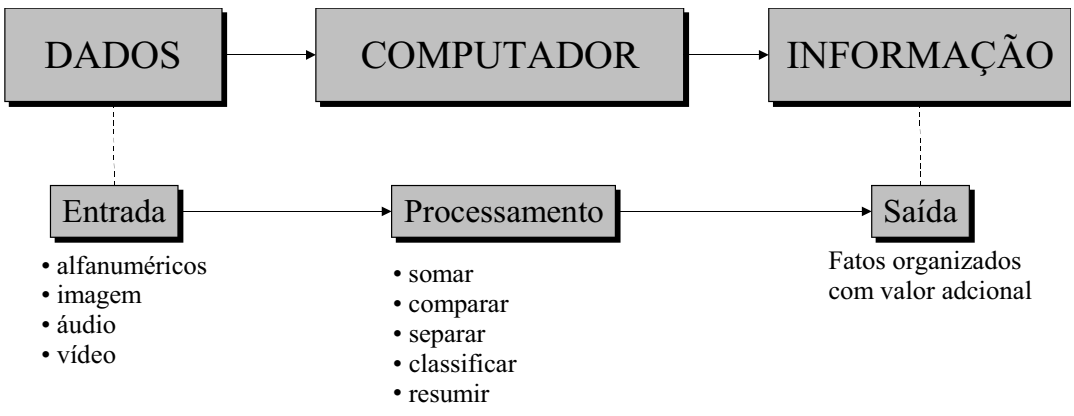

Figura 1.1 – O computador é responsável pelo processamento dos dados fornecidos

através de algum mecanismo de entrada e gera a informação adequada através de algum mecanismo de saída.

1.1.1 Sistema Computacional

Essa visão do computador como uma "caixa-preta" com a função de pura e simplesmente processar dados fornecidos e gerar algum tipo de informação, é considerada hoje um pouco ultrapassada. Falamos atualmente de **Sistema Computacional**.

Um Sistema Computacional é composto de uma tríade: ***Hardware*, *Software* e *Peopleware***. O *Hardware* representa o conjunto de todos os componentes físicos do sistema. O *Software* constitui o conjunto de componentes lógicos (funcionais) do sistema. Um software é um conjunto de instruções com finalidade de resolver alguma tarefa. Assim o termo *Software* se refere a qualquer conjunto de programas, aplicativos e utilitários que executam no computador. São exemplos de software os sistemas operacionais (ex: Windows, Linux), editores de texto (ex: Word), planilhas (ex: Excel), etc. Finalmente, o termo *Peopleware* representa o conjunto de pessoas envolvidas no sistema computacional. São as pessoas que operam as máquinas, que fazem uso de suas funcionalidades ou que criam os softwares que serão utilizados. Esses três elementos trabalham utilizando dados. Aqueles mesmos que são processados e se transformam em informação. A figura 1.2 ilustra um Sistema Computacional.

Para o bom entendimento dos princípios da programação de computadores que nos permite a criação dos diversos softwares que nos ajudam nas tarefas do dia a dia, é imprescindível conhecer bem o funcionamento do Sistema Computacional. Em particular, é extremamente importante conhecer o funcionamento de alguns dos elementos de hardware e como se relacionam para permitir que instruções possam ser executadas sobre um conjunto de dados.

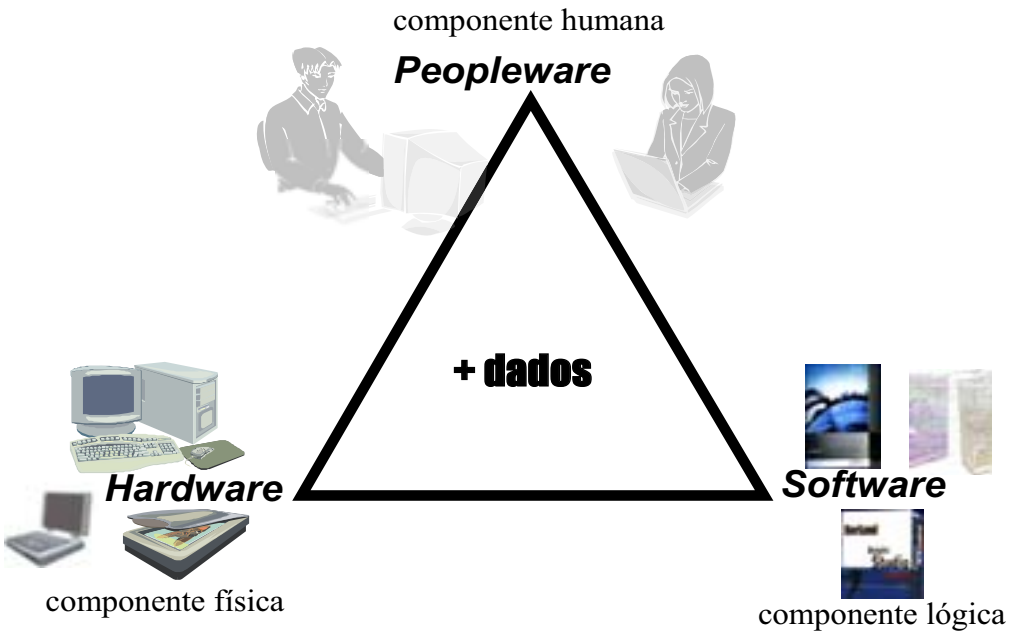

Figura 1.2 – O Sistema Computacional: Hardware, Software e Peopleware. As pessoas utilizam o hardware e o software para manipular dados.

1.1.2 Hardware

Os elementos de hardware de um Sistema Computacional podem ser divididos fundamentalmente em *Unidades de Entrada*, *Unidade Central de Processamento (CPU – Central Processing Unit)*, *Unidades de Armazenamento* e *Unidades de Saída*. A figura 1.3 ilustra a relação entre esses elementos e suas respectivas funções.

Toda a CPU está localizada em um chip chamado de microprocessador. Ela possui a lógica e o circuito para fazer o computador funcionar como um todo, mas não possui espaço para efetuar o armazenamento de instruções e dados durante o processamento. Esta função é exercida pela **Memória Principal (RAM - Random Access Memory)**.

A CPU é composta fundamentalmente por dois componentes denominados de **Unidade de Controle (UC)** e **Unidade Lógico-Aritmética (ULA).** A UC é a parte coordenadora do computador, responsável pela supervisão do funcionamento dos demais componentes funcionais do computador . É responsável por analisar e interpretar cada instrução de programa e ordena a cada parte funcional envolvida que execute sua tarefa na execução daquela instrução. A ULA é responsável pela execução das instruções em si; executa operações aritméticas, comparações entre itens da memória, e movimentos de dados na memória. A figura 1.4 ilustra a

comunicação entre a UC, a ULA, a RAM e as unidades de entrada e de saída.

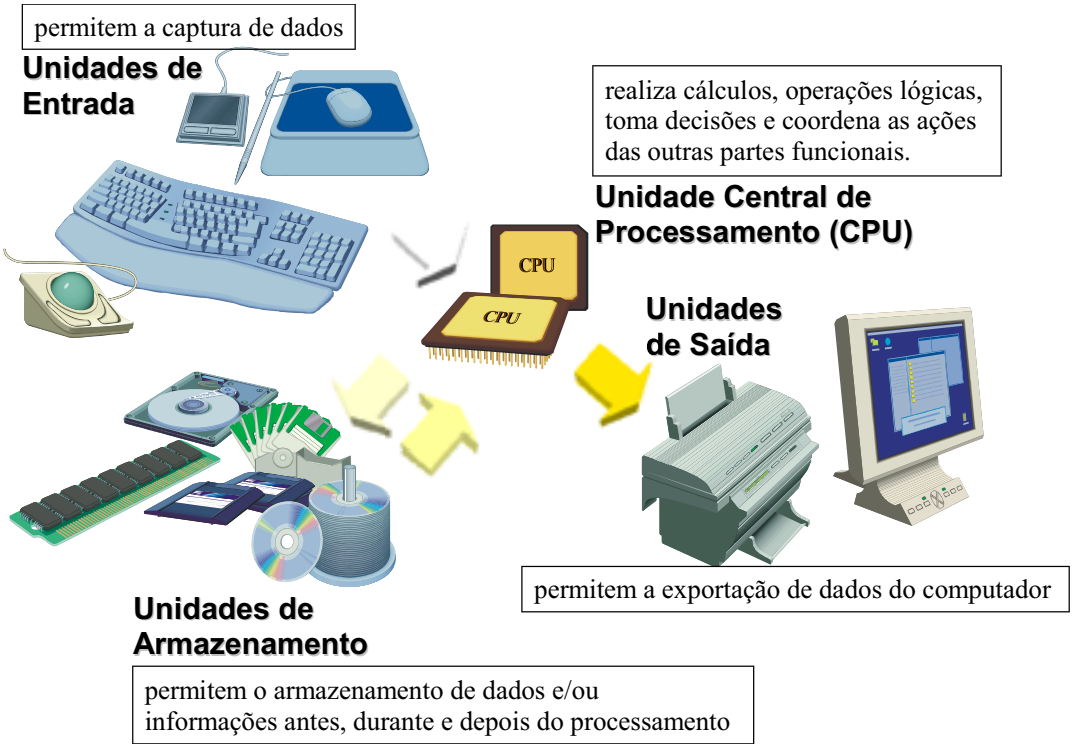

Figura 1.3 – Elementos de hardware de um Sistema Computacional.

A localização de dados na RAM é feita a partir de endereços associados à área da memória onde tais informações foram gravadas. A RAM é organizada como um conjunto de células (áreas de memória), onde cada célula é identificada por um endereço único e armazena dados ou instruções de programas (figura 1.5).

A ação da CPU pode então ser resumida no seguinte fluxo cíclico de atividades:

1. As instruções e os dados são inseridos na memória principal (RAM);
2. A UC extrai da RAM a instrução;
3. A UC extrai da RAM os dados;
4. A UC ordena à ULA a execução da operação, informando o endereço de memória onde por os resultados;
5. A UC identifica na memória a próxima instrução a ser executada.

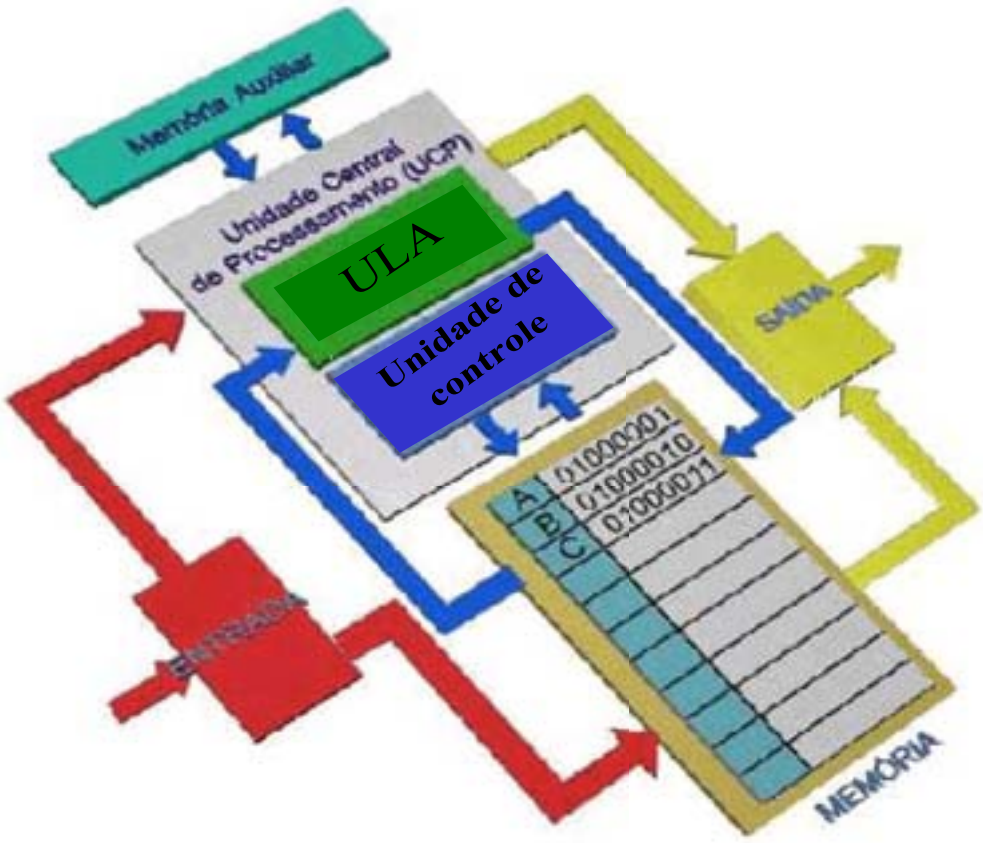

Figura 1.4 – Comunicação entre UC, ULA, RAM e unidades de entrada e saída.

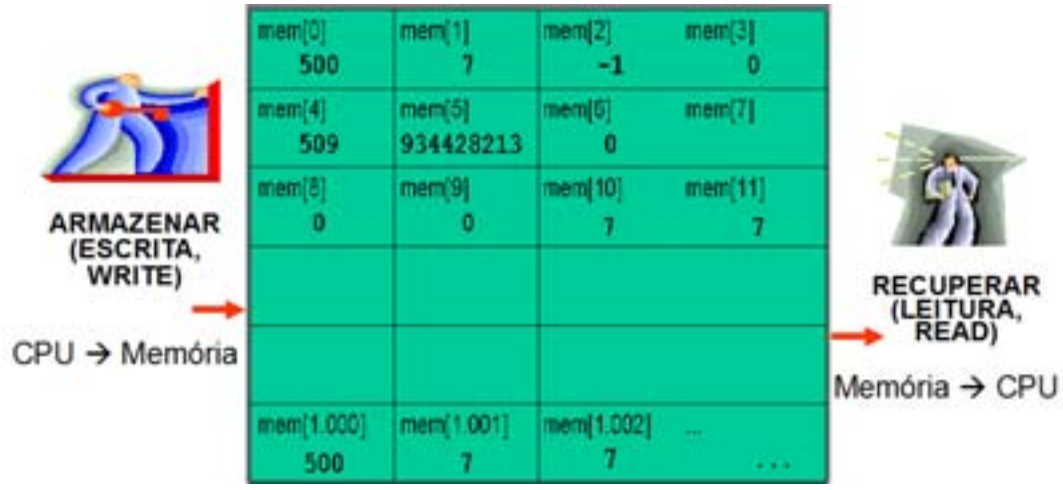

Figura 1.5 – Organização da RAM: cada célula é identificada por um endereço de memória (representado entre colchetes na figura) e armazena o valor do dado ou instrução a ser executada. A CPU "escreve" dados e instruções na memória e "lê" dados e instruções da memória.

1.1.3 Representação e Armazenamento de Dados

Os dados são armazenados nos dispositivos de memória do computador por meio da presença ou ausência de sinais eletrônicos ou magnéticos. No caso de circuitos eletrônicos, o estado pode ser condutor (ON) ou não condutor (OFF). No caso de discos magnéticos o armazenamento é feito através de pontos magnetizados cujos campos possuem uma dentre duas polaridades diferentes. Em ambos os casos, existem apenas duas possibilidades de estados diferentes.

Dessa forma, podemos dizer que o computador é uma **máquina digital binária** que trabalha exclusivamente com dois dígitos, 0 e 1, chamados de ***bits* (Binary Digits)**. Todo dado ou informação manipulado pela máquina é então representado internamente como um conjunto de 0s e 1s. Para o computador, todo tipo de informação são números: números são números, letras são números, sinais de pontuação são números, instruções são números.

Um caractere é a menor informação que precisamos representar internamente e um bit não é suficiente para representá-lo. Assim sendo, foi definido o termo ***byte*** que representa um conjunto de 8 bits. Um byte corresponde ao espaço de memória necessário para guardar um caractere, portanto, pode ser definido como a menor unidade de informação endereçável da memória.

Por essa razão a capacidade de armazenamento da memória no computador é medida em bytes e em seus múltiplos:

1. kilobytes (KB): 2^{10} bytes;
2. megabytes (MB): 2^{20} bytes (~1 milhão de bytes);
3. gigabytes (GB): 2^{30} bytes (~ 1 bilhão de bytes);
4. terabytes (TB): 2^{40} bytes (~ 1 trilhão de bytes);
5. etc.

Quando dizemos que um disco rígido (HD – Hard Disk) possui capacidade de 120GB, estamos dizendo, por exemplo, que ele seria capaz de armazenar um texto com aproximadamente 120 bilhões de caracteres.

Uma vez que todo e qualquer dado ou informação são representados e manipulados pelo computador apenas no formato de números, torna-se necessária a existência de algum tipo correspondência entre os diversos símbolos de nossa linguagem e um grupo de bits que identifique univocamente o referido símbolo. De fato, esta correspondência existe e é padronizada para todos os computadores.

A tabela ASCII (*American Standard Code for Information Interchange*) utiliza vários arranjos de bits para formar bytes que representam os dígitos 0 à 9, as letras, e outros caracteres. A figura 1.6 ilustra alguns símbolos e sua respectiva codificação ASCII.

Caractere	Binário ASCII
ESPAÇO	0010 0000
!	0010 0001
1	0011 0001
9	0011 1001
@	0100 0000
A	0100 0001
b	0110 0010

Figura 1.6 – Alguns símbolos de nossa linguagem e sua respectiva codificação no padrão ASCII.

Compreender a forma de representação e manipulação dos dados e informação pelo computador é muito importante para o aprendizado da programação dos mesmos, como veremos a seguir.

1.2 Linguagens de Programação

Chamamos de **programa de computador** um conjunto de instruções inteligíveis pela máquina que seja capaz de executar alguma atividade.

O principal programa de um computador é seu Sistema Operacional (S.O.). O S.O. é responsável por servir de interface entre o hardware do computador e outros programas (editores de texto, planilhas de cálculo, sistemas de bases de dados, jogos, *browsers* para acesso à Internet, entre outros). Ou seja, ele cria um ambiente onde os usuários podem fazer uso destes programas sem precisar se preocupar com detalhes de hardware específicos de cada máquina. Podemos dizer que é o S.O. é o "supervisor" de todas as operações do computador. São exemplos de Sistemas Operacionais, o Windows (Microsoft), o UNIX e o Linux.

Todo programa em execução no computador deve apresentar ao S.O. uma lista de comandos especificando as operações que o S.O. deverá executar. Estes comandos deverão obrigatoriamente ser apresentados em linguagem de máquina (linguagem binária), a única linguagem que a máquina de fato consegue compreender. Desta forma, os desenvolvedores de programas teriam que escrever todas as instruções de um determinado programa em linguagem de máquina, o que seria extremamente complicado. Como forma de facilitar a escrita de programas de computador, foram criadas as chamadas **Linguagens de Programação (L.P.)**.

Uma linguagem de programação pode então ser definida como um sistema formal de regras de descrição, com sintaxe e semântica bem definidas, capaz de

representar um programa de computador.

Quando a sintaxe da L.P. é mais próxima da sintaxe utilizada pela linguagem humana, dizemos que a L.P. é de ***alto nível***. A sintaxe é composta por notações matemáticas e grupos de palavras para representar as instruções de máquina, tornando o processo de programação mais próximo do entendimento humano. Como exemplo de linguagens desse tipo podemos citar FORTRAN, COBOL, Basic, Pascal, PROLOG, C, C++, Haskell, Java.

Quando a sintaxe da L.P. é distante da sintaxe que nós, seres humanos, compreendemos, dizemos que se trata de uma L.P. de ***baixo nível***. Nesse caso, a sintaxe é composta por números binários, hexadecimais, alguns símbolos e letras. Uma L.P. de baixo nível está muito próxima da linguagem de máquina, onde cada instrução simbólica corresponde, praticamente, a uma instrução de máquina. A figura 1.7 traz algumas linguagens de baixo nível e exemplos de instruções em suas sintaxes respectivas (apenas a título de ilustração).

Linguagem	Exemplo de Programa
Linguagem de Máquina	01001001110101110110 10010010010000110111
Hexadecimal	A1 20 20 63 74 19 20 F2 4D 20 B0 CE 67 20

Assembler	PUSH BP
	MOV BP, SP
	SUB SP, WORKAREA

Figura 1.7 – Exemplos de trechos de programas em algumas linguagens de baixo nível.

1.2.1 Compilação e Execução de Programas

Apesar de facilitar sobremaneira a vida do desenvolvedor, os programas escritos em uma linguagem de alto nível não são diretamente executáveis pela máquina. Eles devem passar por um processo de **tradução** onde as instruções criadas em linguagem de alto nível são convertidas para os respectivos códigos de operação em linguagem de máquina.

A tarefa de tradução de programas é feita por outros programas chamados de **tradutores**. Cada linguagem de programação possui um programa tradutor particular.

Existem fundamentalmente dois tipos de tradutores: **compiladores** e **interpretadores**.

Um compilador traduz programas escritos em linguagem de alto nível para

linguagem de máquina. Ao término do processo, um novo programa, análogo ao programa original, mas escrito em linguagem de máquina, é gerado para posterior execução. Esse programa é chamado de **programa objeto**. Pode existir ainda para alguns compiladores uma fase extra chamada de ligação (*linkage*) em que são feitas ligações entre os códigos de operação gerados pelo compilador e as rotinas e execução das mesmas.

Um interpretador traduz ao mesmo tempo em que executa programas escritos em linguagem de alto nível, sem haver portanto geração de programa objeto. A ação é feita analisando-se linha a linha do código original. Uma vez que também são responsáveis pela execução dos programas, os interpretadores devem estar presentes na memória da máquina junto com o programa a ser utilizado. Por essas características, a execução de um programa interpretado é consideravelmente mais lenta que a de um programa compilado.

1.3 Programação

Programação é a seqüência de planejamento, projeto, escrita e testes de instruções desempenhadas pelo computador. Podemos dizer que a atividade de programar requer uma certa dose de inspiração, de criatividade. Pode, inclusive ser considerada uma arte, porque existem maneiras bastante diferentes de se realizar o trabalho de programação. É obviamente também uma ciência, porque existem algumas regras que devem ser seguidas, porque é necessário o uso de lógica e porque existem alguns métodos rigorosos de programação que asseguram a eficiência, economia e a utilidade dos programas gerados.

O trabalho de programação pode se tornar mais fácil se o dividirmos sistematicamente em partes menos complexas (ver capítulo 6).

Um programa é considerado confiável quando conseguir fazer com que o computador cumpra com o objetivo proposto. Os programas construídos devem ser eficazes, realizando a tarefa definida, e eficientes, utilizando os melhores meios para realizá-la.

O maior problema na construção de programas é a **complexidade**; esta complexidade representa a quantidade de situações diferentes que um problema pode apresentar e que devem ser previstas na solução do mesmo. Portanto, ao se construir um programa, o objetivo principal é vencer a complexidade do problema a ser solucionado.

A fim de lidar com esta complexidade, podemos dividir a programação em **quatro fases distintas** a saber:

1. Definir o problema.

2. Realizar um estudo da situação atual e verificar qual(ais) a(s) forma(s) de resolver o problema.
3. Terminada a fase de estudo, utilizar uma linguagem de programação para escrever o programa que deverá em princípio, resolver o problema.
4. Analisar junto aos usuários se o problema foi resolvido. Se a solução não foi encontrada, deverá ser retornado para a fase de estudo para descobrir onde está a falha.

Estas são de forma bem geral, as etapas que um desenvolvedor de sistemas passa desde a apresentação do problema até a sua efetiva solução.

1.3.1 Conceito de Algoritmo

Um algoritmo pode ser definido como uma seqüência de passos que visam resolver o problema inicialmente proposto. Esses passos devem constituir de **ações claras e precisas**, que a partir de um estado inicial, após um período de tempo finito, produzem um estado final previsível e bem definido. Usualmente, este estado inicial é constituído por dados de entrada e o estado final é constituído por informações resultantes do processamento descrito pelo algoritmo.

A partir dessa definição, podemos perceber que qualquer tarefa pode em princípio ser descrita por um algoritmo.

Considere, por exemplo, que você irá receber uma visita em sua casa e gostaria de oferecer-lhe uma fatia de bolo. Para que o bolo fique pronto, você precisa prepará-lo fazendo uso de alguns ingredientes. Ou seja, seu problema consiste em preparar o bolo. O algoritmo para resolver este problema é a receita do bolo, que irá transformar os ingredientes (dados de entrada) no bolo em si (informação de saída) (ver figura 1.8).

Figura 1.8 – Algoritmo para preparar um bolo.

1.3.2 Representação de um Algoritmo

Uma vez que um algoritmo representa tão somente uma linha de raciocínio, ele pode ser descrito de várias maneiras diferentes. O algoritmo para confecção de um bolo ilustrado anteriormente foi descrito de **forma textual**, fazendo uso de nossa língua portuguesa e sua riqueza gramatical.

Uma alternativa é utilizar uma **representação gráfica** (ex: **fluxogramas**). As formas gráficas são mais fiéis ao raciocínio original e mais concisas, uma vez que substituem um grande número de palavras por convenções de símbolos. A figura 1.9 ilustra um fluxograma que descreve um algoritmo para resolver o problema da troca de uma lâmpada.

Símbolo **TERMINAL**:
início ou fim do algoritmo

Símbolo **PROCESSAMENTO**:
algum tipo de ação

Símbolo **DECISÃO**:
alternativa de fluxo

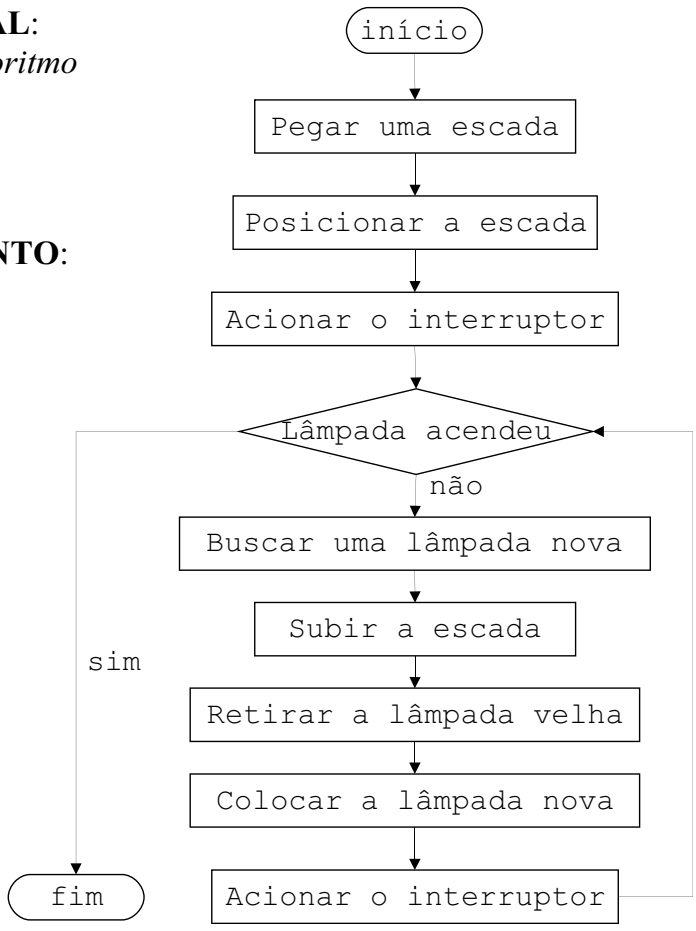

Figura 1.9 – Algoritmo para troca de uma lâmpada com repetição.

Considere agora um outro exemplo. Suponha que queremos desenvolver um programa de computador para calcular a média aritmética de duas notas obtidas por um estudante em uma disciplina na Universidade Federal de Sergipe. Considere ainda que este programa deve informar ao usuário se o estudante está aprovado ou reprovado naquela disciplina, sabendo que a média mínima para

aprovação na referida instituição é 5,0 (CINCO).

A figura 1.10 ilustra um fluxograma que descreve o algoritmo para solucionar este problema.

Tanto a descrição textual quanto o uso de fluxogramas para descrição de algoritmos possui vantagens e desvantagens. Uma das desvantagens da representação gráfica é a necessidade de se estabelecer e conhecer convenções de símbolos, que apesar de simples não são naturais, uma vez que o ser humano está mais condicionado a se expressar por meio de palavras. Uma outra desvantagem é que é sempre mais trabalhoso desenhar algo do que escrever um texto.

Símbolo **ENTRADA DE DADO**:
leitura de dados de alguma fonte

Símbolo **EXIBIR**:
mostra informações, resultados

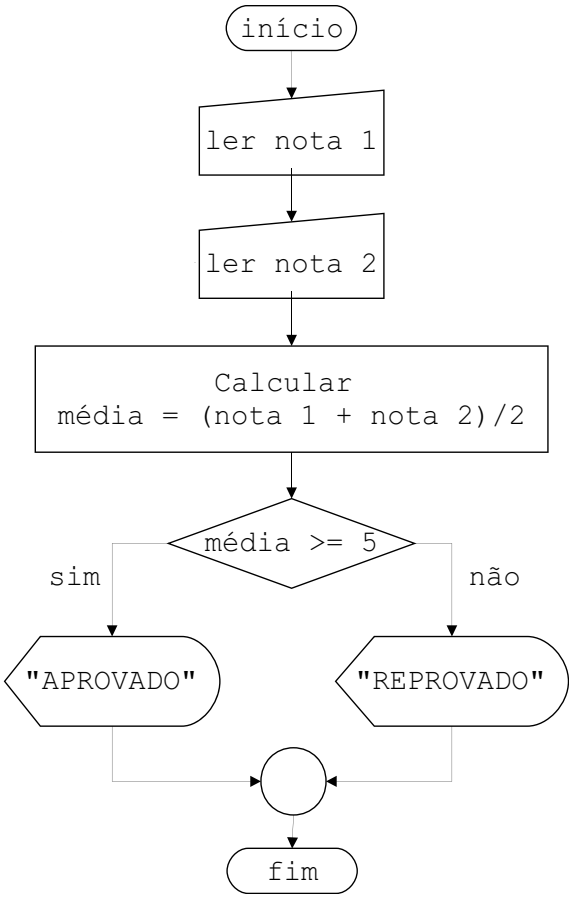

Figura 1.10 – Algoritmo para calcular a média entre duas notas de um estudante e informar se ele foi aprovado ou não.

Por esta razão, a maioria dos autores em cursos de introdução à programação utiliza a descrição textual e este curso fará uso da mesma em toda sua extensão. Para isso, utilizaremos um conjunto de regras que visam restringir e estruturar a língua portuguesa com o objetivo de evitar descrições ambíguas ou prolixas. A linguagem resultante dessa restrição, de fato, se aproxima bastante da sintaxe de linguagens de programação reais tais como Pascal, Java ou C. Isso facilita sobremaneira a respectiva codificação dos algoritmos elaborados.

Daremos o nome de **pseudocódigo** a todo algoritmo escrito nesse sub-conjunto restrito e estruturado da língua portuguesa.

A figura 1.11 mostra o algoritmo para solucionar o problema das notas representado em pseudocódigo. [Nota: *não se preocupe ainda em compreender toda a sintaxe do pseudocódigo em questão*].

```
1.  Início
2.     real: nota1, nota2, media;
3.     leia (nota1, nota2);
4.     media <- (nota1 + nota2)/2;
5.     se (media >= 5)
6.       então
7.         escreva ("aprovado");
8.       senão
9.         escreva ("reprovado");
10.    fimse;
11. Fim.
```

Figura 1.11 – Média de duas notas

1.4 A Linguagem de Programação Java

Para que o algoritmo para solução de um determinado problema possa ser executado por um computador e o resultado de sua computação exibido na tela do monitor, por exemplo, é necessário que o algoritmo seja codificado em uma linguagem de programação real que possa ser então traduzida para a linguagem da máquina em questão.

Ao longo deste curso iremos adotar a linguagem de programação Java para ilustrar a codificação de alguns dos vários pseudocódigos elaborados para solução de vários problemas propostos.

1.4.1 História e Aplicações da Linguagem Java

Java foi criada em 1991 pela empresa Sun Microsystems, Inc., por uma equipe liderada por James Gosling. A motivação inicial para a criação da linguagem era a necessidade de se desenvolver programas que pudessem ser executados em dispositivos com arquiteturas diferentes. Em particular, o objetivo era criar uma linguagem de programação voltada para a programação de dispositivos inteligentes eletrônicos voltados para o consumidor e eletrodomésticos.

Com o rápido avanço e popularidade da *World Wide Web* a partir de 1993, a Sun percebeu o potencial de se utilizar Java para adicionar conteúdo dinâmico, como interatividade e animações, às páginas da *Web*. Mas foi devido à sua capacidade de permitir executar código em um *browser* de navegação na *Web* sem

a necessidade de compilação (os chamados ***Applets***) e o fato de ser **disponibilizada gratuitamente** pela Sun, que Java ganhou rapidamente milhares e milhares de adeptos, tornando-se a **linguagem de programação mais utilizada no mundo** em pouco tempo.

O primeiro kit de desenvolvimento disponibilizado pela Sun foi o *Java Developer's Kit 1.0* (JDK 1.0) que atendia as plataformas *Sun Solaris* e *Microsoft Windows 95/NT*. Em seguida foram disponibilizados kits para as plataformas *IBM OS/2, Linux* e *Macintosh.*

Em 1997 foi lançado o JDK 1.1 incorporando melhorias para o desenvolvimento de aplicações gráficas e distribuídas e em 1999 foi lançado o JDK 1.2, contendo muitas outras melhorias e correções.

A partir de então a Sun vem liberando novas versões ou correções da linguagem frequentemente, bem como novos recursos para o desenvolvimento de aplicações específicas.

Atualmente, Java pode ser utilizada para desenvolvimento de aplicações científicas, aplicativos corporativos de grande porte, aprimorar funcionalidade de servidores *Web*, fornecer aplicativos para diversos dispositivos eletrônicos como telefones celulares, PDAs, *Set Top Box* para aparelhos de TV Digital, entre outros.

1.4.2 Formato de um programa Java

Um programa Java é formado por um conjunto de entidades chamadas **classes**. Uma classe é uma entidade que possui partes específicas chamadas de **subprogramas**. Um subprograma, por sua vez, contém instruções que executam uma determinada tarefa e retornam informações ao concluir. Programadores podem criar classes e diversos subprogramas para codificar da melhor maneira o algoritmo que bolaram para solucionar um determinado problema. Felizmente, boa parte da programação necessária para resolver um determinado problema já está pronta para ser usada em ricas coleções de classes conhecidas como **Java APIs (application programming interfaces)** e disponibilizadas pela Sun e também por fornecedores independentes.

Todo programa Java deve possuir uma **classe principal**. A classe principal deve conter um subprograma chamado **main()** e é onde se inicia e termina a execução de todo o programa. A figura 1.12 ilustra o esqueleto da classe principal de um programa Java.

```
1.  public class nomeDaClasse {
2.     public static void main(String[] args) {
3.        instruções
4.     }
5.  }
```

Figura 1.12 – Esqueleto da classe principal de um programa Java.

Na figura, *nomeDaClasse* é o nome atribuído pelo programador ao programa questão. A classe principal de um programa Java deve estar em um arquivo com o mesmo nome da classe e com **extensão .java**. Neste exemplo, o arquivo deveria se chamar *nomeDaClasse.java*.

Um exemplo mais concreto está ilustrado na figura 1.13. Quando executado, o programa Java em questão exibe a mensagem *Bem vindo ao curso de ICC!* na tela.

```
1.  public class Bemvindo {
2.     public static void main(String[] args) {
3.        System.out.println("Bem vindo ao curso de ICC!");
4.     }
5.  }
```

Figura 1.13 – Exibe mensagem de boas vindas.

Detalhes sintáticos e outras características da linguagem Java serão introduzidos e explicados gradativamente ao longo do curso.

1.4.3 Ambiente de desenvolvimento

Antes de começar a escrever programas Java, é necessário instalar o kit de desenvolvimento disponibilizado pela Sun em seu site (www.java.sun.com). Atualmente, o kit mais recente é o **Java SE Development Kit (JDK) 6 Update 11**. [Nota: *As instruções para download, instalação e configuração em seu computador pessoal estão disponíveis no próprio site e na documentação que acompanha o kit.*]

Uma vez instalado e configurado corretamente o JDK, um programa Java pode então ser criado para representar um determinado algoritmo de solução a um problema. Para isto, cinco etapas são necessárias (figura 1.14): *edição, compilação, carga, verificação e execução*.

O **primeiro passo** consiste em **criar um arquivo** com um software de edição de texto qualquer (ex: Bloco de Notas no Windows) e salvá-lo em seu HD com a extensão .java. Empresas de desenvolvimento de software Java para o mercado ou organizações outras que desenvolvem grandes aplicações usualmente fazem uso de ambientes de desenvolvimento integrados (IDEs – *integrated development environments*) que provêem um arsenal de ferramentas para facilitar a escrita e a depuração (localização de erros) de programas Java. Muitos desses IDEs são gratuitos e estão disponíveis para download. Entre as mais populares se destacam o Eclipse (www.eclipse.org) e o NetBeans (www.netbeans.org).

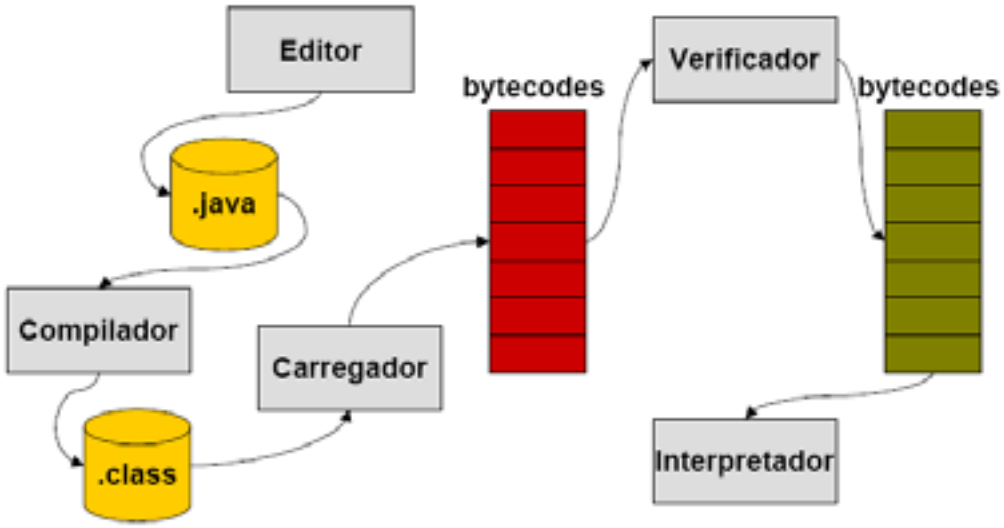

Figura 1.14 – Etapas para criação de um programa Java.

No **segundo passo**, o programador deve utilizar o comando **javac** (compilador java) **para compilar** o programa Java escrito. Para isso, pode-se utilizar a janela de comando do sistema (ex: prompt de comando no Windows XP ou prompt do shell no Linux). Para compilar, por exemplo, o programa `Bemvindo.java` ilustrado anteriormente, poderíamos digitar `javac Bemvindo.java` na janela de comando. Se a codificação não possuir erros sintáticos, o programa será compilado corretamente e o compilador gerará um arquivo **.class** chamado `Bemvindo.class`.

O compilador Java converte o código-fonte Java em **bytecodes**, que representam as instruções a serem executadas durante a etapa de execução. Os bytecodes são executados pela **Java Virtual Machine (JVM)**, que é uma parte da JDKe a base da plataforma Java. Uma máquina virtual é um aplicativo que simula o comportamento de um computador, ocultando o hardware e S.O. Subjacentes. Se uma mesma máquina virtual for implementada para várias plataformas diferentes, os programas que ela executa podem ser utilizados em todas essas plataformas. Isso é o que acontece com um programa escrito em Java graças à JVM, que é implementada para diversas plataformas diferentes. Diferentemente, da linguagem de máquina, os bytecodes são independentes de plataforma, são portáveis.

O comando **java** invoca a JVM e, para executar o programa `Bemvindo`, deveríamos digitar `java Bemvindo`.

Isso dá início ao **passo três**, quando dizemos que o programa é **carregado na memória** principal do computador (RAM). O carregador de classe transfere os arquivos .class contendo os bytecodes do programa para a RAM.

No **passo quatro**, é realizada uma **verificação dos bytecodes** para assegurar

que eles são válidos e não violam restrições de segurança do Java.

Por fim, no **quinto passo**, a JVM executa os bytecodes do programa. Eles são executados utilizando-se de uma **combinação de interpretação e compilação just-in-time (JIT)**, que acelera o processo. No JIT as partes dos bytecodes que executam com freqüência (*hot spots*) são traduzidas para a linguagem de máquina do computador subjacente. Quando a JVM encontra novamente essas partes compiladas, o código de linguagem de máquina mais rápido é executado.

1.4.4 Testando o ambiente de desenvolvimento

Para testar seu ambiente de desenvolvimento vamos executar um primeiro programa Java. Siga os passos atenciosamente e observe o resultado retornado pelo programa.

Utilize um editor de textos qualquer para digitar o exemplo da figura 1.15 [Nota*: não se preocupe ainda em compreender toda a sintaxe do programa em questão*]. O programa simplesmente exibe na tela de execução o mesmo texto fornecido como entrada pelo usuário do programa - por essa razão, lhe foi conferido o nome de `Eco`.

```
1.  public class Eco {
2.      public static void main(String[] args) {
3.          for (int i=0; i<args.length; i++)
4.             System.out.print(args[i] + " ");
5.          System.out.println();
6.      }
7.  }
```

Figura 1.15 – Exibe o mesmo texto fornecido como entrada

Ao finalizar a digitação do código do programa, ele deve ser salvo com a terminação .java. O nome do arquivo armazenado no HD então deve ser `Eco.java`.

O programa Java deve então ser compilado. Para isso, devemos acionar o javac como indicado abaixo (utilize uma janela de comando do sistema, como dito anteriormente):

```
javac Eco.java
```

Esse comando transforma o código digitado em bytecodes, produzindo um arquivo chamado `Eco.class`. Nesse momento podemos invocar a JVM para executar de fato o programa Eco a partir do comando `java`. Digite a linha abaixo e observe o resultado gerado.

```
java Eco Esta mensagem é um eco!
```

Como previsto, o resultado do programa será a exibição da seguinte

mensagem:

```
Esta mensagem é um eco!
```

Resumo

- O computador processa dados a todo instante.
- Processamento de dados consiste em transformar dados brutos fornecidos como entrada em informação útil para alguém.
- Um sistema computacional reúne quatro elementos essenciais para que o processamento de dados seja possível: hardware, software, peopleware e dados.
- O hardware é composto por dispositivos físicos de entrada, saída, processamento e armazenamento.
- O processamento é realizado pela CPU, que lê e escreve dados, informações e instruções na memória principal do computador.
- Uma instrução representa uma ordem de execução.
- Um programa é composto por instruções ordenadas logicamente com o objetivo de solucionar um problema computacional específico.
- A seqüência lógica de instruções para solução de um problema é o que chamamos de algoritmo.
- Um algoritmo pode ser representado em forma de fluxograma ou pseudocódigo.
- Um pseudocódigo utiliza uma linguagem textual de fácil entendimento pelo homem, mas que não é entendível pela máquina.
- Um pseudocódigo pode ser passado para uma linguagem de programação que a máquina entenda.
- Java é uma linguagem de programação que pode ser compilada e executada por uma máquina.
- Compilar um programa significa transformar a sintaxe da linguagem em que foi escrito para a sintaxe da linguagem da máquina em questão.
- Neste curso será utilizada a linguagem Java para implementação real de programas.

Na próxima aula...

… estabeleceremos o conjunto de regras para escrita de pseudocódigos, aprenderemos o conceito de “expressões”, veremos como descrevê-las em formato

de pseudocódigo e como codificá-las na linguagem Java.

Referências e sugestões de leitura

Uma boa introdução à computação, história dos computadores, noções de hardware, microprocessadores, estrutura e organização da informação, linguagens de programação, sistemas operacionais, redes de computadores e Internet é apresentada em:

FEDELI, R. D., GIULIO, E., POLLONI, F., PERES, F. E. Introdução à Ciência da Computação, Thomson Pioneira. 2003.

Uma boa referência para introdução à programação, confecção de algoritmos, pseudocódigos e fluxogramas é o capítulo 3 de:

LEITE, M., Técnicas de Programação: uma Abordagem Moderna, BRASPORT, 2006.

Os capítulos 1 e 2 de:

DEITEL, H., DEITEL, P. Java como Programar, Prentice-Hall, 2005.

trazem uma excelente introdução ao ambiente de desenvolvimento Java e à sua sintaxe.

Para quem quiser compreender profundamente o funcionamento de compiladores e interpretadores recomendo a leitura dos livros

WATT, D., BROWN, D. Programming Language Processors in Java, Prentice Hall, 2000.

e

DELAMARO, M. Como Construir um Compilador Utilizando Ferramentas Java, Novatec, 2004.

Exercícios propostos

1.1 – Consultar a Web e listar exemplos de cada um dos tipos de elementos de hardware citados na figura 1.3.

1.2 – Cite exemplos de linguagens de programação atuais para aplicações comerciais e para aplicações científicas.

1.3 – Faça um diagrama que descreva o processo de compilação e execução de programas, indicando que "ferramentas" (softwares) são utilizados em cada etapa e

suas respectivas funções.

1.4 – Descreva, utilizando uma linguagem textual, um algoritmo bem detalhado para “Trocar o pneu de um carro”. Use a criatividade para imaginar percalços.

1.5 – Utilize a representação de fluxograma para descrever um algoritmo para:

a) calcular o volume de uma esfera de raio R, onde R é fornecido pelo usuário. Considere V = 4/3 * pi * R3;

b) exibir em ordem decrescente três valores inteiros diferentes lidos como entrada.

1.6 – Siga os passos estudados e execute o programa Java a seguir:

```
public class Media {
    public static void main(String[] args) {
          float nota1, nota2, media;
          nota1 = System_in.readFloat();
          nota2 = System_in.readFloat();
          media = (nota1 + nota2)/2;
          if (media >= 5)
                    System.out.println("aprovado");
          else
                    System.out.println("reprovado");
    }
}
```

[Nota 1: o programa em questão corresponde ao pseudocódigo da figura 1.11]

[Nota 2: para compilar o programa em questão, crie um arquivo Java e salve a classe chamada System_in que está lista no Apêndice A deste livro. Esta classe é uma classe auxiliar para facilitar leitura de dados fornecidos pelo usuário.]

[Nota 3: após executar o comando java, o programa irá esperar que você digite duas notas. Digite a primeira nota (ex: 8.0) e pressione ENTER. A seguir digite a segunda nota e pressione ENTER novamente. O resultado então será mostrado na tela.]