

Lógica de Programação

Introdução e primeiros passos dentro da programação de computadores

Elaborada por

Prof^a Aline Mendonça Cordeiro

CAPÍTULO 1

Introdução à lógica de programação

No dia-a-dia é comum utilizarmos a expressão “é lógico” quando nos deparamos com algo óbvio, algo que é claro. No entanto, a palavra lógica pode ter um outro significado. Se recorrermos ao dicionário teremos os seguintes significados (dicionário Michaelis):

1 Modo de raciocinar tal como de fato se exerce: *Lógica natural*.

2 Estudo que tem por objeto determinar quais as operações que são válidas e quais as que não o são: *Lógica formal*, que trata dos conceitos, juízos e raciocínios, independentemente de seu conteúdo.

Ou seja, trata-se de uma forma de raciocinar, segundo os conceitos, juízos e raciocínios. Logo, podemos concluir que a Lógica de programação trata-se de uma forma de raciocinar baseado em premissas e afirmações válidas. Mais precisamente, podemos dizer que a lógica é utilizada para resolver problemas computacionais, seguindo as regras dos paradigmas de programação.

Por exemplo, caso as seguinte premissas sejam verdadeiras:

André é maior que Estela

Estela é maior que Pedro

Pelo raciocínio lógico posso afirmar que André é maior que Pedro.

Em informática, utilizaremos a lógica sempre que formos resolver um problema computacional.

A lógica de programação trata ainda de ordenar o pensamento, ou seja, os passos a serem seguidos para se solucionar um problema computacional.

Esse problema pode ser a execução de um cálculo matemático, a análise de um dado, a gravação de uma informação num banco de dados, etc... Infinitas são as possibilidades do que a lógica de programação pode solucionar.

Para pensar...

Utilizamos a lógica corriqueiramente, quando temos que resolver problemas em nosso dia-a-dia. Dado o problema a seguir, indique uma seqüência lógica para se resolver o problema com 100% de certeza!

Numa gaveta, em um quarto escuro existem 6 pares de meias vermelhas, sendo que tem-se 2,5 vezes desse valor de meias brancas. A quantidade de meias brancas dividida por 3 resulta no total de meias na cor azul, e a quantidade de meias amarelas é 2 vezes a quantidade de meias azuis.

Sabendo-se que o quarto está totalmente escuro, e que não há como identificar as meias pela cor, indique a quantidade mínima de pés de meia que deve ser retirada da gaveta em uma única retirada, que possa garantir que ao menos um par de meias da mesma cor será sorteado.

Algoritmos

Como já dito, a lógica trata de ordenar o pensamento para que os problemas possam ser resolvidos. Em se tratando de informática, o raciocínio lógico nos permite ordenarmos o pensamento para resolvermos problemas computacionais, ou seja, a lógica de programação trata de indicar os passos a serem seguidos para que um problema seja resolvido.

Para tal existem uma gama de ferramentas que auxiliam o programador a estruturar sua lógica de programação, sendo que a ferramenta inicial é o ALGORITMO.

Um **algoritmo** é uma sequência de passos lógicos, ordenados e enumerados que visam atingir um objetivo pré-definido. Ou seja, o algoritmo indica os passos a serem seguidos para se resolver um determinado problema.

Por exemplo:

A camisa está na gaveta.

A gaveta está fechada.

Preciso pegar a camisa.

Logo, preciso abrir a gaveta para pegar a camisa.

Nesse caso, o algoritmo para se pegar a camisa poderia ser assim escrito:

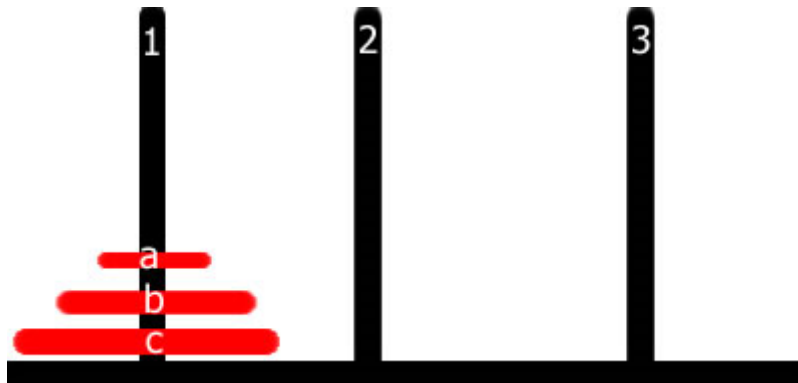
- 1-Abra a gaveta
- 2-Pegue a camisa
- 3- Feche a gaveta

Caso os passos sejam invertidos, por exemplo, se primeiro eu tentar pegar a camisa com a gaveta ainda fechada o objetivo não será atingido, que é pegar a camisa!

Para treinar...

Elaborar o algoritmo que resolva as situações problema abaixo relacionadas:

a) Dada a torre de Hanói:



Criar o algoritmo que permita transferir os três discos da torre 1 para a torre 2, sendo que

- Só pode ser movido um disco de cada vez;
- Um disco maior nunca pode ser colocado sobre um disco menor.

b) Três jesuítas e três canibais precisam atravessar um rio; para tal, dispõem de um barco com capacidade para duas pessoas. Por medidas de segurança não se permite que em alguma margem a quantidade de jesuítas seja inferior à de canibais. Qual a sequência de passos que permitiria a travessia com segurança, sendo que:

O barco não trafega sozinho.

c) Três homens Luis, Carlos e Paulo são casados com Lucia, Patrícia e Maria, mas não sabemos quem é casado com quem. Eles trabalham com Engenharia, Advocacia e Medicina, mas também não sabemos quem faz o que. Com base nas dicas abaixo, descubra o nome de cada esposa e a profissão de cada um.

O médico é casado com Maria

Paulo é Advogado

Patrícia não é casada com Paulo

Carlos não é médico

- d) Uma família de 5 pessoas precisa atravessar uma ponte. No máximo dois integrantes por vez podem atravessar a ponte. Cada pessoa anda em uma velocidade diferente, demorando 1, 3, 6, 8 e 12 segundos na travessia sobre a ponte. A dupla anda na velocidade do mais lento, isto é, se na travessia for a pessoa que demora 1 segundo com a pessoa que demora 12 segundos, eles levarão 12 segundos para atravessar a ponte. Porém está escuro e eles precisam utilizar um lampião, que dura apenas 30 segundos. Levando em conta somente a soma do tempo gasto nas travessias, como você levaria esta família até o outro lado da ponte em até 30 segundos?
- e) Você possui dois recipientes irregulares um com capacidade de 3 litros e outro com capacidade de 5 litros, sem nenhuma marcação e uma torneira. Elaborar um algoritmo para obter precisamente 4 litros de água no recipiente maior.

Concluindo...

Já vimos que para resolvermos um problema através da construção de algoritmos podemos seguir um método, entre outros existentes, que traduza uma sequência lógica de passos. Estes envolvem descobrir as informações, ações e resultados.

Quando se trata de um algoritmo computacional, a lógica que devemos seguir é a mesma. Devemos identificar o problema (objetivo), descobrir as informações e enumerar as ações a serem seguidas para se chegar ao objetivo.

CAPÍTULO 2

ALGORITMOS COMPUTACIONAIS

Quando criamos um algoritmo para um computador o princípio é o mesmo. Temos que transformar as ações a serem executadas em comandos, ordens, que possibilitam, se seguidas, alcançarmos um determinado objetivo. Vejamos o exemplo a seguir que demonstra a forma de notação de um algoritmo computacional que resolve um problema específico.

Exemplo de algoritmo computacional

Criar um algoritmo que faça a leitura de dois valores numéricos e apresente a soma dos mesmos.

Levantamento de dados:

Objetivo: Apresentar a soma de dois valores lidos

Informações: Os dois números a serem somados

Ações: Efetuar a soma e apresentar o resultado

O algoritmo seria assim escrito:

- 1- Leia o primeiro valor (valor1)
- 2- Leia o segundo valor (valor2)
- 3- Efetuar a soma (resultado \leftarrow valor1+valor2)
- 4- Apresentar a soma (resultado)

Onde valor1, valor2 e resultado são uma espécie de incógnitas, onde os valores lidos e o resultado podem ficar armazenados temporariamente para serem exibidos posteriormente. Essas incógnitas são chamadas de variáveis, que estudaremos a seguir.

Variáveis

As variáveis são espaços reservados na memória do computador para que dados sejam armazenados temporariamente. As variáveis recebem esse nome porque seus valores podem se alterar durante a execução do programa, ou seja, seus valores variam de acordo com as ações a serem executadas.

Imagine a memória como um grande arquivo, cheio de gavetas. Cada uma dessas gavetas seriam as variáveis, onde os dados são guardados, sendo que:

- Toda gaveta (variável) deve conter um identificador único;
- Cada gaveta (variável) só é capaz de armazenar um único dado por vez.

As variáveis são alocadas na memória RAM do computador, logo, ao se desligar ou reiniciar um computador seus dados são perdidos.

Toda variável deve ser declarada antes de ser utilizada, ou seja, é necessário informar à memória do computador que um espaço deverá ser reservado para se armazenar esses dados.

Identificadores

Como já dito, toda variável deve ser identificada, ou seja, deve ter um nome único. Para facilitar a sua identificação, devemos seguir as regras abaixo no momento de definir os identificadores das variáveis:

- Todas as letras em minúsculas, somente a primeira letra da segunda sentença deverá ser escrita em maiúscula, no chamado estilo *camel case*. Exemplo: idadeAluno, nomePais, dataNascimento, valor1;
- O primeiro caractere deve obrigatoriamente ser uma letra;
- Pode conter números, mas não se iniciar com um número;
- Não podem ser usados caracteres especiais como %, \$, #, @, ", ç, ! , espaços em branco, caracteres com acentos, etc... Os únicos caracteres permitidos além de letras e números é o underscore (_) e o hífen (-);
- Os identificadores devem ser escolhidos de modo a explicitar as informações que serão armazenadas dentro da variável, por exemplo, nome, endereço, idade, telefone, sexo, são exemplos de identificadores adequados.

Constantes

Por vezes, dentro da programação, devemos utilizar valores pré-definidos, que não se alteram durante a execução do programa. Esses valores são chamados de constantes, pois permanecem os mesmos, sendo que o usuário não consegue alterá-los em momento algum.

As constantes devem ser declaradas com tipo, nome e o valor que conterão.

Para declarar constantes, utilizaremos sempre letras maiúsculas, utilizando o *underline* (_) para separar duas ou mais palavras. Exemplo: VALOR_PI, COTACAO_DOLAR, etc.

Exemplo de utilização de uma constante:

Criar um algoritmo que leia o valor do raio de uma circunferência e calcule e apresente a área.

- 1- Leia o Raio (raio)
- 2- Definir o valor de PI (VALOR_PI \leftarrow 3.14)
- 3- Calcule a área da circunferência (area \leftarrow VALOR_PI*raio*raio)

4- Apresente a area (area)

Neste caso o valor de π é uma constante, sempre será 3.14, indiferentemente do valor do raio. Nesse caso o usuário poderá escolher o valor do raio, mas não o de π , pois π é uma constante e seu valor não se altera durante a execução do programa.

Tipos de dados primitivos

Os dados a serem armazenados pelas variáveis podem ser de quatro tipos primitivos, a saber:

INTEIRO:

Toda e qualquer informação numérica que pertença ao conjunto de números inteiros (negativo, nulo ou positivo). Exemplo: 234, -9, 0, -90, 3422, 5634, etc;

REAL:

Toda e qualquer informação numérica que pertença ao conjunto dos números reais (negativo, nulo ou positivo). Ex: 1.75 m de altura, R\$ 325.42 de saldo bancário, 2.5 m de fio, R\$1252.56 de salário líquido.

CARACTERE ou TEXTO:

Toda e qualquer informação composta por um conjunto de caracteres alfanuméricos (A..Z, 0..9) e ou especiais (#,\$,@,+,..., inclusive espaço em branco) .

Obs.: Todas as informações do tipo caractere devem ser sempre delimitadas por apóstrofes.

Ex: 'Proibido Fumar', 'Fabio', 'Rua XV de Novembro, 422', 'F' (representa sexo feminino), 'S' (representa resposta sim).

LÓGICO:

Toda e qualquer informação que pode apenas assumir duas situações, VERDADEIRO ou FALSO, SIM ou NÃO. Também conhecido como tipo de dados BOOLEANO. Ex: Você é aluno dessa ETEC? VERDADEIRO ou FALSO. Você é eleitor? SIM ou NÃO. Você é maior de idade? SIM ou NÃO.

DECLARAÇÃO DE VARIÁVEIS

No ambiente computacional as informações são armazenadas, sendo que cada variável pode guardar apenas uma informação de cada vez, sendo sempre do mesmo tipo. Portanto precisamos associar as variáveis a tipos, conforme a informação que se pretende armazenar. A declaração destas variáveis deve seguir a seguinte sintaxe:

IDENTIFICADOR DA VARIÁVEL: tipo de dados;

Ex:

x, a, idade: inteiro;

b: inteiro;

nome, endereco: caractere;

salario: real;

resposta: lógico;

COMANDO DE ATRIBUIÇÃO

Um comando de atribuição permite-nos fornecer um valor (conteúdo) a uma certa variável, sendo que o tipo dessa informação deve ser compatível com o tipo da variável. Possui a seguinte sintaxe:

identificador \leftarrow expressão ou valor;

Ex.: $x \leftarrow 2$;

$y \leftarrow 3 / x$;

$a \leftarrow \text{verdadeiro}$;

$b \leftarrow x + 8$;

OPERADORES ARITMÉTICOS

Nas linguagens de programação podemos utilizar operadores matemáticos para realizar cálculos de expressões. Os principais operadores aritméticos, em ordem de precedência são:

OPERADOR	TIPO DE DADOS	OPERAÇÃO
*	REAL ou INTEIRO	Multiplicação
/	REAL	Divisão real (com casas decimais)
DIV ou /	INTEIRO	Divisão com resto (sem casas decimais)
MOD ou %	INTEIRO	Resto da divisão
+	REAL ou INTEIRO	Adição
-	REAL ou INTEIRO	Subtração

Exemplos:

$r \leftarrow 5/2$ (r será = 2,5)

$r \leftarrow 5 \text{ DIV } 2$ (r será = 2)

$r \leftarrow 5 \text{ MOD } 2$ (r será = 1)

$r \leftarrow 5 + 1$ (r será = 6)

$r \leftarrow 5 * 2$ (R será = 10)

Construindo algoritmos

Agora que já sabemos algumas regras e ferramentas para a construção de programas, podemos criar algoritmos computacionais utilizando variáveis, constantes e operadores aritméticos.

Exemplo:

Criar um programa que leia uma temperatura em graus CELSIUS apresente-a convertida em graus FAHRENHEIT:

Levantamento de dados:

Objetivo: Apresentar uma temperatura em graus Celsius convertida em graus Fahrenheit.

Informações: A temperatura em graus Celsius

Ações: Efetuar a conversão utilizando a fórmula: $f = c \times 1,8 + 32$

Onde f é a temperatura em Fahrenheit e c em Celsius

A solução para esse problema ficaria assim escrita:

Algoritmo temperatura:

- 1- *Leia temperatura em graus Celsius (c)*
- 2- *Converter a temperatura em Fahrenheit ($f \leftarrow c * 1.8 + 32$)*
- 3- *Apresentar a temperatura convertida em Fahrenheit (f)*

Diagrama de blocos

O algoritmo não é a única ferramenta utilizada para solucionarmos problemas computacionais. Embora seja bastante eficiente, o algoritmo não possui uma padronização. Por exemplo, a instrução “escreva” pode ser escrita por alguns como “exiba” ou “mostre”, fazendo com que cada programador crie a sua própria maneira de resolver os problemas.

O diagrama de blocos é uma representação gráfica do algoritmo, tornando padronizadas as ações expressas nos passos do algoritmo computacional.

No diagrama de blocos cada ação é representada por um símbolo, a saber:



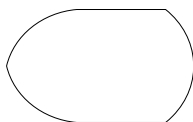
Terminal: Utilizado para indicar o início e o fim de um diagrama de blocos.



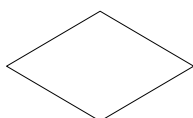
Processamento: Utilizado para indicar um processamento qualquer, como uma atribuição ou cálculo.



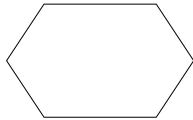
Entrada: Indica uma entrada (leitura) de dados manual.



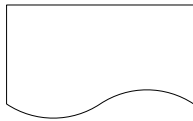
Display: Utilizado para indicar que uma informação será exibida na tela (escrita).



Decisão: Utilizado para indicar uma tomada de decisão, ou seja, um desvio dentro do programa.



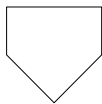
Preparação: utilizado para inicializar, testar e incrementar uma variável num laço de repetição.



Impressão: Utilizado para indicar que um dado será impresso. Muito utilizado em relatórios.



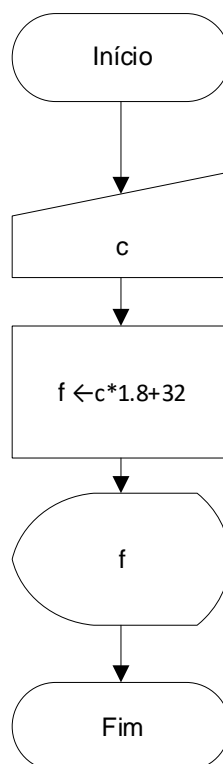
Referência na página: Conector utilizado para indicar que o diagrama continua em algum outro ponto na mesma página.



Referência fora da página: Conector utilizado para indicar que o diagrama continua em algum outro ponto na próxima página.

Exemplo:

Criar o diagrama de blocos do algoritmo de temperatura da página 10:



Outro Exemplo:

Criar o algoritmo e o diagrama de blocos do seguinte problema proposto:

Criar um programa que calcule e apresente o volume de uma lata de óleo, onde o raio e a altura deverão ser informados pelo usuário.

Levantamento de dados:

Objetivo: Apresentar o volume de uma lata de óleo

Informações: O raio e a altura e o valor de $\text{PI}=3.14$.

Ações: Efetuar o cálculo utilizando a fórmula:

$$V = \text{PI} * R^2 * h$$

Algoritmo:

- 1- Leia o valor do raio(r)
- 2- Leia o valor da altura (h)
- 3- Calcule o volume ($v \leftarrow 3.14 * r * r * h$)
- 4- Apresente o volume (v)

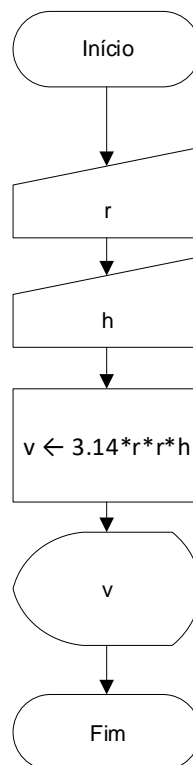
Onde:

V é o volume

R é o Raio

H é a altura

Diagrama de blocos:



Exercícios de fixação:

Criar o diagrama de blocos de todos os algoritmos já resolvidos no caderno.

CAPÍTULO 3

Português Estruturado (Pseudocódigo ou Portugol)

O diagrama de blocos tem a função de padronizar os algoritmos de forma gráfica. Porém, após a sua criação, é necessário ainda codificar o diagrama de blocos em uma linguagem de programação qualquer, como o Turbo Pascal, Java, ou Linguagem C ou Visual Basic, por exemplo.

Porém, acontece que cada linguagem de programação tem as suas palavras reservadas, que diferem de uma linguagem para outra. É como se cada linguagem de programação tivesse o seu próprio dicionário, sendo que muitas vezes o que serve para uma linguagem não sirva para outra. Além disso, todas as linguagens de programação do mercado utilizam-se do idioma Inglês como base, tornando ainda mais difícil a sua interpretação.

Para facilitar a compreensão das linguagens de programação utilizaremos um pseudocódigo conhecido como Português Estruturado ou Portugol, que se trata de uma linguagem fictícia criada para estruturar um programa da mesma forma que em uma linguagem de programação, porém utilizando o nosso idioma pátrio.

O Portugol como qualquer outra linguagem de programação possui uma lista de palavras chave, ou seja, palavras reservadas que indicam que uma ação está sendo executada. Ao longo do semestre conheceremos melhor cada uma dessas palavras, sendo que as principais são:

inicio, programa, var, leia, escreva, se...então...senão...fim_se, enquanto...fim_enquanto, para...de...até...faça...fim_para, escolha, caso, pare, caso_contrário, fim.

Essas palavras reservadas não podem ser utilizadas para identificar variáveis dentro de um programa.

Vamos agora estruturar um diagrama de blocos utilizando as palavras reservadas do Portugol para que fique mais claro como o mesmo é utilizado.

Exemplo:

Criar o código em Português estruturado do algoritmo da página 10 (temperatura).

programa TEMPERATURA

 var c, f: real

inicio

 leia(c)

$f \leftarrow c * 1.8 + 32$

 escreva (f)

fim

Explicação:

A palavra chave **programa** indica o nome que foi dado ao mesmo.

A palavra chave **var** indica que nessa área serão declaradas as variáveis a serem utilizadas pelo programa.

A palavra chave **início** marca o início do bloco de programação.

A palavra chave **leia** indica que o dado foi lido.

A expressão $f \leftarrow c * 1.8 + 32$ indica que um processamento, um cálculo ocorreu e foi guardado na variável F.

A palavra chave **escreva** indica que um dado foi apresentado na tela.

A palavra chave **fim** indica que o bloco de programação se encerrou.

Exercícios de fixação

Criar o código em português estruturado de todos os algoritmos criados no caderno.

CAPÍTULO 4

Codificação em Java

O Português Estruturado é uma pseudo linguagem, ou seja, na prática o mesmo não existe. Algumas universidades criaram uma versão de português para auxiliar a aprendizagem de lógica por parte de seus alunos, entretanto, para solidificar os conhecimentos em programação faz-se necessário a implementação dos algoritmos na prática, ou seja, testar o algoritmo numa linguagem que permita executar as ações a fim de solucionar o problema proposto.

O Java é uma linguagem de programação multiplataforma, que devido à sua JVM (*Java Virtual Machine*) é capaz de ser executado em qualquer dispositivo, indiferente do sistema operacional que está instalado no computador onde o programa será executado. Quando criamos um programa em Java o mesmo deve ser salvo com a extensão .JAVA. Ao compilarmos, é criado um código chamado Byte Code, que é intermediário, e pode ser executado pela JVM de qualquer sistema operacional. Exatamente por isso o Java conquistou o mercado e é a linguagem de programação mais utilizada para dispositivos móveis e principalmente em dispositivos que não são especificamente um computador.

Para programarmos em JAVA você deverá baixar e instalar a JDK (Java Development Kit) em seu computador, e se desejar, uma IDE (Interface de Desenvolvimento) como o Eclipse ou Netbeans, por exemplo.

O código que está após o símbolo `//` ou entre os símbolos `/** ...*/` são comentários, que serão ignorados pelo compilador. Os comentários devem ser utilizados sempre que possível, pois ajudam a organizar o código fonte e irão ajudar na manutenção do código.

História da Linguagem Java

A contribuição mais importante da revolução do microprocessador até agora é que ele tornou possível o desenvolvimento de computadores pessoais, que agora contam com centenas de milhões em todo o mundo. Os computadores pessoais modificaram profundamente a vida das pessoas e a maneira que as organizações conduzem e gerenciam seus negócios.

Os microprocessadores têm um impacto profundo em dispositivos inteligentes eletrônicos voltados para o consumidor. Reconhecendo isso a SUN Microsystems, em 1991, financiou um projeto de pesquisa corporativa interna com codinome Green, que resultou no desenvolvimento de uma linguagem baseada em C++ que seu criador, James Gosling, chamou de Oak em homenagem a uma árvore de carvalho vista por sua janela na Sun. Descobriu-se mais tarde que já havia uma linguagem de computador chamada Oak. Quando uma equipe da Sun visitou uma cafeteria local, o nome Java (cidade de origem de um tipo de café importado) foi sugerido; e o nome pegou.

O projeto Green passou por algumas dificuldades. O mercado de dispositivos eletrônicos inteligentes voltado para o consumo popular não estava desenvolvendo, no início da década de 1990, tão rápido como a Sun havia antecipado. O projeto corria o risco de ser cancelado. Por uma feliz casualidade, a World Wide Web explodiu em popularidade em 1993 e a equipe da Sun viu o imediato potencial de utilizar o Java para adicionar conteúdo dinâmico, como interatividade e animações, às páginas da Web. Isso deu nova vida ao projeto.

A Sun anunciou o Java formalmente em uma importante conferência em maio de 1995. O Java chamou a atenção da comunidade de negócios por causa do enorme interesse na World Wide Web. O Java é agora utilizado para desenvolver aplicativos corporativos de grande porte, aprimorar a funcionalidade de servidores Web (os computadores que fornecem o conteúdo que vemos em nossos navegadores da web), fornecer aplicativos para dispositivos voltados para o consumo popular (por exemplo, telefones celulares, pagers e PDAs) e para muitos outros propósitos.

Termos Técnicos de JAVA

Ambientes de desenvolvimento Java

■ JSE (Java Standard Edition)

- ☐ Seu uso é voltado a PCs e servidores.
- ☐ Contém todo o ambiente necessário para a criação e execução de aplicações desktop e web de pequeno e médio porte.
- ☐ Pode-se dizer que essa é a plataforma principal, já que, o JEE e o JME tem sua base aqui.

■ JEE (Java Enterprise Edition)

- ☐ Voltada para o desenvolvimento de softwares corporativos.
- ☐ Baseados em componentes que são executados em um servidor de aplicação.

■ JME (Java Micro Edition)

- ☐ Ambiente de desenvolvimento para dispositivos móveis ou portáteis, como telefones celulares e palmtops.

Componentes básicos da linguagem Java

■ JRE (Java Runtime Environment)

- ☐ Significa Ambiente de Tempo de Execução
- ☐ É um pacote de softwares, que é executado como um aplicativo do sistema operacional e que interpreta a execução de programas Java
- ☐ A JRE é composta pela JVM somada ao conjunto de API's. (JVM + API's = JRE)

■ API (Application Programming Interface)

- ☐ Significa Interface de Programação de Aplicativos
- ☐ Biblioteca (ou uma série delas) com funções e procedimentos públicos que permitem aos programadores desenvolverem aplicações fazendo uso de recursos já definidos.

■ JVM (Java Virtual Machine)

- ☐ Significa Máquina virtual Java
- ☐ Software que emula uma CPU e Memória para a execução de programas Java.

■ JDK (Java Development Kit) ou SDK (Software Development Kit)

- ☐ Significa Kit de Desenvolvimento Java
- ☐ Conjunto de ferramentas para a compilação, documentação e debug de aplicativos Java.
- ☐ Composto pela JRE somada as ferramentas de desenvolvimento.

JVM - Java Virtual Machine

Todos os que lidam com informática sabem que os computadores apenas entendem a chamada linguagem binária, representada pela conhecida notação de zeros e uns. As linguagens de alto nível como C, Pascal, Java etc, não tem utilidade para a máquina, a não ser que passem pelo processo de transformação que conhecemos como compilação. Ao término desse processo, o computador é suprido por uma massa de código binário capaz de ser compreendida e executada pelo processador.

O esquema de compilação em Java resulta em uma forma intermediária de código binário chamado de bytecodes. Essa forma intermediária não é uma linguagem de máquina que possa ser entendida por qualquer processador. Os bytecodes são construídos de forma que seja possível submetê-los a um processo de tradução (interpretação) em cada máquina.

O interpretador Java, responde pela tradução dos bytecodes em código binário reconhecido pelo processador específico, recebe o nome de “Máquina Virtual Java”. Isso significa que cada um dos diferentes sistemas (Windows, Macintosh, Solaris..) terá que ser suprido pelo próprio interpretador Java. Os bytecodes, portanto, representam a independência de arquitetura há anos buscada sem sucesso pela tecnologia da informação.

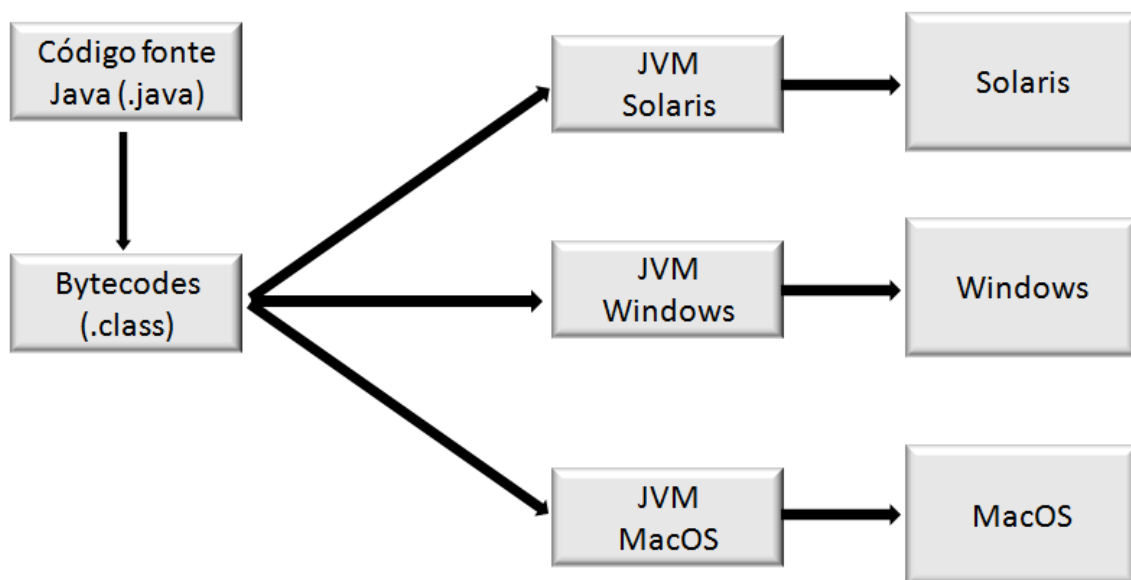
Em C e Pascal, temos o seguinte quadro quando vamos compilar um programa:

Código Fonte em C – compila – código fonte para um determinado Sistema Operacional.

Temos um código executável para cada Sistema Operacional.

O Java se utiliza do conceito de máquina virtual, onde existe, entre o sistema operacional e a aplicação uma camada extra responsável por “traduzir” o que sua aplicação deseja fazer para as respectivas chamadas do Sistema Operacional onde ela está rodando no momento.

“Write Once, Run Anywhere” (Escreva uma vez, execute em qualquer lugar)



Fases de um programa em Java

Os programas em Java passam por cinco fases: edição, compilação, carga, verificação e execução.

1 – Edição: consiste em editar um arquivo com um programa editor. Você digita o código fonte, faz as modificações necessárias e salva o programa em um dispositivo de armazenamento.

IDEs (Integrated Development Environment – Ambiente de desenvolvimento integrado), estão disponíveis a partir de muitos fornecedores de software por exemplo Eclipse, Jcreator entre outros.

2 – Compilação: O compilador Java converte o código fonte em bytecodes que representam tarefas a serem realizadas durante a fase de execução. Os bytecodes são executados pela JVM uma parte do JDK e a base da plataforma Java. A máquina virtual é um aplicativo de software que simula um computador, mas oculta o sistema operacional e o hardware subjacente dos programas que interagem com a VM.

Os bytecodes são instruções independentes de plataforma isto é, os mesmos bytecodes podem executar em qualquer plataforma uma JVM que entende a versão do Java em que os bytecodes foram compilados.

3 – Carga: O programa deve ser colocado na memória antes de poder executar – um processo conhecido como carga. O carregador de classe transfere os arquivos .class contendo os bytecodes do programa para a memória principal.

4 – Verificação: enquanto as classes são carregadas, o verificador de bytecodes examina seus bytecodes para assegurar que eles são válidos e não violam restrições de segurança do Java.

5 – Execução: A JVM executa os bytecodes do programa, realizando assim as ações especificadas pelo programa. Os programas Java passam por duas fases de compilação – uma em que o código fonte é traduzido em bytecodes (para a portabilidade entre JVMs em diferentes plataformas de computador) e uma segunda em que durante a execução, os bytecodes são traduzidos em linguagem de máquina para o computador real em que o programa é executado.

Objetos e Classes

Uma classe é um tipo definido pelo usuário que contém o molde, a especificação para os objetos, algo mais ou menos como o tipo inteiro contém o molde para as variáveis declaradas como inteiros. A classe envolve, associa, funções e dados, controlando o acesso a estes, defini-la implica em especificar os seus atributos (dados) e seus métodos (funções).

Atributos e Métodos

Atributos são características, propriedades dos objetos que possuem valor e tipos definidos. Métodos são funções, comportamentos dos objetos.

Por exemplo uma classe chamada Pessoa, tem como atributos nome, data de nascimento, endereço, características comuns a todos os objetos do tipo Pessoa. Um de seus métodos pode ser calcular a idade, comportamento comum aos objetos da classe Pessoa.

Operadores Aritméticos do JAVA

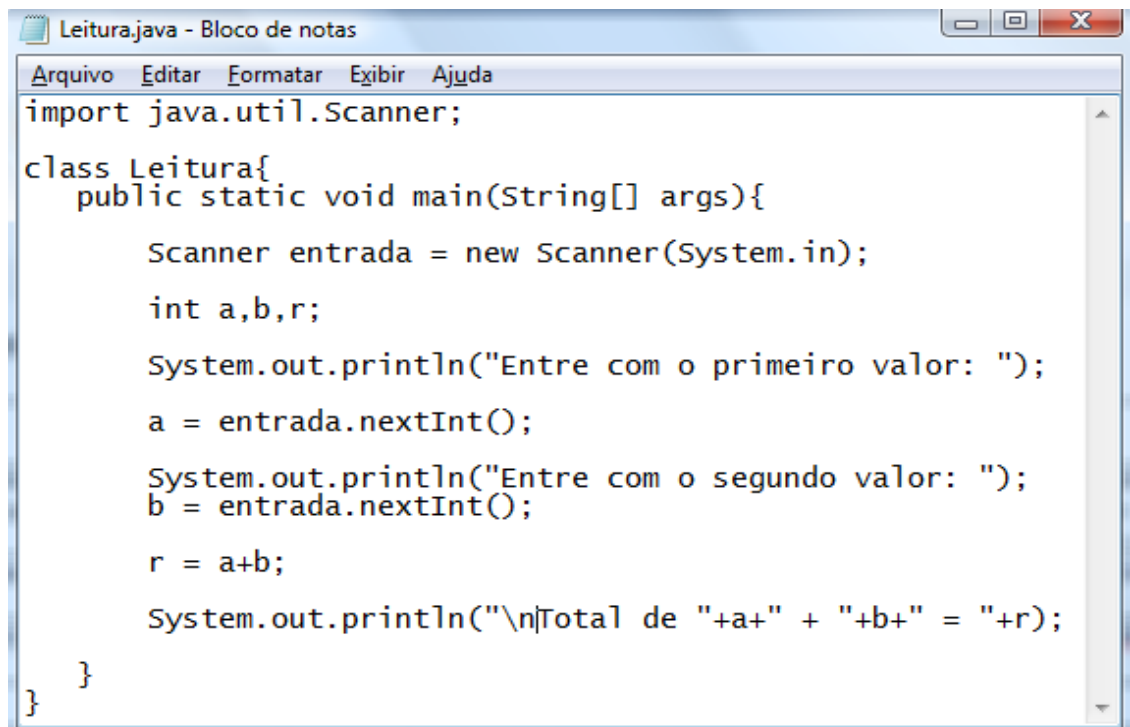
+	Adição
-	Subtração
*	Multiplicação
/	Divisão Inteira e Real
%	Resto da Divisão

Tipos de dados Primitivos em Java

Para:	Tipo	
Valores lógicos	boolean	Armazena true ou false
Aritmética de Inteiros	byte	-128 a 127
	short	-32768 a 32767
	int	-2147483648 a 2147483647
	long	-9223372036854775808 a 9223372036854775807
Aritmética de Reais	float	-3.4E38 a 3.4E38
	double	-1.7E308 a 1.7E308
Manipulação de caracteres	char	É baseado em inteiros

```
public class Primeira {  
  
    public static void main(String[] args) {  
        System.out.println("Olá Mundo!!!");  
    }  
  
}
```

Classe Scanner



```
Leitura.java - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda
import java.util.Scanner;
class Leitura{
    public static void main(String[] args){
        Scanner entrada = new Scanner(System.in);
        int a,b,r;
        System.out.println("Entre com o primeiro valor: ");
        a = entrada.nextInt();
        System.out.println("Entre com o segundo valor: ");
        b = entrada.nextInt();
        r = a+b;
        System.out.println("\nTotal de "+a+" + "+b+" = "+r);
    }
}
```

Métodos para leitura de diversos tipos de dados:

Inteiro – nextInt();

Float – nextFloat();

Double – nextDouble();

Char – nextChar();

String – next();

Vejamos a seguir como fica o código do programa temperatura escrito na IDE (Interface Development – Interface de desenvolvimento) Eclipse do Java.

```
//importação da Classe Scanner para leitura de dados
import java.util.Scanner;

/**criação da classe ConverteTemperatura - sempre a 1ª letra maiúscula */
public class ConverteTemperatura {

    //assinatura do método main - Linha obrigatória
    public static void main(String[] args) {
        //Criação do objeto in para leitura dos dados
        Scanner in = new Scanner(System.in);
        //criação das variáveis c e f
        double c, f;

        //envio de mensagem na tela para o usuário
        System.out.println("Entre coma temperatura em graus Celsius:");
        //Faz a leitura e guarda na variável c

        c = in.nextDouble();

        //calcular o valor convertido e guarda em f

        f = c * 1.8+32;

        //mostra o resultado na tela para o usuário

        System.out.println("A temperatura convertida é: "+f);
    }
}
```

Exercícios de fixação

Criar o código em Java dos algoritmos já feitos no caderno.

Para treinar: Lista de Exercícios I

Criar o algoritmo, diagrama de blocos, código em Português Estruturado e Código em Java que resolva os problemas abaixo:

- 1) Criar um programa que leia o salário de uma pessoa. Sabendo-se que para viver sem dívidas a pessoa pode comprometer apenas 30% da sua renda com dívidas, informar o valor máximo de dívidas que o usuário poderá contrair mensalmente.
- 2) Criar um programa que leia o nome, o ano de nascimento e o ano atual, e ao final apresente a seguinte mensagem: "FULANO, VOCÊ TEM X ANOS" onde FULANO é o nome do usuário e X é a idade do mesmo.
- 3) Criar um programa que leia quatro notas de um usuário, calcule e apresente a sua média aritmética.
- 4) Criar um programa que calcule o salário líquido de um professor. Para fazer esse programa você deverá ler o valor da hora aula, o número de aulas dadas no mês e o percentual de desconto do INSS. O salário líquido será obtido multiplicando-se o número de aulas dadas pelo valor da hora aula e descontando o percentual do INSS.
- 5) Efetuar o cálculo de uma prestação em atraso utilizando a fórmula:

$$\text{PRESTACAO} \leftarrow \text{VALOR} + (\text{VALOR} * (\text{TAXA}/100) * \text{TEMPO})$$

Onde:

VALOR é o valor original da prestação

TAXA é a taxa de juros aplicada ao dia para correção da prestação

TEMPO é a quantidade de dias que a prestação está atrasada

- 6) Ler uma temperatura em graus Fahrenheit e apresentá-la convertida em graus Celsius utilizando a fórmula: $C = (F - 32) * (5/9)$ onde F é a temperatura em Fahrenheit e C em Celsius.
- 7) Uma loja de animais precisa calcular o custo da criação de coelhos. O custo é calculado com a fórmula: $\text{CUSTO} = (\text{NUM_COELHOS} * 0,70)/18 + 10$.
- 8) Calcular e apresentar a área de uma sala retangular sendo que a largura de suas duas paredes devem ser informadas pelo usuário.

CAPÍTULO 5

Tomada de Decisão

Até o momento já vimos como podemos lidar com a entrada, processamento e a saída de dados. Entretanto, existirão situações em que teremos que programar desvios de execução dentro de um código. Por exemplo, o programa não irá terminar após processar os dados. Pode ser necessário analisar o resultado processado para então se tomar uma decisão dentro do programa, escolhendo por onde o fluxo de dados irá trafegar.

Imaginemos a seguinte situação: um programa que leia duas notas de um aluno, calcule e apresente a sua média e além da média, exiba uma mensagem informando se o aluno foi reprovado ou aprovado seguindo os critérios abaixo:

- Se a média for menor do que 6 o aluno será reprovado;
- Se a média for maior ou igual a 6 o aluno será aprovado.

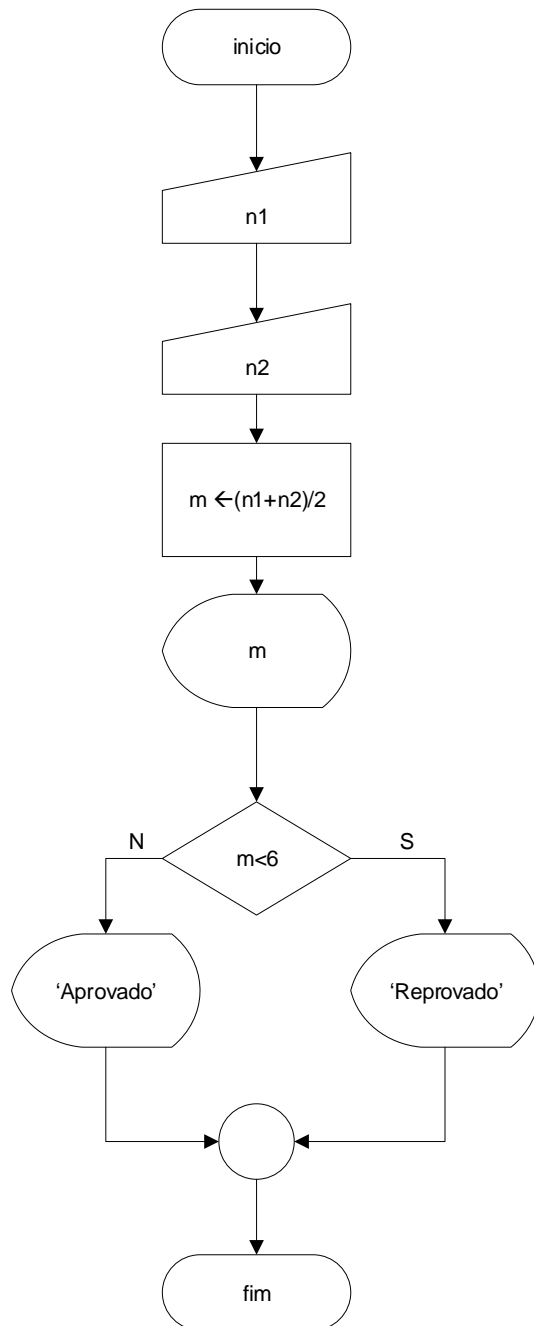
Observe que somente uma das duas mensagens será exibida: ou *aprovado* ou *reprovado*, sendo impossível que o mesmo aluno receba as duas mensagens. Nesse caso será necessário prever um desvio condicional dentro do programa, também conhecido como *tomada de decisão*. O fluxo do programa deverá ser desviado de acordo com a situação da média do aluno.

A solução para esse problema em 4 etapas poderá ser assim escrita:

Algoritmo

- 1- Leia a primeira nota (N1)
- 2- Leia a segunda nota (N2)
- 3- Calcular a média ($M \leftarrow (N1+N2)/2$)
- 4- Apresentar a média (M)
- 5- Se ($M < 6$) então
 - 6 – Escreva ('Reprovado')
- 7 - Senão
 - 8- Escreva ('Aprovado')

Diagrama de blocos



Nessa iteração do diagrama de blocos o programa irá analisar o valor de m . Caso m seja menor do que 6, ou seja, caso essa condição seja **VERDADEIRA** o programa imprimirá a mensagem 'Reprovado' na tela. Caso m não seja menor do que 6, ou seja, caso a condição seja **FALSA** o programa imprimirá a mensagem 'Aprovado' na tela. Depois do final do bloco de decisão o programa segue seu curso normal até encontrar o FIM.

Código em Português Estruturado

programa NOTAS

var

n1, n2, m: real

início

leia (n1)

leia(n2)

$m \leftarrow (n1+n2)/2$

escreva (m)

se (m<6) então

escreva ('Reprovado')

senão

escreva('Aprovado')

fim_se

fim.

Código em Java

```
import java.util.*;
public class SomaNumeros {
    public static void main(String args[]){
        Scanner in = new Scanner(System.in);
        double n1, n2, m;

        System.out.println("Entre com a 1a nota");
        n1 = in.nextDouble();
        System.out.println("Entre com a 2a nota:");
        n2 = in.nextDouble();
        m = (n1+n2)/2;
        System.out.println("A média é: "+m);
        if(m<6) {
            System.out.println("Reprovado! ");
        } else {
            System.out.println("Aprovado! ");
        }
    }
}
```

Operadores relacionais

Para testarmos as condições nas tomadas de decisão, teremos que utilizar um dos operadores abaixo, a saber:

Operador	Nome
==	Igual a
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
!=	Diferente de

Tipos de tomada de decisão

Quando falamos de programação existem 3 tipos de tomada de decisão:

- Tomada de decisão simples

Neste tipo de decisão uma condição será verificada. Caso a mesma seja verdadeira, um bloco de ações será executado. Caso seja falsa a condição é encerrada, sem que nenhuma ação seja executada.

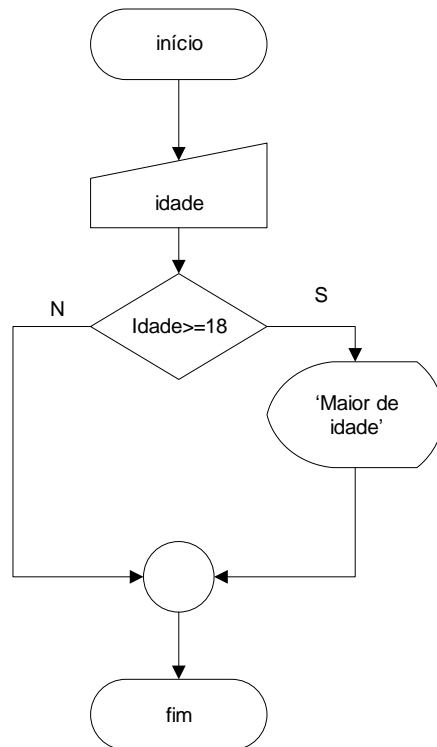
Exemplo de tomada de decisão simples:

Criar um programa que apresente a mensagem “Maior de idade” caso a idade lida seja igual ou superior a 18 anos.

Algoritmo

- 1- Leia a idade(idade)
- 2- Se (idade >=18) então
- 3- Escreva (“Maior de idade”)

Diagrama de blocos



Código em Português Estruturado

```
programa DECISAOSIMPLES  
  
var  
    idade: inteiro  
  
inicio  
    leia(idade)  
    se(idade >= 18) então  
        escreva('Maior de idade')  
    fim_se  
  
fim.
```

- **Tomada de decisão composta**

Neste tipo de decisão uma condição será verificada. Caso a mesma seja verdadeira, um bloco de ações será executado. Caso seja falsa um outro bloco de ações é executado.

Como exemplo de tomada de decisão composta podemos analisar o exercício da página 19.

- **Tomada de Decisão Encadeada:**

Neste tipo de tomada de decisão uma condição é verificada. Caso seja verdadeira, um bloco de ações é executado. Caso seja falsa, uma nova condição pode ser verificada, até que se esgotem todas as possibilidades de teste.

Exemplo: Criar um programa que leia duas notas de um usuário, calcule e apresente a sua média, e em seguida uma mensagem segundo a condição da média:

- Se a média for < 4: Reprovado

- Se a média for >= 6: Aprovado.

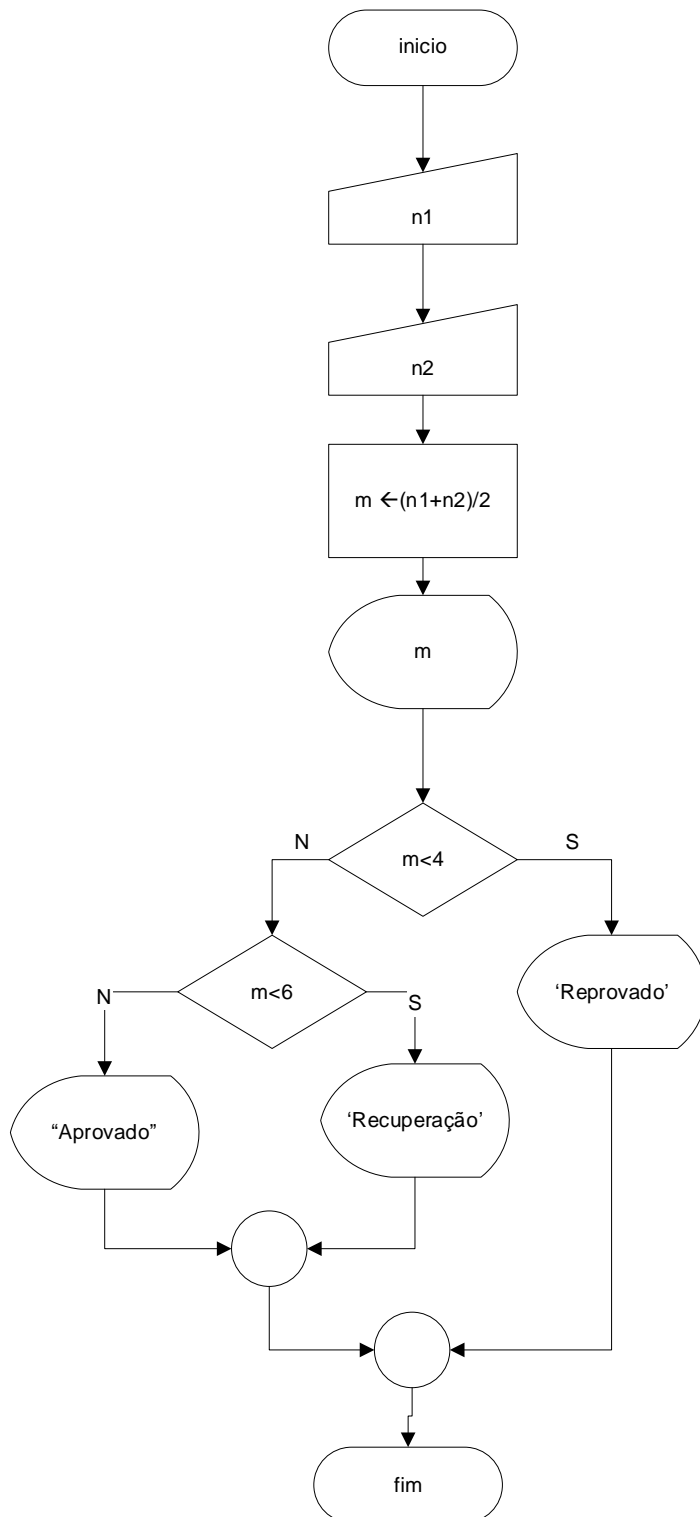
- Se a média for ≥ 4 e < 6 : Recuperação

Neste caso a solução para esse problema, nas 4 etapas, ficaria assim escrita:

Algoritmo

- 1- Leia a primeira nota (n_1)
- 2- Leia a segunda nota (n_2)
- 3- Calcular a média ($m \leftarrow (n_1 + n_2) / 2$)
- 4- Apresentar a média (m)
- 5- Se ($m < 4$) então
6 – Escreva ('Reprovado')
- 7- Senão
8- Se ($m < 6$) então
9- Escreva ('Recuperação')
- 10- Senão
11- Escreva ('Aprovado')

Diagrama de blocos



Código em Português Estruturado

programa NOTAS

var

n1, n2, m: real

início

leia (n1)

leia(N2)

$m \leftarrow (n1+n2)/2$

escreva (m)

se (m<4) então

escreva ('Reprovado')

senão

se (m<6) então

escreva('Recuperação')

senão

escreva('Aprovado')

fim_se

fim_se

fim.

Código em java

```
import java.util.*;
public class TomadaDecisao {
    public static void main(String args[]){
        Scanner in = new Scanner(System.in);
        double n1, n2, m;
        System.out.println("Entre com a 1a nota:");
        n1 = in.nextDouble();
        System.out.println("Entre com a 2a nota:");
        n2 = in.nextDouble();
        m = (n1+n2)/2;
        System.out.println("A média é: "+m);
        if(m<4) {
            System.out.println("Você está reprovado!");
        } else if (m<6) {
            System.out.println("Você está de recuperação ");
        } else {
            System.out.println("Você está aprovado! ");
        }
    }
}
```

Exercícios de fixação

Criar o algoritmo, diagrama de blocos, código em português estruturado e código em Java para os enunciados abaixo:

- a- Criar um programa que leia o salário de um usuário e informe se o salário lido está abaixo ou acima do salário mínimo, que atualmente é de R\$ 954,00.
- b- Criar um programa que leia um número e informe se o mesmo é neutro (zero), positivo ou negativo.
- c- Criar um programa que leia uma temperatura em Fahrenheit e converta-a em graus Celsius. Apresentar a temperatura convertida, junto com a mensagem indicada:
 - i. Se a temperatura estiver abaixo de 15°C: “Frio”
 - ii. Se a temperatura estiver igual ou acima de 15° e abaixo de 22°C: “Ameno”
 - iii. Se a temperatura estiver igual ou acima de 22°C: “Calor”

Operadores lógicos

Quando programamos utilizando estruturas de tomada de decisão, não raramente precisamos reunir mais de um teste lógico em um único SE. Para trabalharmos com mais de uma condição utilizamos os operadores lógicos, a saber:

OPERADOR	TABELA VERDADE		
	CONDIÇÃO 1	CONDIÇÃO 2	RESULTADO
E (AND)	VERDADEIRA	VERDADEIRA	VERDADEIRA
	VERDADEIRA	FALSA	FALSA
	FALSA	VERDADEIRA	FALSA
	FALSA	FALSA	FALSA
OU (OR)	VERDADEIRA	VERDADEIRA	VERDADEIRA
	VERDADEIRA	FALSA	VERDADEIRA
	FALSA	VERDADEIRA	VERDADEIRA
	FALSA	FALSA	FALSA
NÃO (NOT)	VERDADEIRA	_____	FALSA
	FALSA	_____	VERDADEIRA

Exemplos:

OPERADOR E: Para se alistar no exército, uma pessoa tem que ser do sexo masculino E ter 17 anos. O código em português estruturado ficaria assim escrito:

```
se (idade =17) e (sexo = 'm') então
    escreva ('Usuário deve se alistar')
fim_se
```

OPERADOR OU: Para se aposentar uma pessoa deve ter 30 anos ou mais de contribuição ou 65 anos ou mais de idade:

```
se (tempo_contribuicao >=30) ou (idade >=65) então
    escreva('Usuário pode se aposentar')
fim_se
```

OPERADOR NÃO: Para continuar a executar um programa a resposta da pergunta “Deseja continuar?” não pode ser igual a N:

```
Se não(resposta = 'n') então
    escreva('Programa continuará sua execução')
    ...
```

Lista de exercícios nº 02: Tomada de Decisão

Criar o algoritmo, diagrama de blocos, código em português estruturado e código em JAVA dos enunciados a seguir (no caderno). Utilize o JAVA como ferramenta de apoio para testar seus programas.

- 1- Criar um programa que leia o ano de nascimento de um usuário e o ano atual. Calcule e apresente a sua idade junto com mensagem correspondente:
 - iv. Idade < 10 anos: criança;
 - v. Idade >=10 e < 18: Adolescente;
 - vi. Idade >=18 e Idade <60: Adulto;
 - vii. Idade >=60 : Idoso.
- 2- Criar um programa que calcule e apresente o gasto médio de combustível (km por litro – km/L) de um veículo. O usuário deverá digitar a distância percorrida (em Kilômetros) e a capacidade do tanque de combustível (em litros). O gasto médio é obtido dividindo-se a distância percorrida pela capacidade do tanque. Informar se o carro é econômico ou não, segundo a regra abaixo:
 - a. Se o consumo for >=10 km/L: Econômico
 - b. Se o consumo < 10 km/L: Não econômico
- 3- Criar um programa que leia o peso e a altura de um usuário. Informar o seu IMC junto com a respectiva condição:

IMC	Classificação
< 18,5	Excesso de Magreza
18,5 - 25	Peso Normal
25 - 30	Excesso de Peso
30 - 35	Obesidade (Grau I)
35-40	Obesidade (Grau II)
>40	Obesidade (Grau III)

- 4- Criar um programa que leia os valores A, B e C de uma equação de segundo grau (Ax^2+Bx+C). Calcular as duas raízes reais de X lembrando que:
- Se $\Delta < 0$: não existem raízes reais para a equação;
 - Se $\Delta = 0$: existe apenas uma raiz real para a equação;
 - Se $\Delta > 0$: existem duas raízes reais para a equação.

Fórmulas:

$$\Delta = B^2 - 4 \cdot A \cdot C$$

$$\text{Bhaskara: } (-B \pm \sqrt{\Delta}) / (2 \cdot A)$$

- 5- Criar um programa que leia 3 valores A, B e C para um possível triângulo e informe o tipo de triângulo lido. Verificar ainda se as medidas formam um triângulo segundo a lei:
- Para que se possa formar um triângulo é necessário que a medida de qualquer um dos lados seja menor que a soma das medidas dos outros dois.

Triângulo Equilátero: Três lados iguais;

Triângulo Escaleno: Três lados diferentes;

Triângulos Isósceles: Dois lados iguais e um lado diferente.

- 6- Criar um programa que leia três valores inteiros A, B e C e os apresente em forma crescente (menor para o maior).
- 7- Criar um programa que leia o nome e a idade de 5 usuários. Ao final apresente o nome e a idade do usuário mais velho e o nome e a idade do usuário mais novo.
- 8- Criar um programa que leia duas notas para um aluno. Calcular e apresentar a sua média, sendo que:
- SE a média for menor do que 3 o aluno está REPROVADO;
 - SE a média for ≥ 6 o aluno está APROVADO;
 - SE a média for ≥ 3 e < 6 avisar via mensagem que o aluno está em EXAME. Solicitar então uma nota de EXAME, extrair uma nova média entre a média anterior e a nota de EXAME. Caso a nova média seja ≥ 6 o aluno será APROVADO. Caso contrário está reprovado.
- 9- Criar um programa que leia o salário de um empregado e apresente o desconto do IRPF segundo a tabela abaixo:

Base de cálculo mensal em R\$	Alíquota %	Parcela a deduzir do imposto em R\$
Até 1.434,59	-	-
De 1.434,60 até 2.150,00	7,5	107,59
De 2.150,01 até 2.866,70	15,0	268,84
De 2.866,71 até 3.582,00	22,5	483,84
Acima de 3.582,00	27,5	662,94

Exemplo: Se o salário lido for igual a R\$ 4000,00 o cálculo será: $\text{IRPF} \leftarrow 4000 \cdot 27,5 / 100 - 662,94$

CAPÍTULO 6

Estrutura Escolha..Caso (Switch..case)

Esse tipo de estrutura nos permite escolher um valor guardado dentro de uma variável e, ao depender do valor contido nessa variável, propor uma lista finita de opções que podem ser realizadas em cada caso. Essa estrutura somente pode ser usada para comparações exatas, não podendo ser aplicada em intervalos numéricos.

Por exemplo, imagine a seguinte situação:
Qual o dia da semana que você nasceu?

Para responder a essa questão existe uma lista com 7 dias da semana, de 1 a 7 (sendo o 1 o domingo e 7 o sábado) que poderiam responder a essa pergunta. Qualquer valor fora desse intervalo seria interpretado com uma resposta errada.

Para esses casos, a melhor estrutura a ser utilizada é a Escolha..Caso ou Switch..case. Observe o exemplo a seguir.

Criar um programa que solicite ao usuário um número inteiro de 1 a 7, e devolva o dia da semana correspondente, sendo:

- 1 – Domingo
- 2 – Segunda-feira
- 3 – Terça-feira
- 4- Quarta-feira
- 5- Quinta-feira
- 6- Sexta-feira
- 7- Sábado

A solução para esse problema ficaria assim escrita:

Algoritmo

- 1- Leia o dia da semana em que nasceu como inteiro (d)
- 2- Escolha (d)
 - 3- Caso 1:
 - 4- Escreva ("domingo")
 - 5- Caso 2:
 - 6- Escreva("segunda-feira")
 - 7- Caso 3:
 - 8- Escreva("terça-feira")
 - 9- Caso 4:
 - 10- Escreva("Quarta-feira")
 - 11- Caso 5:
 - 12- Escreva("Quinta-feira")
 - 13- Caso 6:
 - 14- Escreva("Sexta-feira")
 - 15- Caso 7:
 - 16- Escreva("Sábado")
 - 17- Caso contrário:
 - 18- Escreva("Dia inválido")

Português Estruturado

programa DiaDaSemana

var

 d: inteiro

início

 leia(d)

 escolha(d)

 caso 1: escreva('Domingo')

 pare

 caso 2: escreva('Segunda')

 pare

 caso 3: escreva('Terça')

 pare

 caso 4: escreva('Quarta')

 pare

 caso 5: escreva ('Quinta')

 pare

 caso 6: escreva ('Sexta')

 pare

 caso 7: escreva ('Sábado')

 pare

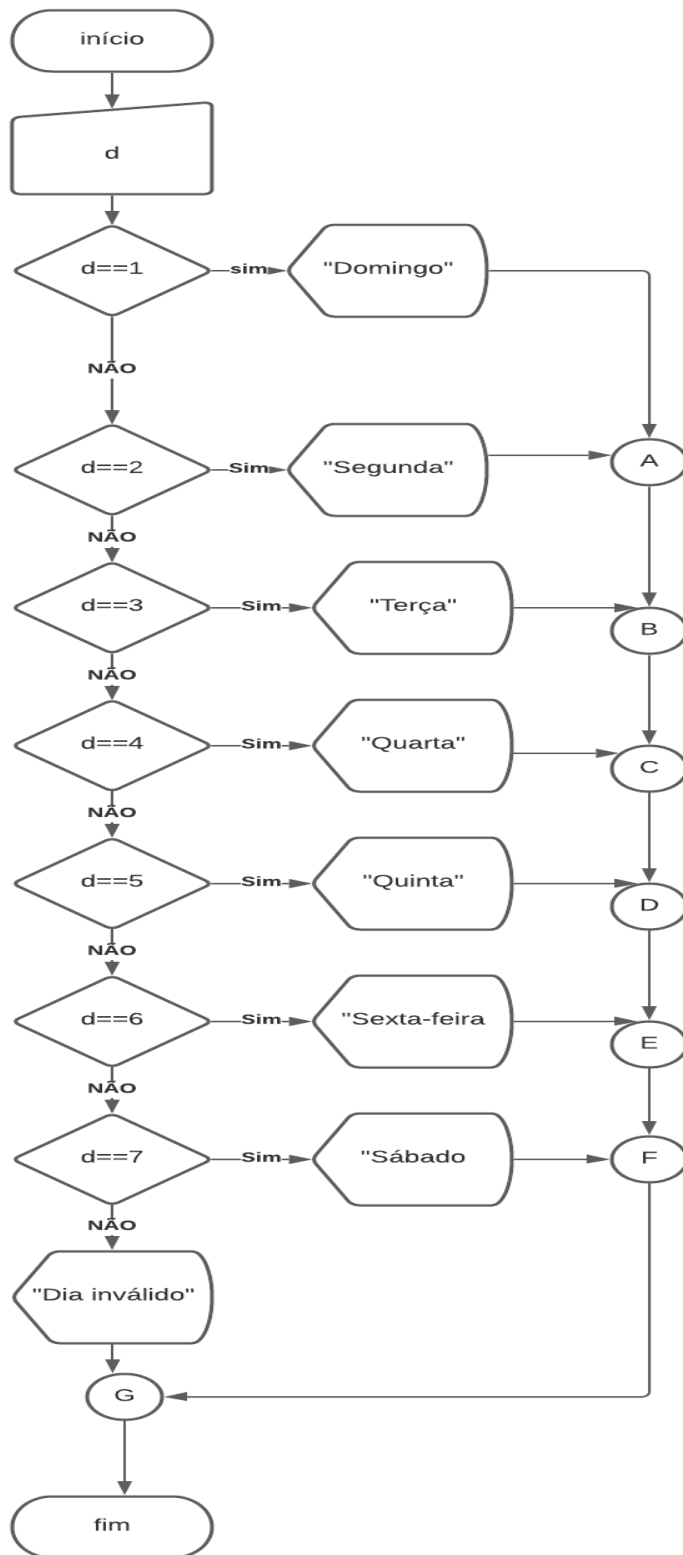
 caso_contrário:

 escreva ('Dia inválido')

 fim_escolha

fim

Diagrama de blocos



Código em Java

```
1 import java.util.Scanner;
2 public class DiaDaSemana {
3     public static void main(String[] args) {
4         Scanner in = new Scanner(System.in);
5         int d;
6         System.out.println("Entre com um número inteiro para saber a "
7             + "que dia esse número corresponde");
8         d = in.nextInt();
9         //Estrutura switch-case
10        switch(d) {
11            case 1:
12                System.out.println("Domingo");
13                break;
14            case 2:
15                System.out.println("Segunda-feira");
16                break;
17            case 3:
18                System.out.println("Terça-feira");
19                break;
20            case 4:
21                System.out.println("Quarta-feira");
22                break;
23            case 5:
24                System.out.println("Quinta-feira");
25                break;
26            case 6:
27                System.out.println("Sexta-feira");
28                break;
29            case 7:
30                System.out.println("Sábado");
31            default:
32                System.out.println("Dia inválido");
33        }
34    }
35 }
36 }
```

Podemos ainda fazer um agrupamento com o switch..case conforme exemplo a seguir:

```
1 import java.util.Scanner;
2 public class DiaDaSemana {
3     public static void main(String[] args) {
4         Scanner in = new Scanner(System.in);
5         int d;
6         System.out.println("Entre com um número inteiro para saber a "
7             + "que dia esse número corresponde");
8         d = in.nextInt();
9
10        //Estrutura switch-case
11        switch(d) {
12            case 1:
13            case 3:
14            case 5:
15            case 7:
16                System.out.println("Dia ímpar");
17                break;
18            case 2:
19            case 4:
20            case 6:
21                System.out.println("Dia par");
22                break;
23            default:
24                System.out.println("Dia inválido");
25        }
26    }
27 }
28 }
```

Exercícios sobre switch..case (Fazer nas 4 etapas: algoritmo, diagrama de blocos, código em português estruturado e código em Java).

- 1- Criar um programa que leia o último número da placa de um veículo e informe qual dia da semana o veículo não poderá circular em São Paulo.
- 2- Criar um programa que leia um número correspondente a um mês qualquer (de 1 a 12) e devolva se o referido mês tem 28, 30 ou 31 dias (desconsiderar anos bissextos).

LAÇOS DE REPETIÇÃO

Quando criamos um programa, cada linha do código é executada uma única vez. Para que possamos repetir uma seqüência de passos dentro do código-fonte, devemos executar o programa quantas vezes forem necessárias. Entretanto, imagine a seguinte situação: um programa que leia duas notas de um determinado aluno, calcule e apresente a sua média aritmética e informe se o aluno está aprovado (caso a média seja ≥ 6) ou reprovado (caso a média seja < 6). Esse programa deverá ser executado para 5 alunos.

Para que isso seja possível, com os comandos e estruturas que aprendemos até o momento, teríamos que criar um programa enorme, repetindo 5 vezes a mesma seqüência de passos, pois a fórmula para saber se um aluno está aprovado ou reprovado é a mesma para qualquer aluno. Imagine ainda que ao invés de 5 alunos tenhamos que repetir essa operação para 40 alunos. O

código-fonte ficaria enorme, repetindo 40 vezes o mesmo trecho. O mesmo ocorreria caso tivéssemos 100 ou 1000 alunos.

Quando devemos repetir um determinado trecho de um programa ou até mesmo todo o código, utilizamos uma estrutura chamada de **LAÇO DE REPETIÇÃO** que nos permite executarmos mais de uma vez um mesmo trecho do código-fonte, baseado em uma condição (teste lógico).

Existem três tipos básicos de laços de repetição, conforme veremos a seguir.

LAÇO DE REPETIÇÃO COM TESTE NO INÍCIO (ENQUANTO ... FAÇA / WHILE ... DO)

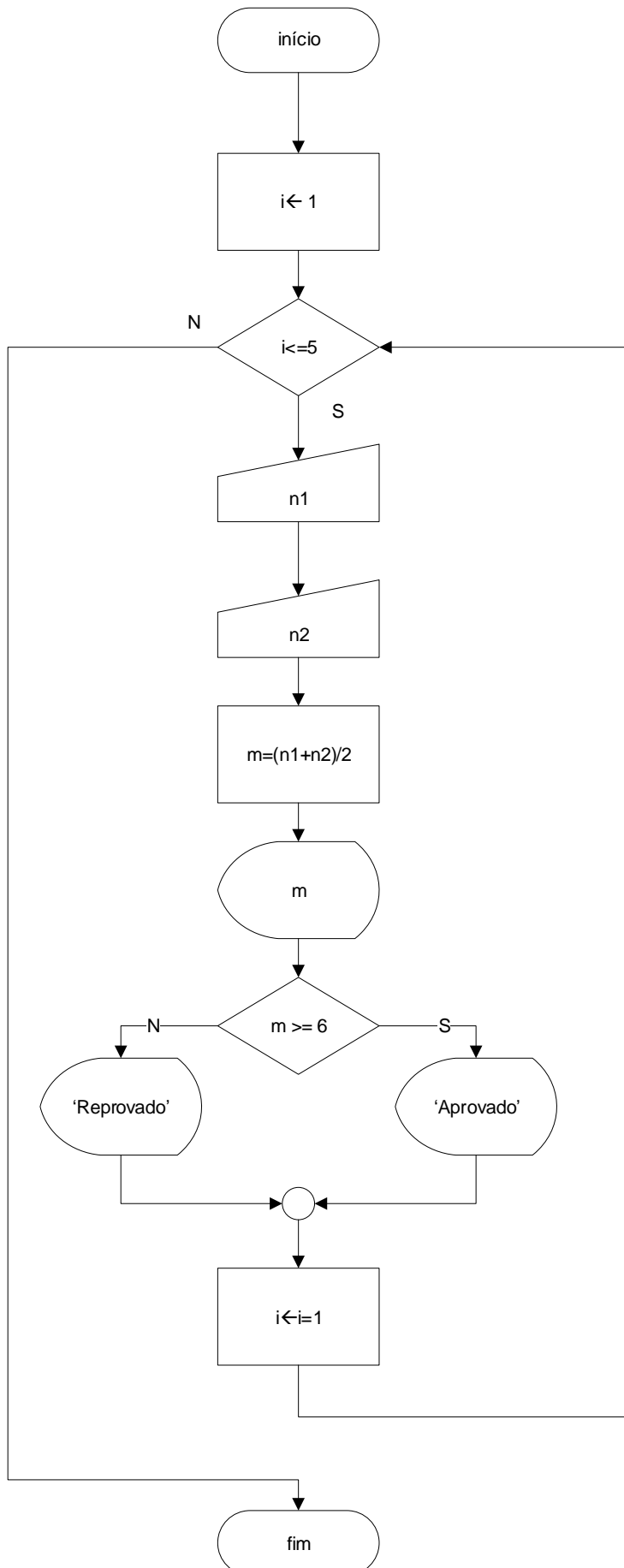
Nesse tipo de estrutura, é realizado um teste lógico no início do bloco. Caso a condição seja VERDADEIRA, o bloco de comandos associado à condição será executado. Quando a condição for FALSA, o bloco é abandonado, e a execução do programa seguirá seu fluxo.

Vejamos como ficaria o exemplo acima utilizando essa estrutura:

Algoritmo

- 1- Inicializar o contador ($i \leftarrow 1$)
- 2- Enquanto ($i \leq 5$) faça do passo 3 ao 11
 - 3- Leia a primeira nota do aluno (n_1)
 - 4- Leia a segunda nota do aluno (n_2)
 - 5- Calcule a sua média ($m \leftarrow (n_1 + n_2) / 2$)
 - 6- Apresente a sua média (m)
 - 7- Se ($m \geq 6$) então
 - 8- Escreva ('Aprovado')
 - 9- Senão
 - 10- Escreva ('Reprovado')
 - 11- Incrementar o contador ($i \leftarrow i + 1$)

Diagrama de blocos



Código em Português Estruturado

```
programa LACOINICIO
var
    i: inteiro
    n1, n2, m: real
inicio
    i ← 1
    enquanto ( i <= 5) faça
        leia (n1)
        leia (n2)
        m ← (n1+n2)/2
        escreva (m)
        se (m >= 6) então
            escreva ('Aprovado')
        senão
            escreva ('Reprovado')
        fim_se
        i ← i + 1
    fim_enquanto
fim
```

Código em Java

```
import java.util.*;
public class LacoInicio {
    public static void main(String args[]){
        Scanner in = new Scanner(System.in);
        int i=1;
        double n1, n2,m;
        while(i<=5){
            System.out.println("Digite a 1a nota do "+i+"º
aluno:");
            n1 = in.nextDouble();
            System.out.println("Digite a 2a nota do "+i+"º
aluno:");
            n2 = in.nextDouble();
            m = (n1+n2)/2;
            System.out.println("A média é: "+m);
            if (m>=6){
                System.out.println("Aprovado!");
            } else{
                System.out.println("Reprovado!");
            }
            i++;
        }
    }
}
```

Exercícios de fixação

1- Criar um programa que leia o salário de funcionário e calcule e apresente o desconto do INSS que será de:

- Caso o salário esteja abaixo de R\$ 2000,00 o desconto será de 8,5%;
- Caso o salário seja igual ou acima de R\$ 2000,00 o desconto será de 11%.

Repita o procedimento para os 4 funcionários da empresa.

2- Criar um programa que calcule e apresente a tabuada de um número inteiro digitado pelo usuário.

3- Criar um programa que leia o nome e a idade de 10 usuários. Informar o nome e a idade do mais novo e o nome e a idade do mais velho.

4- Criar um programa que leia o preço de um produto e a margem de lucro (em porcentagem). Calcule e apresente o preço de venda. Executar esse programa até que o usuário responda 'N' para a pergunta ('Deseja continuar a execução? (S/N) ').

LAÇO DE REPETIÇÃO COM TESTE NO FIM (FAÇA...ENQUANTO)

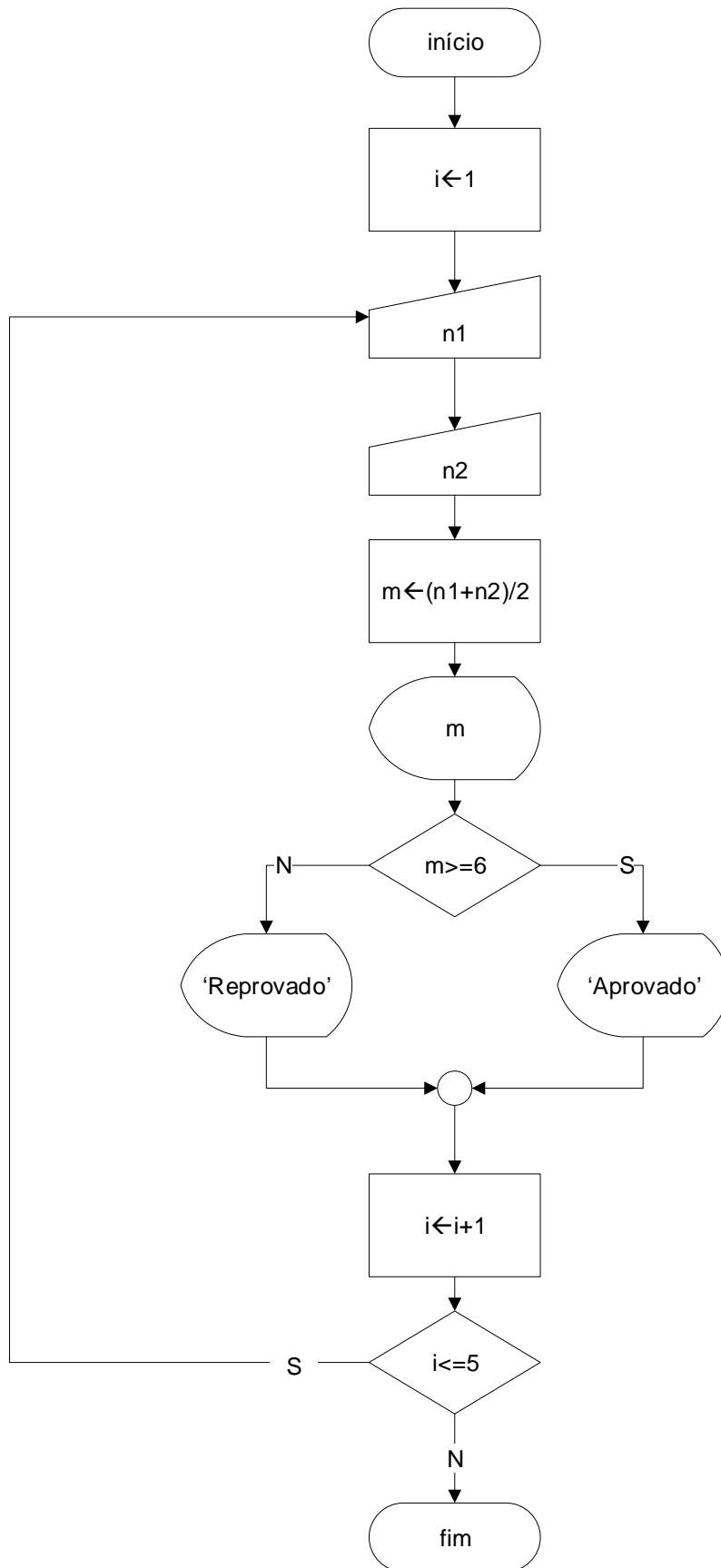
Nesse tipo de estrutura, o bloco de execução associado ao laço de repetição é executado, e ao final é realizado um teste lógico. Caso a condição seja VERDADEIRA, o bloco de comandos associado à condição será executado novamente. Enquanto a condição for VERDADEIRA, o bloco é repetido, e a execução do programa seguirá seu fluxo.

Vejamos como ficaria o exemplo anterior utilizando essa estrutura:

Algoritmo

- 1- Inicializar o contador ($i \leftarrow 1$)
- 2- Faça do passo 3 ao 12
 - 3- Leia a primeira nota do aluno ($n1$)
 - 4- Leia a segunda nota do aluno ($n2$)
 - 5- Calcule a sua média ($m \leftarrow (n1+n2)/2$)
 - 6- Apresente a sua média (m)
 - 7- Se ($m \geq 6$) então
 - 8- Escreva ('Aprovado')
 - 9-Senão
 - 10- Escreva ('Reprovado')
 - 11- Incrementar o contador ($i \leftarrow i + 1$)
 - 12- Enquanto ($i \leq 5$)

Diagrama de blocos



Código em Português Estruturado

```
programa LacoFim
var
    i: inteiro
    n1, n2, m: real
inicio
    i ← 1
    faça
        leia (n1)
        leia (n2)
        m ← (n1+n2)/2
        escreva (m)
        se (m >=6) então
            escreva('Aprovado')
        senão
            escreva('Reprovado')
        fim_se
        i ← i+1
    enquanto(i<=5)
fim.
```

Código em Java

```
import java.util.*;
public class LacoFim {
    public static void main(String args[]){
        Scanner in = new Scanner(System.in);
        int i=1;
        double n1, n2,m;
        do{
            System.out.println("Digite a 1a nota do "+i+"º aluno:");
            n1 = in.nextDouble();
            System.out.println("Digite a 2a nota do "+i+"º aluno:");
            n2 = in.nextDouble();
            m = (n1+n2)/2;
            System.out.println("A média é: "+m);
            if (m<6){
                System.out.println("Reprovado!");
            } else{
                System.out.println("Aprovado!");
            }
            i++;
        }while(i<=5);
    }
}
```

Atividade de fixação

Refaça os exercícios da página 36 utilizando o laço de repetição com teste no fim.

Exercício extra:

Criar um programa que apresente o fatorial de um número inteiro digitado pelo usuário.

Obs:

O fatorial de um número é obtido através do cálculo:

$$5! = 5 * 4 * 3 * 2 * 1 = 120$$

$$4! = 4 * 3 * 2 * 1 = 24$$

Utilizar um dos dois laços de repetição para obter o cálculo.

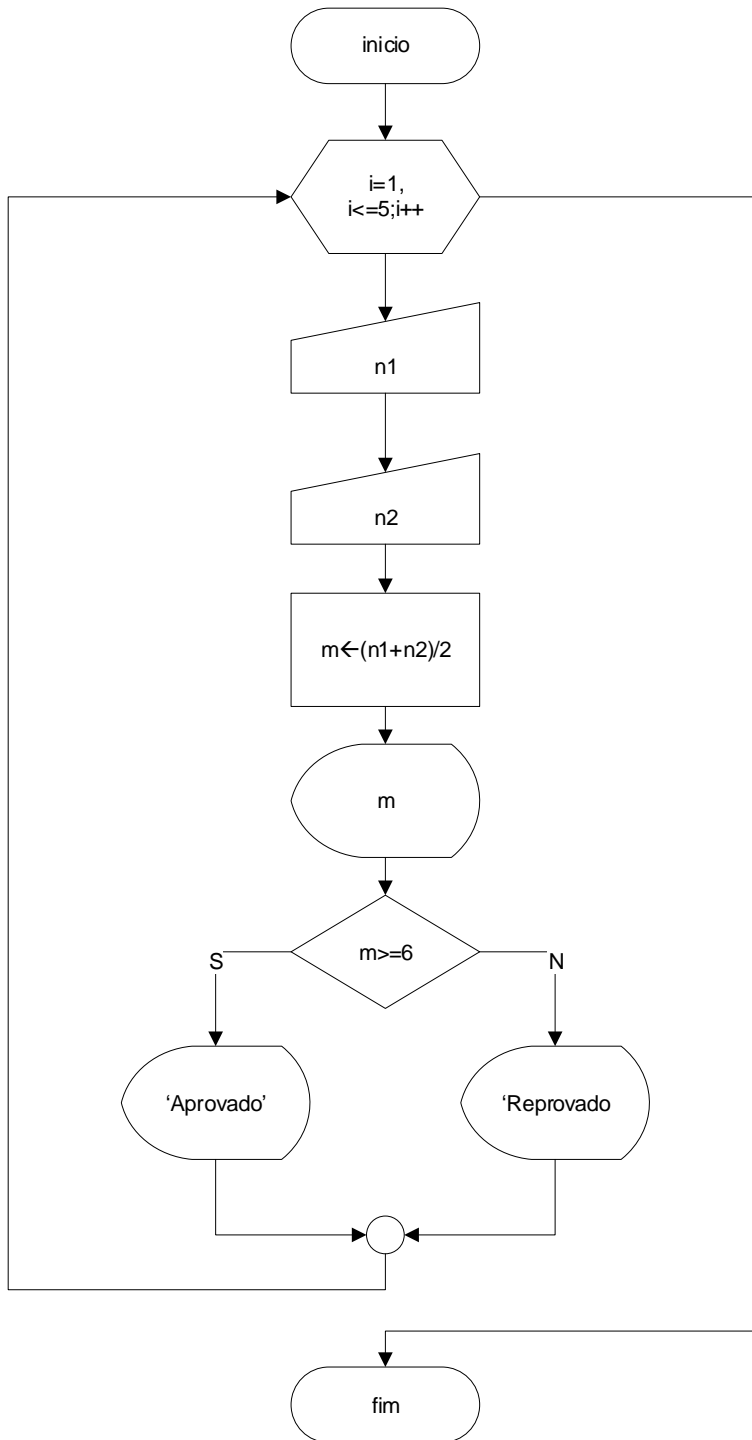
Laço de repetição com variável de controle (para...até...passo...faça / for...to ...do)

Nesse tipo de estrutura, no início do laço de repetição uma variável de controle (contador) é iniciada, incrementada e testada numa única instrução. O exemplo da página 32 (ler duas notas, calcular e apresentar a média e a situação – aprovado ou reprovado - de 5 alunos) escrito com esse tipo de laço ficaria assim:

Algoritmo

- 1- Para i=1 até 5 faça da instrução 2 a 9 passo 1
 - 2- Leia a primeira nota (n1)
 - 3- Leia a segunda nota (n2)
 - 4- Calcular a média ($m \leftarrow (n1+n2)/2$)
 - 5- Apresentar a média (m)
 - 6- Se ($m \geq 6$) então
 - 7- Escreva ('Aprovado')
 - 8- Senão
 - 9- Escreva ('Reprovado')

Diagrama de bloco



Português estruturado

programa LACOCONTROLE

var

i: inteiro

n1, n2, m: real

início

para i=1 até 5 passo 1 faça

leia (n1)

leia (n2)

$m \leftarrow (n1+n2)/2$

escreva(m)

se (m >=6) então

escreva ('Aprovado')

senão

escreva('Reprovado')

fim_se

fim_para

fim.

Código em JAVA

```
import java.util.*;
public class LacoInicio {
    public static void main(String args[]){
        Scanner in = new Scanner(System.in);
        int i;
        double n1, n2,m;
        for (i=1;i<=5;i++){
            System.out.println("Digite a 1a nota do "+i+"º aluno:");
            n1 = in.nextDouble();
            System.out.println("Digite a 2a nota do "+i+"º aluno:");
            n2 = in.nextDouble();
            m = (n1+n2)/2;
            System.out.println("A média é: "+m);
            if (m<6){
                System.out.println("Reprovado!");
            } else{
                System.out.println("Aprovado!");
            }
        }
    }
}
```

Exercícios:

Refazer os exercícios 1, 2, 3, 5 e 6 de laços de repetição utilizando o laço para. Testar os exercícios no Java para eliminar os erros.

CAPÍTULO 7

VETORES

Vetores são estruturas de dados que armazenam usualmente uma quantidade fixa de dados de um certo tipo.

Internamente, um vetor armazena diversos valores, cada um associado a um número que se refere à posição de armazenamento, e é conhecido como índice. Os vetores são estruturas indexadas, em que cada valor que pode ser armazenado em uma certa posição (índice) é chamado de elemento do vetor.

Cada elemento do vetor pode ser utilizado individualmente de forma direta, ou seja, pode ser lido ou escrito diretamente, sem nenhuma regra ou ordem preestabelecida, fazendo dos vetores estruturas de dados de acesso aleatório. O número de posições de um vetor corresponde ao tamanho que ele tem; assim, um vetor de tamanho 10 tem esse número de elementos, isto é, pode armazenar até dez elementos distintos.

O **Java** como as linguagens C e C++ são linguagens com vetores zero-based, isto é, as posições do vetor iniciam a numeração a partir do valor 0, portanto, um vetor de tamanho 10 teria índices iniciados em 0 prosseguindo até o 9.

Exemplo de declaração, leitura e apresentação de um vetor em Java:

```
import java.util.Scanner;
public class LeCalculaVetor {
    public static void main(String[] args) {
        Scanner ler = new Scanner(System.in);
        final int TAM = 10; //definindo uma constante de tamanho 10
        int i, a[], b[];

        a = new int[TAM];
        b = new int[TAM];
        //leitura do vetor a
        for (i=0; i<TAM; i++) {
            System.out.println("Entre com o " +(i+1)+ "o. valor");
            a[i] = ler.nextInt();
            b[i] = a[i];
        }

        //apresentação do vetor b

        for (i=0; i<TAM; i++) {
            System.out.println("O " +(i+1)+ "o. valor de b eh: " +b[i]);
        }
    }
}
```


}

Exercícios

Criar o código em Java dos exercícios abaixo:

1. Criar um programa que leia um vetor A com 10 posições de inteiro e imprima na tela um vetor b sendo que cada elemento de B seja o quadrado de A
2. Ler um vetor A de 10 posições de inteiro, ler um vetor B com 10 posições de inteiro e apresentar o um vetor C que será a soma de A com B
3. Criar um vetor A que leia 10 posições de inteiro. Imprimir um vetor B que será o vetor A invertido (ou seja a ultima posição de A será a 1ª de B, a penúltima de A será a 2ª de B e assim sucessivamente)
4. Criar um vetor A que leia 10 valores inteiros. Calcule e apresente a média dos dez valores lidos.