INTRODUÇÃO À PROGRAMAÇÃO

Prof. Me. Wallace Rodrigues de Santana

www.neutronica.com.br

Versão 1.5 Preliminar

Atribuição-NãoComercial-CompartilhaIgual 3.0 Brasil (CC BY-NC-SA 3.0)

Você tem a liberdade de:

Compartilhar — copiar, distribuir e transmitir a obra.

Remixar — criar obras derivadas.

Sob as seguintes condições:

Atribuição — Você deve creditar a obra da forma especificada pelo autor ou licenciante (mas não de maneira que sugira que estes concedem qualquer aval a você ou ao seu uso da obra).

Uso não comercial — Você não pode usar esta obra para fins comerciais.

Compartilhamento pela mesma licença — Se você alterar, transformar ou criar em cima desta obra, você poderá distribuir a obra resultante apenas sob a mesma licença, ou sob uma licença similar à presente.

Ficando claro que:

Renúncia — Qualquer das condições acima pode ser **renunciada** se você obtiver permissão do titular dos direitos autorais.

Domínio Público — Onde a obra ou qualquer de seus elementos estiver em **domínio público** sob o direito aplicável, esta condição não é, de maneira alguma, afetada pela licença.

Outros Direitos — Os seguintes direitos não são, de maneira alguma, afetados pela licença:

- Limitações e exceções aos direitos autorais ou quaisquer **usos livres** aplicáveis;
- Os **direitos morais** do autor;
- Direitos que outras pessoas podem ter sobre a obra ou sobre a utilização da obra, tais como **direitos de imagem** ou privacidade.

Aviso — Para qualquer reutilização ou distribuição, você deve deixar claro a terceiros os termos da licença a que se encontra submetida esta obra. A melhor maneira de fazer isso é com um link para esta página.

Introdução à Programação

Apresentação da disciplina

Módulos

- Módulo 1 – Introdução à arquitetura de computadores
- Módulo 2 – Sistemas de numeração
- Módulo 3 – Tipos primitivos de dados
- Módulo 4 – Operadores aritméticos, relacionais e lógicos, precedência de operadores e expressões
- Módulo 5 – Identificadores, constantes, variáveis e comandos de atribuição
- Módulo 6 – Algoritmos, fluxogramas e codificação
- Módulo 7 – Estruturas de decisão
- Módulo 8 – Estruturas de repetição
- Módulo 9 – Vetores e matrizes

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Ementa

- Conceitos básicos sobre programação, com foco na arquitetura básica do computador; sistema binário; operadores aritméticos, lógicos e relacionais; precedência de operadores; expressões; comandos de atribuição e constantes e variáveis;
- Conceitos básicos sobre lógica de programação, com foco em fluxogramas, algoritmos e estruturas de decisão;
- Estruturas de repetição;
- Vetores e matrizes.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Objetivo Geral

Apresentar ao aluno os conceitos de programação em seus aspectos teórico e prático, bem como familiarizá-lo com as estratégias de aplicação e uso nas organizações das tecnologias de informação.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Referências

BÁSICAS

ALCADE, Eduardo; GARCIA, Miguel; PEÑUELAS, Salvador; **Informática Básica**. Makron Books, São Paulo, 1991.

FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico, **Lógica de Programação**. Makron Books, São Paulo, 2000.

VELLOSO, Fernando de Castro; **Informática: conceitos básicos**. Editora Campus, Rio de Janeiro, 1999.

WEBER, Raul Fernando; **Arquitetura de Computadores Pessoais**. Editora Sagra Luzzatto, Porto Alegre, 2000.

WIRTH, Niklaus; ***Algoritmos e Estruturas de Dados***. Editora LTC, Rio de Janeiro, 1989.

COMPLEMENTARES

ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi ; **Fundamentos da programação de computadores**. Editora Pearson, São Paulo, 2012.

MEIRELLES, Fernando de Souza; **Informática, novas aplicações com microcomputadores**, Makron Books, São Paulo, 1994.

SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madsen; **Algoritmos**. Makron Books, São Paulo, 1998.

XAVIER, Gley Fabiano Cardoso; **Lógica de programação**. Editora SENAC, São Paulo, 2007.

Sistemática de Trabalho

- Aulas expositivas em sala de aula;
- Aulas no laboratório de informática;
- Listas de exercícios;
- Atividades;
- Avaliações.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Critérios de Avaliação

No decorrer de cada unidade são aplicadas atividades individuais, que devem ser entregues nas datas determinadas. Se entregues após esta data mas antes da data de aplicação da avaliação, a mesma valerá metade dos pontos.

Para compor as notas N1 e N2, faz-se a soma da atividade que vale 3 (três) com a primeira avaliação que vale 7 (sete):

$$N1 = Atividade + Avaliação$$

$$N2 = Atividade + Avaliação$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Critérios de Avaliação

Ao final do semestre, será feita a média entre as notas N1 e N2, que deverá ser igual ou superior a 7 (sete) para que o aluno possa ser aprovado na disciplina sem a necessidade de realizar o exame final:

$$Média\ Final = \frac{N1 + N2}{2}$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Critérios de Avaliação

Caso o aluno não atinja Média Final igual ou superior a 7 (sete), mas tenha obtido ao menos Média Final igual ou superior a 3 (três), poderá fazer um exame ao final do semestre.

O Exame Final é uma avaliação individual e sem consulta que vale de 0 (zero) a 10 (dez), onde será cobrado o conteúdo de todo o semestre.

A Nota Final será então a soma da Média Final mais a Nota do Exame divididos por 2 (dois).

O aluno para ser aprovado na disciplina deverá obter então Nota Final igual ou superior a 5 (cinco).

$$Nota\ Final = \frac{Média\ Final + Nota\ do\ Exame}{2}$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Avaliações e exame

A avaliação é individual e sem consulta.

Datas previstas para entrega das atividades:

- Atividade 1: **verificar calendário acadêmico**
- Atividade 2: **verificar calendário acadêmico**

Datas previstas para aplicação das avaliações:

- Avaliação N1: **verificar calendário acadêmico**
- Avaliação N2: **verificar calendário acadêmico**

Data prevista para aplicação do exame:

- Exame: **verificar calendário acadêmico**

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Regra de Três Simples

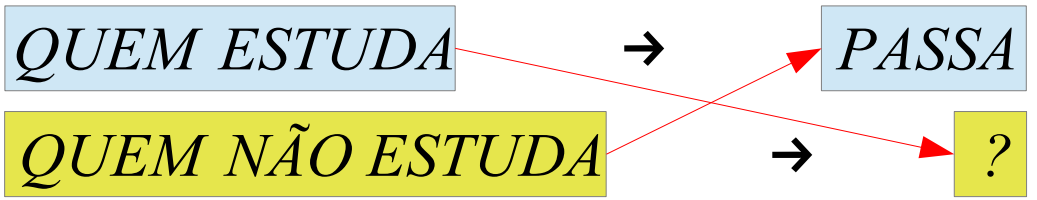
Não se esqueça:

QUEM ESTUDA	→	*PASSA*
QUEM NÃO ESTUDA	→	*?*

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Regra de Três Simples

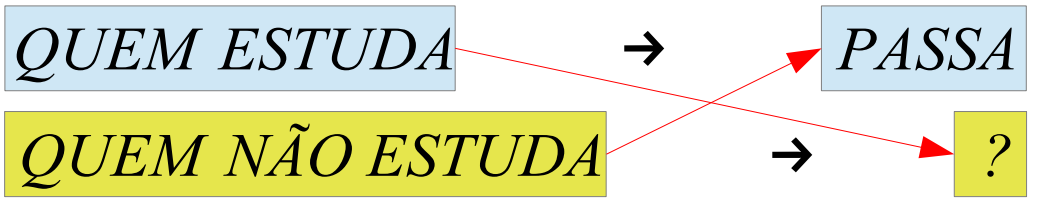
Não se esqueça:



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Regra de Três Simples

Não se esqueça:

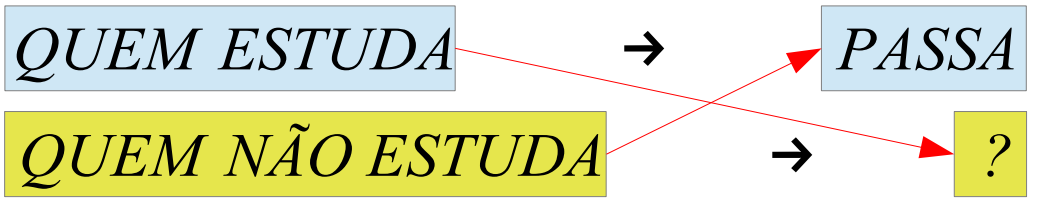


$$QUEM\ ESTUDA \quad x \quad ? \quad = \quad QUEM\ NÃO\ ESTUDA \quad x \quad PASSA$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Regra de Três Simples

Não se esqueça:



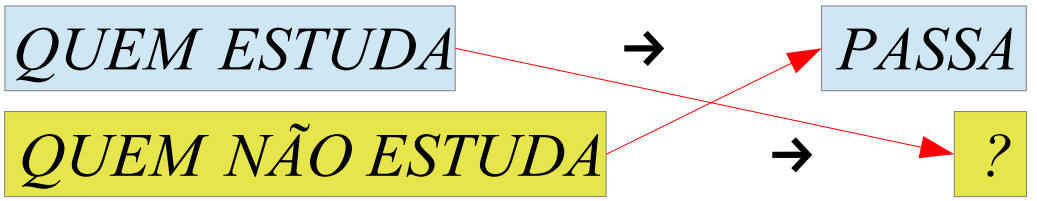
$$QUEM\ ESTUDA \quad x \quad ? \quad = \quad QUEM\ NÃO\ ESTUDA \quad x \quad PASSA$$

$$? = \frac{QUEM\ NÃO\ ESTUDA \quad x \quad PASSA}{QUEM\ ESTUDA}$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Regra de Três Simples

Não se esqueça:



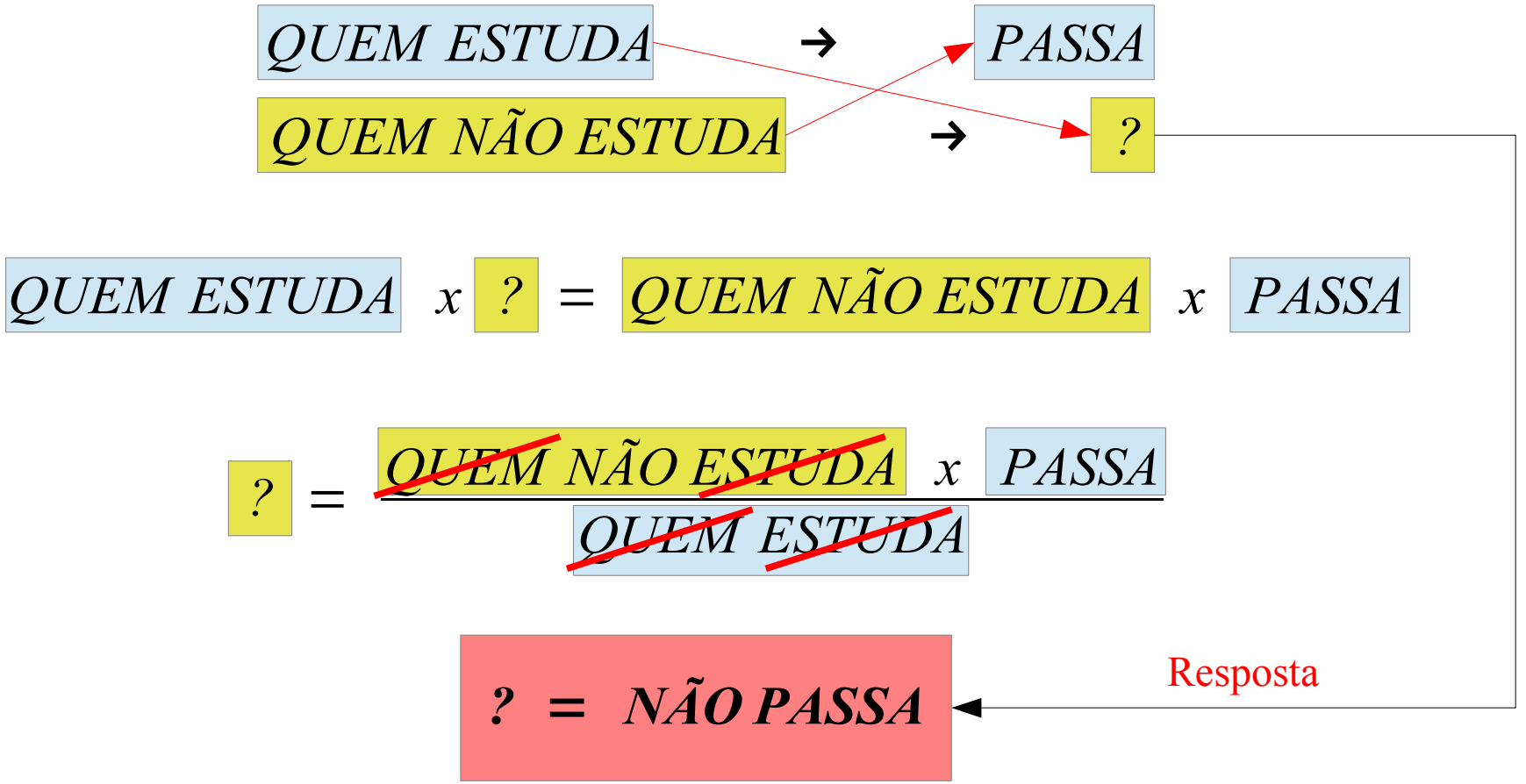
$$QUEM\ ESTUDA \quad x \quad ? \quad = \quad QUEM\ NÃO\ ESTUDA \quad x \quad PASSA$$

$$? = \frac{QUEM\ NÃO\ ESTUDA \quad x \quad PASSA}{QUEM\ ESTUDA}$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Regra de Três Simples

Não se esqueça:



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Módulo 1

Introdução à arquitetura de computadores

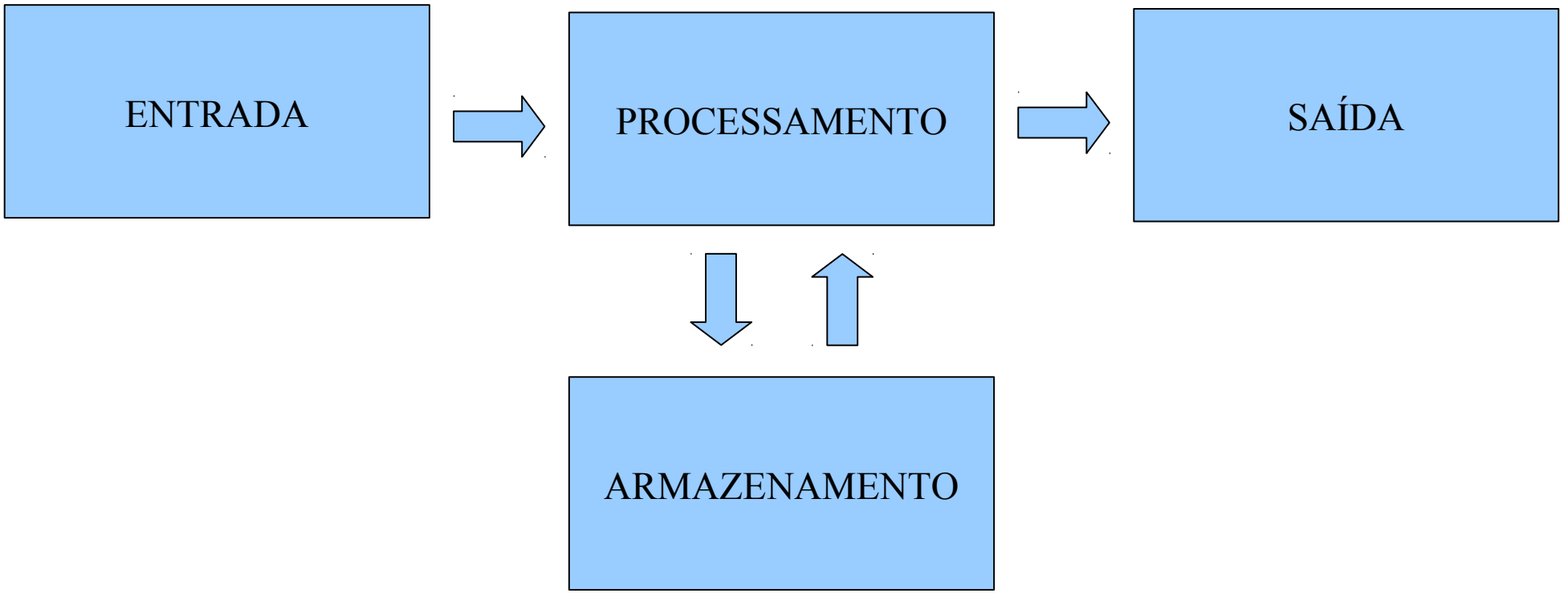
O computador

O computador é uma máquina que basicamente manipula números.

Uma determinada sequência de números pode representar uma música, uma imagem, um vídeo ou qualquer outro tipo de informação.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Organização do computador



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Unidade de entrada

A unidade de entrada é composta por elementos ou componentes que permitem que o computador seja alimentado com algum tipo de informação, instrução ou comando.

Componentes típicos da unidade de entrada são o teclado e o *mouse*.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Unidade de processamento

A unidade de processamento é formada pelo processador, pela memória RAM e pelos canais de comunicação com as outras unidades.

O processador é responsável pelos cálculos, comparações e tomada de decisões.

A memória RAM (*Random Access Memory*), também conhecida como memória principal, é responsável por armazenar as informações que estão sendo processados no momento.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Unidade de saída

A unidade de saída possui os meios pelos quais o computador informa ao usuário ou operador o resultado do processamento das informações.

Componentes típicos da unidade de saída são o monitor de vídeo e a impressora.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

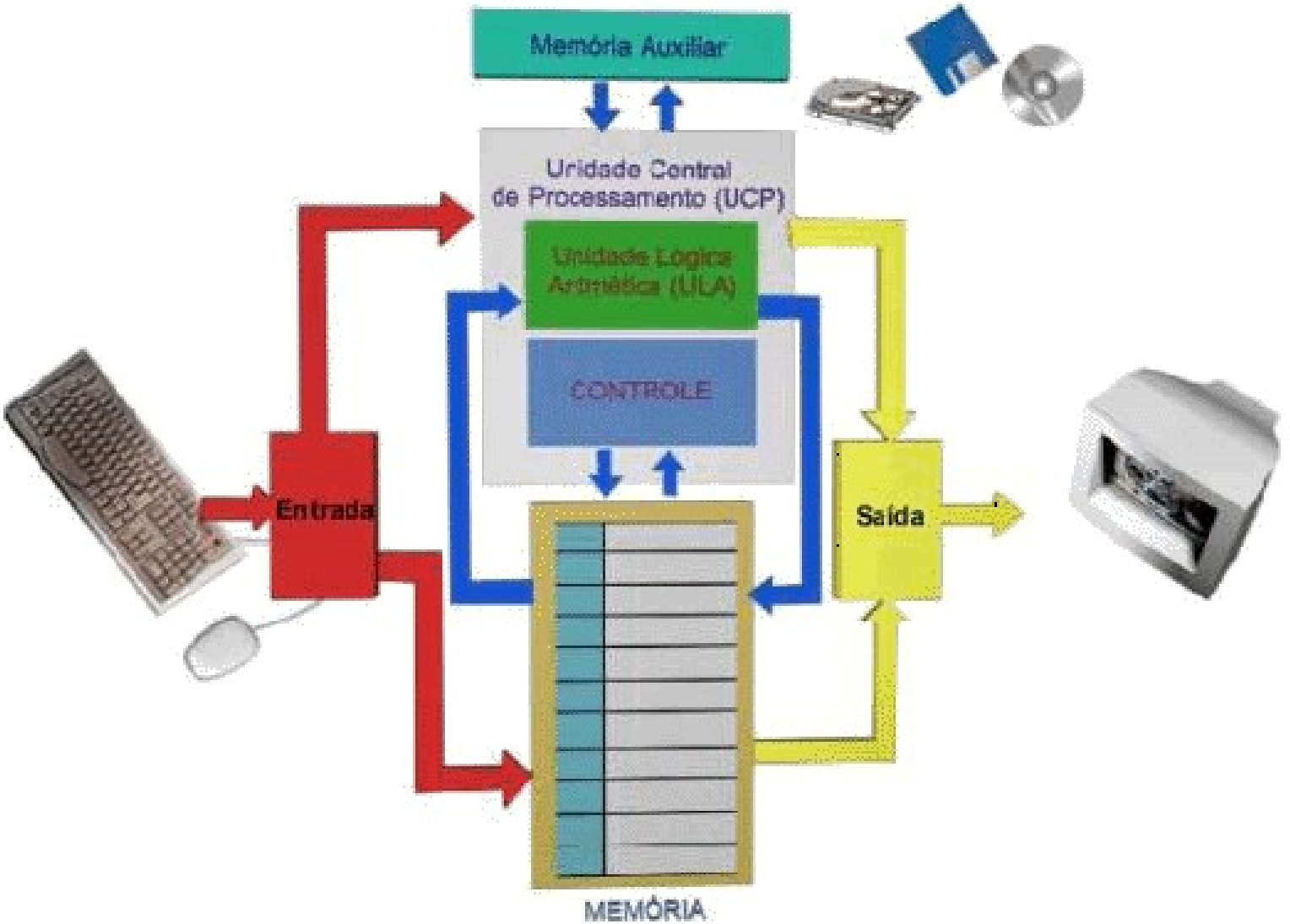
Unidade de armazenamento

A unidade de armazenamento, também conhecida como memória secundária ou memória auxiliar, é responsável por armazenar as informações que não estão sendo processados no momento.

Componentes típicos da unidade de armazenamento são os discos rígidos (HD), CD-ROM, DVD-ROM, *pendrives*, disquetes, etc.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Organização do computador



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Unidades de medida

Diferente dos seres humanos, que usam um sistema numérico baseado em dez símbolos (sistema decimal) para representar quantidades, o computador usa um sistema numérico baseado em dois símbolos, 0 e 1, que recebe o nome de sistema binário.

Um conjunto de 8 dígitos binários, ou bits, formam 1 byte, que é usado pelo computador para representar 1 caracter, que pode ser uma letra, um número, um símbolo gráfico, etc.

8 bits = 1 byte = 1 caracter

Assim, se quisermos armazenar o nome MARIA na memória de um computador, usaremos 5 bytes, pois o nome MARIA possui cinco caracteres.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Unidades de medida - múltiplos

Sendo então o byte a unidade de medida principal para armazenamento de dados em um computador, teremos os seguintes múltiplos de acordo com o Sistema Internacional de Unidades:

1.000 bytes = 1 kilobyte ou 1 kB
1.000 kilobytes = 1 megabyte ou 1 MB
1.000 megabytes = 1 gigabyte ou 1 GB
1.000 gigabytes = 1 terabyte ou 1 TB
1.000 terabytes = 1 petabyte ou 1 PB
1.000 petabytes = 1 exabyte ou 1 EB
1.000 exabytes = 1 zettabyte ou 1 ZB
1.000 zettabytes = 1 yottabyte ou 1 YB

Unidades de medida - exemplos

Capacidade de armazenamento de algumas mídias disponíveis no mercado:

Disquete 3 ½” = 1,44 MB

CD = 700 MB

DVD = de 4.700 a 8.500 MB, ou de 4,7 a 8,5 GB

Blu-ray = de 25.000 a 50.000 MB, ou de 25 GB a 50 GB

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Unidades de medida - observação

Os prefixos kilo (k), mega (M), giga (G) etc, são usados pelo Sistema Internacional de Unidades e são baseados no sistema decimal. Como o computador usa o sistema binário, estes prefixos não indicam a capacidade correta de armazenamento. Assim, a Comissão Eletrotécnica Internacional (IEC) estabeleceu os seguintes múltiplos:

1.024 bytes = 1 kibibyte ou 1 KiB
1.024 kibibytes = 1 mebibyte ou 1 MiB
1.024 mebibytes = 1 gibibyte ou 1 GiB
1.024 gibibytes = 1 tebibyte ou 1 TiB
1.024 tebibytes = 1 pebibyte ou 1 PiB
1.024 pebibytes = 1 exbibyte ou 1 EiB
1.024 exbibytes = 1 zebibyte ou 1 ZiB
1.024 zebibytes = 1 yobibyte ou 1 YiB

Unidades de medida – prefixo SI

Unidades de medida de acordo com o Sistema Internacional de Unidades:

Prefixo decimal - SI			
Nome	**Símbolo**	**Múltiplo**	**Quantidade de bytes**
byte	B	10^0	1
kilobyte	kB	10^3	1.000
megabyte	MB	10^6	1.000.000
gigabyte	GB	10^9	1.000.000.000
terabyte	TB	10^{12}	1.000.000.000.000
petabyte	PB	10^{15}	1.000.000.000.000.000
exabyte	EB	10^{18}	1.000.000.000.000.000.000
zettabyte	ZB	10^{21}	1.000.000.000.000.000.000.000
yottabyte	YB	10^{24}	1.000.000.000.000.000.000.000.000

Unidades de medida – prefixo IEC

Unidades de medida de acordo com a Comissão Eletrotécnica Internacional:

Prefixo binário - IEC			
Nome	**Símbolo**	**Múltiplo**	**Quantidade de bytes**
byte	B	2^{0}	1
kibibyte	KiB	2^{10}	1.024
mebibyte	MiB	2^{20}	1.048.576
gibibyte	GiB	2^{30}	1.073.741.824
tebibyte	TiB	2^{40}	1.099.511.627.776
pebibyte	PiB	2^{50}	1.125.899.906.842.620
exbibyte	EiB	2^{60}	1.152.921.504.606.850.000
zebibyte	ZiB	2^{70}	1.180.591.620.717.410.000.000
yobibyte	YiB	2^{80}	1.208.925.819.614.630.000.000.000

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Para saber mais...

… acesse o material online de Introdução ao Computador, da Universidade Federal de Campina Grande, Paraíba, Brasil.

Módulo 2

Sistemas de numeração

Introdução

Diferente dos seres humanos, que usam um sistema numérico baseado em dez símbolos (sistema decimal) para representar quantidades, o computador usa um sistema numérico baseado em dois símbolos, 0 e 1, que recebe o nome de sistema binário.

Além dos sistemas binário e decimal, existem ainda os sistemas octal, que usa oito símbolos, e o sistema hexadecimal, que usa dezesseis símbolos.

É importante saber que seja lá qual for o sistema numérico adotado, é possível representar qualquer quantidade por meio dele.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Sistemas de numeração

Sistema binário usa dois símbolos:

0 e 1

Sistema octal usa oito símbolos:

0, 1, 2, 3, 4, 5, 6 e 7

Sistema decimal usa dez símbolos:

0, 1, 2, 3, 4, 5, 6, 7, 8 e 9

Sistema hexadecimal usa dezesseis símbolos:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Sistemas de numeração - exemplos

Para representar uma quantidade de dois mil, trezentos e sessenta e quatro elementos, usamos a seguinte notação em cada um dos sistemas numéricos disponíveis:

Sistema binário: 100100111100_2 (12 dígitos)

Sistema octal: 4474_8 (4 dígitos)

Sistema decimal: 2364_{10} (4 dígitos)

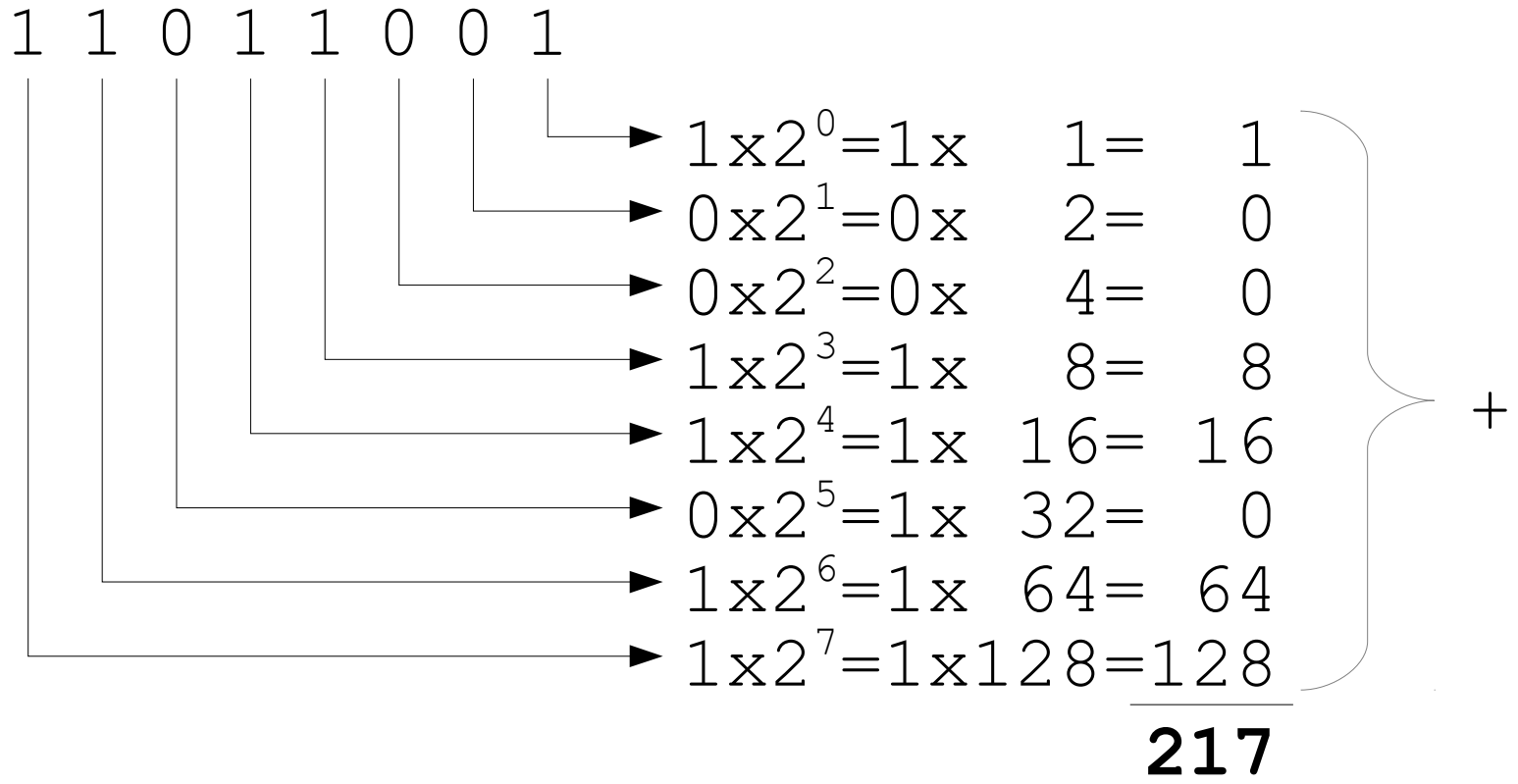
Sistema hexadecimal: $93C_{16}$ (3 dígitos)

A vantagem de se usar um sistema de numeração com mais símbolos é que o número fica com menos dígitos.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Convertendo de binário para decimal

Para converter um número de binário para decimal, basta seguir o esquema abaixo:

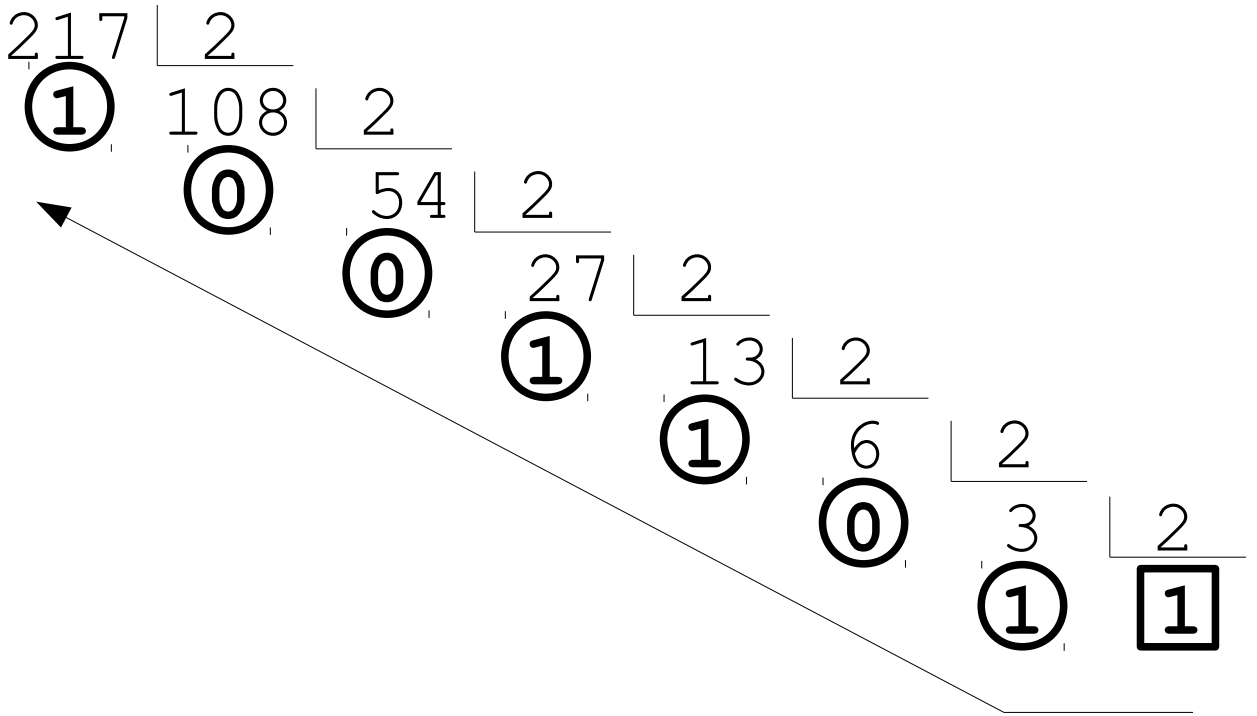


$$11011001_{2} = 217_{10}$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Convertendo de decimal para binário

Para converter um número de decimal para binário, basta fazer divisões sucessivas por 2 e ao final copiar o último quociente e o demais restos de divisão de baixo para cima:



$$217_{10} = 11011001_{2}$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Para saber mais...

… acesse o material online de Sistemas de Numeração, do Professor Rui Mano, da Pontificia Universidade Católica do Rio de Janeiro, Brasil.

… pratique o jogo online Cisco Binary Game, da Cisco.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Módulo 3

Tipos primitivos de dados

Tipos primitivos de dados

O computador pode manipular os mais variados tipos de informações, desde números e letras até imagens e sons.

Como o computador manipula somente números, todo e qualquer tipo de informação deve ser representado no formato numérico.

Assim, foram criados alguns tipos primitivos de dados, que são os seguintes:

- Numérico;
- Caracter ou alfanumérico;
- Lógico.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Dados numéricos

O tipo de dado numérico pode representar os dados do conjunto de números inteiros ou do conjunto de números reais (fracionários).

O tipo de dado inteiro pode ser usado para representar a idade de uma pessoa, a quantidade de itens em um estoque, etc.

O tipo de dado real pode ser usado para representar um valor monetário e números fracionários em geral.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Dados caracter ou alfanuméricos

O tipo de dado alfanumérico representa os dados dos conjuntos de números (0..9), alfabéticos (a..z e A..Z) e de caracteres especiais (!, @. #, $, %, ¨, &, *, (,), etc.).

Esse tipo de dado geralmente usa o padrão ASCII (*American Standard Code for Information Interchange*), que significa Código Padrão Americano para o Intercâmbio de Informação.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Dados lógicos

O tipo de dado lógico é usado para representar informações que só podem assumir os valores Verdadeiro ou Falso.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Tipos de dados

Para cada linguagem de programação, um tipo de dado pode ter nomes e tamanhos diferentes, como exemplificado na tabela abaixo para as linguagens Pascal, C++ e Java:

Tipo de Dado	Pascal			C++			Java		
	Nome	Bytes	Faixa	Nome	Bytes	Faixa	Nome	Bytes	Faixa
Caracter	*char*	*1*	*-*	*char*	*1*	*-*	*char*	*2*	*-*
Inteiro	*integer*	*2*	*-32768 a 32767*	*int*	*4*	*-2.147.483.648 a 2.147.483.647*	*int*	*4*	*-2.147.483.648 a 2.147.483.647*
Real	*real*	*6*	$2,9x10^{-39}$ a $1,7x10^{38}$	*float*	*4*	$3,4x10^{-38}$ a $3,4x10^{38}$	*float*	*4*	$3,4x10^{-38}$ a $3,4x10^{38}$
Lógico	*boolean*	*1*	*-*	*bool*	*1*	*-*	*boolean*	*1*	*-*

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Para saber mais...

… leia o Capítulo 2 do livro Lógica de Programação, de André Luiz Forbellone e Henri Frederico Eberspacher, disponível na biblioteca da faculdade.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Módulo 4

Operadores aritméticos, relacionais e lógicos, precedência de operadores e expressões

Operadores aritméticos

Os operadores aritméticos representam as operações básicas usadas em expressões matemáticas:

- Adição → +
- Subtração → -
- Multiplicação → *
- Divisão → /

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Outros operadores aritméticos

Além dos operadores aritméticos básicos, existem outros operadores, como os mostrados a seguir:

- Divisão Inteira → div
- Resto de Divisão → mod
- Potência → ^
- Raiz quadrada → sqrt()

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores aritméticos - exemplos

A seguir, alguns exemplos com os operadores aritméticos básicos:

- 9 + 2 = 11 → adição
- 9 - 2 = 7 → subtração
- 9 * 2 = 18 → multiplicação
- 9 / 2 = 4,5 → divisão exata

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Outros operadores aritméticos - exemplos

A seguir, alguns exemplos com outros operadores aritméticos:

- 9 div 2 = 4 → divisão inteira
- 9 mod 2 = 1 → resto de divisão
- 9 ^ 2 = 81 → potência
- sqrt(9) = 3 → raiz quadrada

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores aritméticos - ordem de precedência

A ordem de precedência indica quais partes da expressão serão calculadas antes das outras.

A primeira regra é que todas as expressões são avaliadas da esquerda para a direita.

Exemplo:

- 9 * 2 + 3 = 18 + 3 = 21 → correto
- 9 * 2 + 3 = 9 * 5 = 45 → errado

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores aritméticos - ordem de precedência

A segunda regra é que multiplicação e divisão são calculadas antes da adição e subtração.

Exemplo:

- 9 + 2 * 3 = 9 + 6 = 15 → correto
- 9 + 2 * 3 = 11 * 3 = 33 → errado

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores aritméticos - ordem de precedência

Para alterar a ordem de precedência, usa-se os parênteses.

Exemplo:

- (9 + 2) * 3 = 11 * 3 = 33 → correto
- 9 + 2 * 3 = 11 * 3 = 33 → errado

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores aritméticos - ordem de precedência

Assim, a ordem de precedência para uma expressão matemática usando operadores aritméticos é a seguinte:

- Parênteses;
- Potência e raiz quadrada;
- Multiplicação, divisão, divisão inteira e resto de divisão;
- Adição e subtração.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores relacionais

Operadores relacionais servem para fazer comparações lógicas e sempre retornam como resultado Verdadeiro ou Falso. São os seguintes os operadores relacionais:

- == → igual
- != ou <> → diferente
- > → maior que
- < → menor que
- >= → maior ou igual que
- <= → menor ou igual que

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores relacionais - exemplos

Segue abaixo alguns exemplos da aplicação dos operadores relacionais:

- 3 == 3 → verdadeiro
- 5 == 3 → falso
- 5 != 2 → verdadeiro
- 7 > 7 → falso
- 9 >= 5 → verdadeiro
- 3 <= 2 → falso

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores lógicos

Os operadores lógicos usam as regras da álgebra *booleana*. Os operadores lógicos básicos são os seguintes:

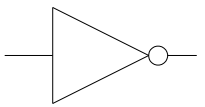
- !, NOT ou NÃO (negação)
- &, AND ou E (conjunção)
- |, OR ou OU (disjunção)
- ^, XOR ou XOU (disjunção exclusiva)

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores lógicos

A seguir, a tabela verdade para a operação lógica de negação “NÃO”:

Operação Lógica: !, NOT ou NÂO	
ENTRADA	**SAÌDA**
falso	***verdadeiro***
verdadeiro	***falso***

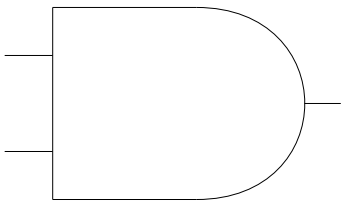


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores lógicos

A seguir, a tabela verdade para a operação lógica de conjunção “E”:

Operação Lógica: &, AND ou E		
ENTRADA 1	**ENTRADA 2**	**SAÍDA**
falso	*falso*	***falso***
falso	*verdadeiro*	***falso***
verdadeiro	*falso*	***falso***
verdadeiro	*verdadeiro*	***verdadeiro***

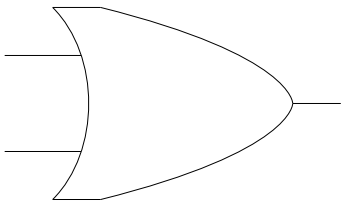


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores lógicos

A seguir, a tabela verdade para a operação lógica de disjunção “OU”:

Operação Lógica: |, OR ou OU		
ENTRADA 1	**ENTRADA 2**	**SAÍDA**
falso	*falso*	***falso***
falso	*verdadeiro*	***verdadeiro***
verdadeiro	*falso*	***verdadeiro***
verdadeiro	*verdadeiro*	***verdadeiro***

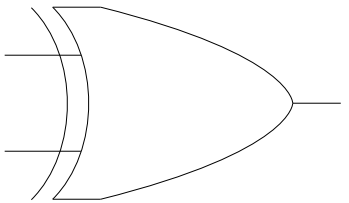


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Operadores lógicos

A seguir, a tabela verdade para a operação lógica de disjunção exclusiva “XOU”:

Operação Lógica: ^, XOR ou XOU		
ENTRADA 1	**ENTRADA 2**	**SAÍDA**
falso	*falso*	***falso***
falso	*verdadeiro*	***verdadeiro***
verdadeiro	*falso*	***verdadeiro***
verdadeiro	*verdadeiro*	***falso***



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Expressões - exemplos

Exemplos de expressões usando adição, subtração e divisão.

$$2+3-5 \quad \Rightarrow \quad 2+3-5$$

$$2+\frac{3}{5} \quad \Rightarrow \quad 2+3/5$$

$$\frac{2+3}{5} \quad \Rightarrow \quad (2+3)/5$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Expressões - exemplos

Exemplos de expressões usando potência.

$$3 \times 5 + 7^4 \quad \Rightarrow \quad 3 * 5 + 7 \wedge 4$$

$$3 \times (5 + 7)^4 \quad \Rightarrow \quad 3 * (5 + 7) \wedge 4$$

$$3 \times \left(\frac{5 + 7}{9} \right)^4 \quad \Rightarrow \quad 3 * ((5 + 7) / 9) \wedge 4$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Expressões - exemplos

Exemplos de expressões usando raiz quadrada.

$$4-\sqrt{\frac{3}{2}} \quad \Rightarrow \quad 4-sqrt(3/2)$$

$$4-\sqrt{\frac{3}{2^2}} \quad \Rightarrow \quad 4-sqrt(3/2\hat{}2)$$

$$\frac{2\sqrt{\frac{2}{3}}}{2^3-3} \quad \Rightarrow \quad 2*sqrt(2/3)/(2\hat{}3-3)$$

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Para saber mais...

… leia o Capítulo 2 do livro Lógica de Programação, de André Luiz Forbellone e Henri Frederico Eberspacher, disponível na biblioteca da faculdade;

… leia a nota de Aula sobre Expressões e Operadores Aritméticos, Lógicos e Relacionais, da Professora Maria das Graças da Silva Teixeira, da Universidade Federal do Espírito Santo, Espírito Santo, Brasil.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Módulo 5

Identificadores, constantes, variáveis e comandos de atribuição

Constantes e variáveis

Constantes e variáveis são espaços na memória que são nomeados por meio de identificadores e servem para armazenar dados que serão usados pelo computador.

A diferença entre uma e outra é que a constante mantem o seu valor inalterado durante toda a execução do programa, enquanto que a variável pode ter o seu valor alterado no decorrer da execução do programa.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Identificadores

Identificadores são os nomes atribuídos a constantes e variáveis e que servem para identificá-los em um programa.

Um identificador válido deve seguir as seguintes regras de nomeação:

- deve ser iniciado por um caracter alfabético, ou seja, uma letra válida do alfabeto;
- o segundo caracter em diante pode ser uma letra ou um número;
- não podem conter espaços em branco;
- não podem conter caracteres especiais, como !, @, #, $, %, ¨, &, *, (,), etc., com exceção do caracter “_” (*underline*);
- não podem ser usados nomes reservados da linguagem.

Identificadores - exemplos

Identificadores válidos:

salario
idade
nota1
EnderecoDestino
nome_func
imp_renda_fonte
RG_Func

Identificadores não válidos:

13ºsalario
@idade
nota(1)
Endereco Destino
nome:func
imp renda fonte
RG->Func

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Identificadores – *case sensitive*

Em linguagens de programação como Pascal, por exemplo, identificadores com nomes **`DescontoIR`** e **`descontoir`** referem-se à mesma constante ou variável;

Já em liguagens de programação como C++ e Java, por exemplo, identificadores com nomes **`DescontoIR`** e **`descontoir`** referem-se a constantes ou variáveis diferentes.

Linguagens de programação que fazem diferenciação entre identificadores escritos com letras maiúsculas e minúsculas são conhecidos como "*case sensitive*".

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Declarando constantes e variáveis

Na maioria das linguagens de programação, para se declarar uma constante ou variável, segue-se o seguinte esquema:

```
identificador: tipo de dado
```

Exemplos:

```
Idade: inteiro

SalarioFunc: real
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Tipos de dados

Para cada linguagem de programação, um tipo de dado pode ter nomes e tamanhos diferentes, como exemplificado na tabela abaixo para as linguagens Pascal, C++ e Java:

Tipo de Dado	Pascal			C++			Java		
	Nome	Bytes	Faixa	Nome	Bytes	Faixa	Nome	Bytes	Faixa
Caracter	*char*	*1*	*-*	*char*	*1*	*-*	*char*	*2*	*-*
Inteiro	*integer*	*2*	*-32768 a 32767*	*int*	*4*	*-2.147.483.648 a 2.147.483.647*	*int*	*4*	*-2.147.483.648 a 2.147.483.647*
Real	*real*	*6*	$2,9x10^{-39}$ a $1,7x10^{38}$	*float*	*4*	$3,4x10^{-38}$ a $3,4x10^{38}$	*float*	*4*	$3,4x10^{-38}$ a $3,4x10^{38}$
Lógico	*boolean*	*1*	*-*	*bool*	*1*	*-*	*boolean*	*1*	*-*

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Declarando constantes e variáveis

Exemplos em Pascal:

```
Idade: integer;
SalarioFunc: real;
```

Exemplos em C++:

```
int Idade;
float SalarioFunc;
```

Exemplos em Java:

```
int Idade;
float SalarioFunc;
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Comandos de atribuição

Na maioria das linguagens de programação, para se atribuir um valor a uma constante ou variável, segue-se o seguinte esquema:

```
identificador <- valor
```

Exemplos:

```
Idade <- 25

SalarioFunc <- 1345,90
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Comandos de atribuição

Exemplos em Pascal:

```
Idade := 25;
SalarioFunc := 1345.90;
```

Exemplos em C++:

```
Int = 25;
SalarioFunc = 1345.90;
```

Exemplos em Java:

```
Idade = 25;
SalarioFunc = 1345.90;
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Para saber mais...

… leia o Capítulo 2 do livro Lógica de Programação, de André Luiz Forbellone e Henri Frederico Eberspacher, disponível na biblioteca da faculdade.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Módulo 6

Algoritmos, fluxogramas e codificação

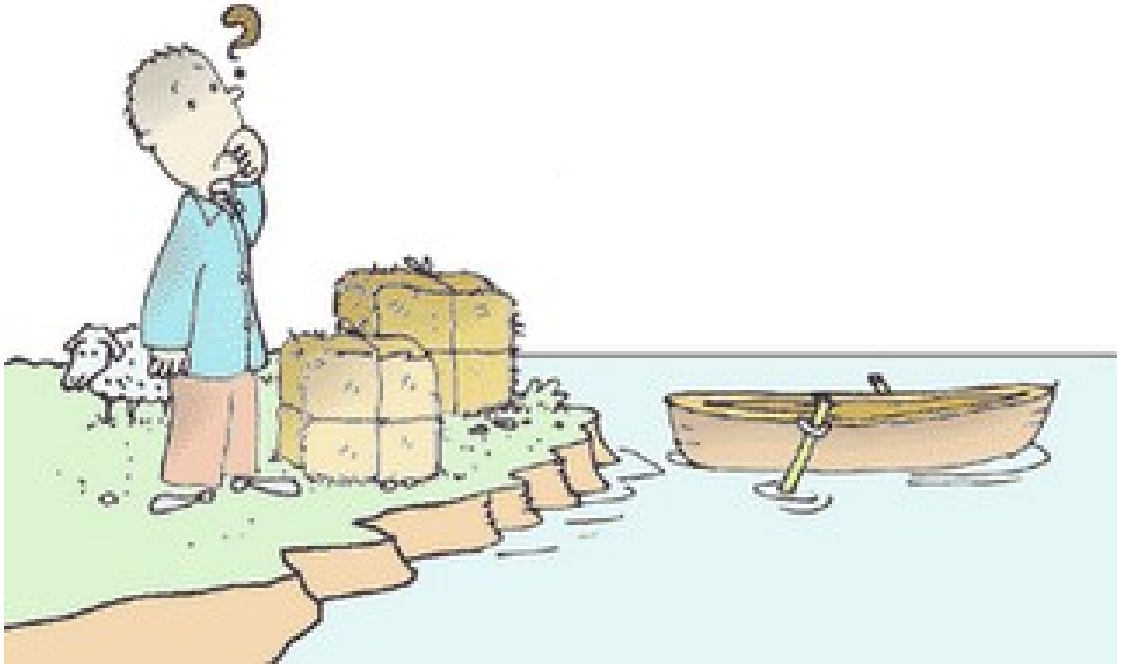
Algoritmo

É um conjunto finito de regras que fornece uma sequência de operações para resolver um problema específico.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Dilema do Canoeiro - Problema

Um canoeiro precisa transportar de uma margem a outra do rio uma onça, um bode e um feixe de capim. Mas como a canoa é frágil, ele só pode transportar um de cada vez. Se ele levar a onça primeiro, o bode come o capim. Se ele levar o capim, a onça come o bode. Qual deve ser a estratégia (algoritmo) usada pelo canoeiro para transportar a onça, o bode e o feixe de capim?



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

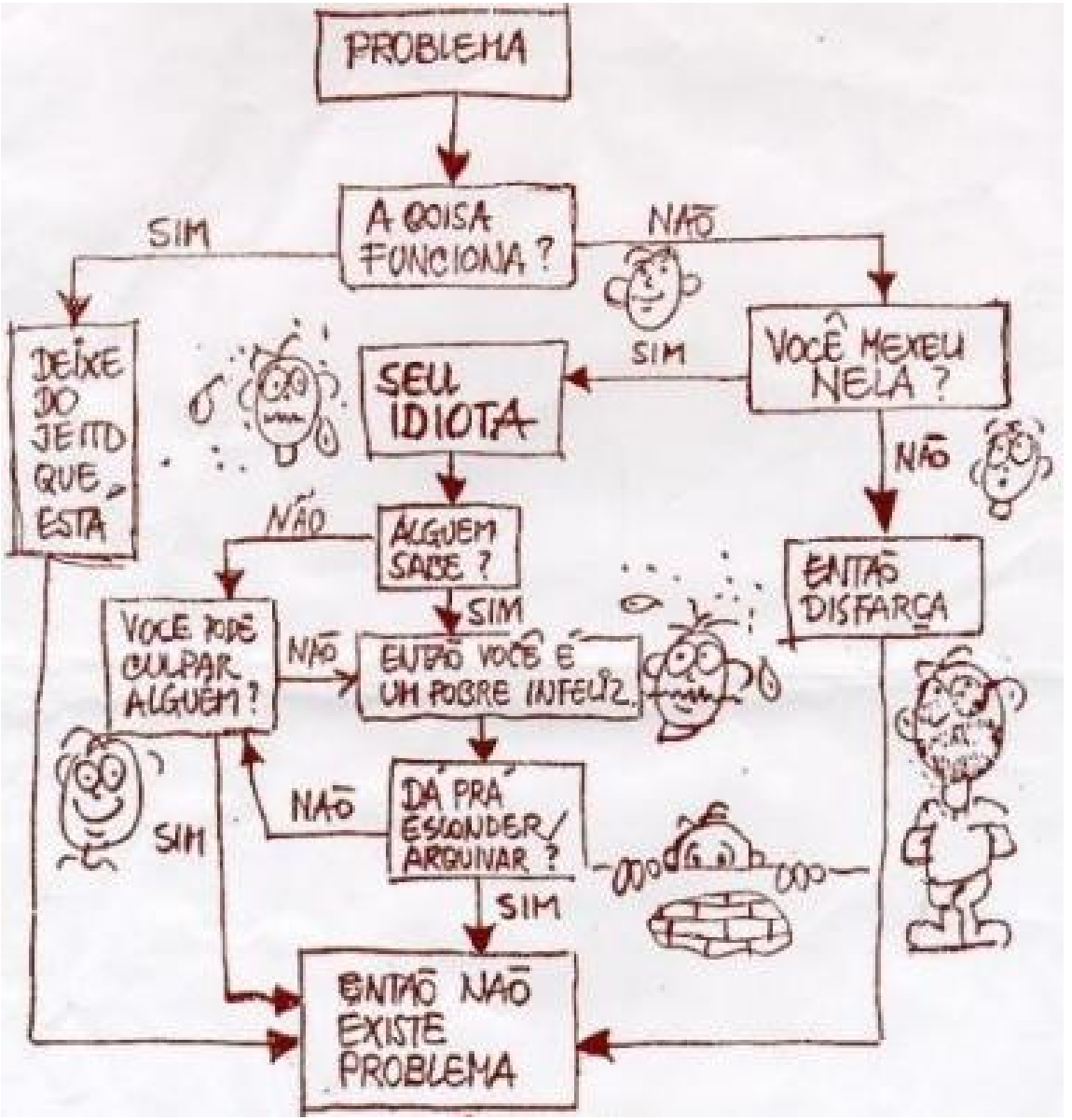
Dilema do Canoeiro - Solução

1. O canoeiro leva primeiro o bode;
2. O canoeiro volta vazio e leva o capim;
3. O canoeiro deixa o capim e retorna com o bode;
4. O canoeiro deixa o bode e leva a onça;
5. O canoeiro volta vazio para pegar o bode;
6. O canoeiro leva o bode novamente.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Fluxograma

É a representação gráfica de um algoritmo.



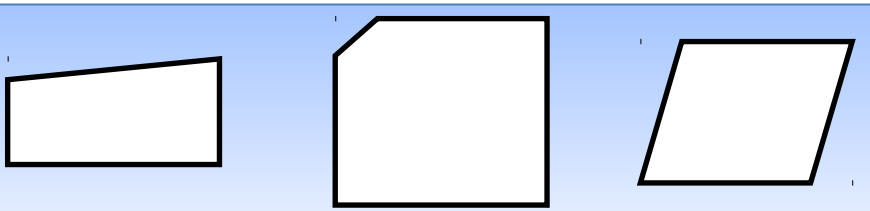
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Fluxograma - símbolos

TERMINAL

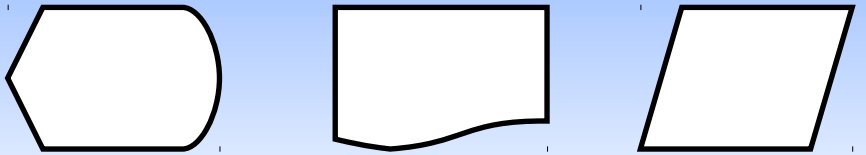
CONECTOR

ENTRADA DE DADOS



PROCESSAMENTO

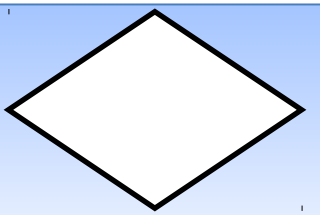
SAÍDA DE DADOS



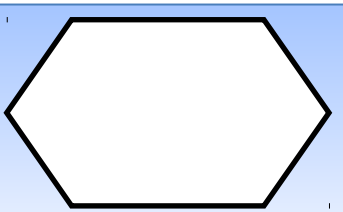
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Fluxograma - símbolos - continuação

DECISÃO



PREPARAÇÃO



SUB-PROGRAMA OU SUB-ROTINA

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação

Objetivo (ou problema a ser resolvido): determinar se um dado número é par ou ímpar.

Entrada: número a ser verificado.

Saída: se o dado número é par ou ímpar.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação - continuação

Solução: para determinar se um número é par ou ímpar, podemos dividi-lo por 2 e verificar o seu resto. Todo número dividido por 2 que apresenta resto 0 é par, e todo número dividido por 2 que apresenta resto 1 é ímpar.

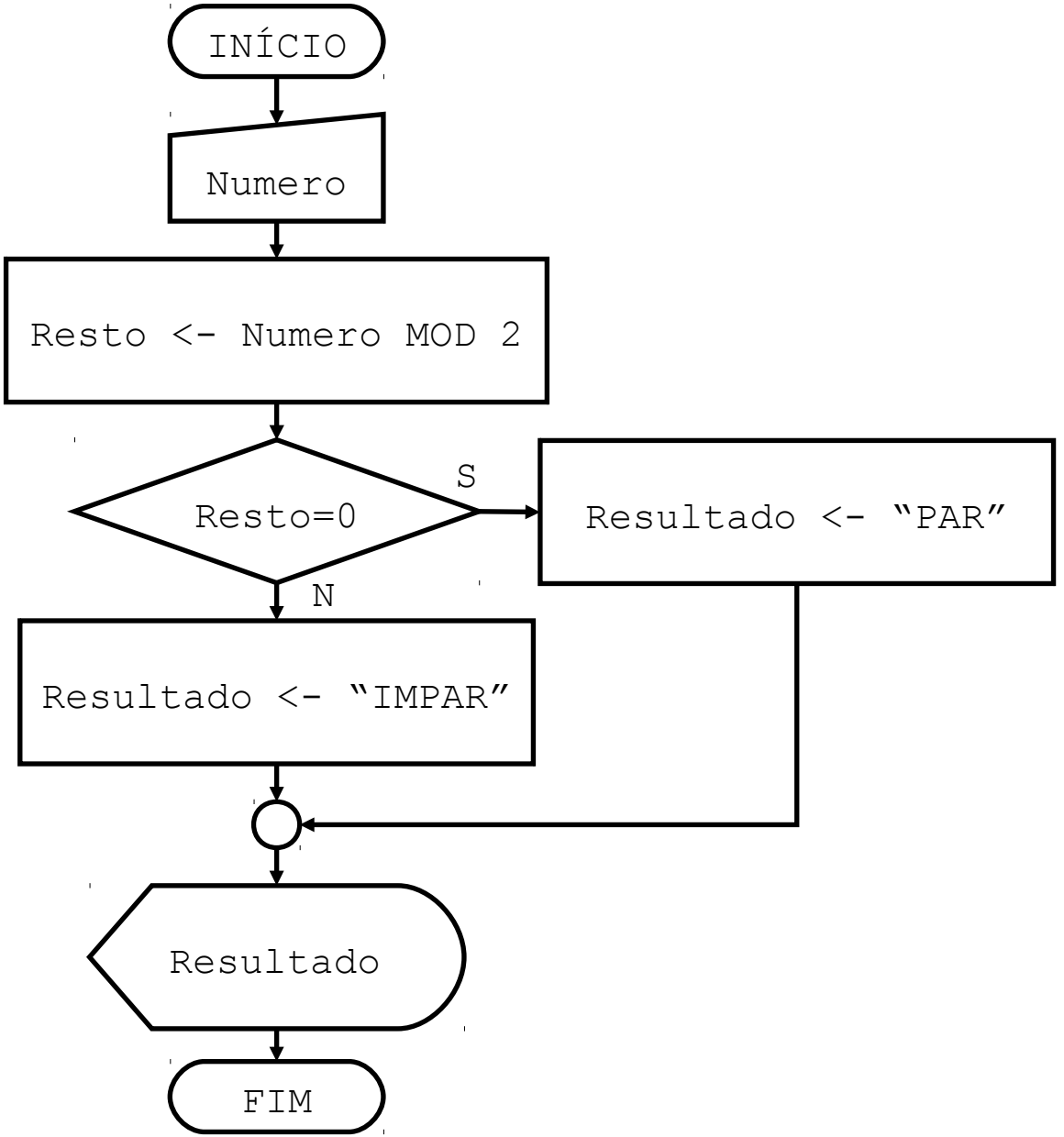
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação - algoritmo

1. Início;
2. Obter o número a ser verificado;
3. Fazer a operação de resto de divisão entre o número a ser verificado e o número 2;
4. Se o resto da divisão for igual a 0, o número é par;
5. Caso contrário, se o resto da divisão for igual a 1, o número é ímpar;
6. Mostrar o resultado;
7. Fim.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação - fluxograma



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação – código em Portugol

```
algoritmo "ParImpar"
var
   Numero, Resto: inteiro
   Resultado: caracter
inicio
   escreva("Digite um numero inteiro: ")
   leia(Numero)
   Resto <- Numero MOD 2
   se Resto=0 entao
      Resultado <- "PAR"
   senao
      Resultado <- "IMPAR"
   fimse
   escreva("O numero eh: ", Resultado)
fimalgoritmo
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação – código em Pascal

```
program ParImpar;
var
   Numero, Resto: integer;
   Resultado: string;
begin
   write('Digite um numero inteiro: ');
   read(Numero);
   Resto := Numero MOD 2;
   if Resto=0 then
      Resultado := 'PAR'
   else
      Resultado := 'IMPAR';
   write('O numero eh: ', Resultado);
end.
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação – código em C

```
#include <stdio.h>
int main(void)
{
   int Numero, Resto;
   char Resultado[6];
   printf("Digite um numero inteiro: ");
   scanf("%d", &Numero);
   Resto = Numero % 2;
   if (Resto==0)
      strcpy(Resultado, "PAR");
   else
      strcpy(Resultado, "IMPAR");
   printf("O numero eh: %s", Resultado);
   return 0;
}
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação – código em C++

```
#include <iostream>
#include <string>
using namespace std;
void main()
{
   int Numero, Resto;
   string Resultado;
   cout << "Digite um número inteiro: ";
   cin >> Numero;
   Resto = Numero % 2;
   if (Resto == 0)
      Resultado = "PAR";
   else
      Resultado = "IMPAR";
   cout << "O numero eh: " << Resultado << endl;
}
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação – código em C#

```
using System;
class ParImpar
{
   static void Main(string[] args)
   {
      int Numero, Resto;
      string Resultado;
      Console.Write("Digite um número inteiro: ");
      Numero = Convert.ToInt32(Console.ReadLine());
      Resto = Numero % 2;
      if (Resto == 0)
         Resultado = "PAR";
      else
         Resultado = "IMPAR";
      Console.WriteLine("O numero eh: {0}", Resultado);
   }
}
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação – código em VB.NET

```
Module ParImpar
    Sub Main()
        Dim Numero As nteger
        Dim Resto As Integer
        Dim Resultado As String
        Console.Write("Digite um número inteiro: ")
        Numero = Convert.ToInt32(Console.ReadLine())
        Resto = Numero Mod 2
        If Resto = 0 Then
            Resultado = "PAR"
        Else
            Resultado = "IMPAR"
        End If
        Console.WriteLine("O numero eh: {0}", Resultado)
    End Sub
End Module
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação – código em Java

```
import java.util.Scanner;
class ParImpar
{
   public static void main (String args[])
   {
      int Numero, Resto;
      String Resultado;
      System.out.println("Digite um numero inteiro: ");
      Numero = (new Scanner(System.in)).nextInt();
      Resto = Numero % 2;
      if (Resto == 0)
         Resultado = "PAR";
      else
         Resultado = "IMPAR";
      System.out.println("O numero eh: " + Resultado);
   }
}
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Para saber mais...

… leia a Tabela de Símbolos de Fluxograma, da BreezeTree Software.

… leia a Norma ECMA-4 Flow Charts 2nd Edition, da European Computer Manufacturers Association.

… leia o Manual IBM Flowcharting Techniques, da International Business Machines Corporation.

… leia a Apostila de Introdução a Algoritmos e Programação, dos Professores Fabricio Ferrari e Cristian Cechinel, da Universidade Federal do Pampa, Rio Grande do Sul, Brasil.

… leia a Apostila de Lógica de Programação, da Professora Flávia Pereira de Carvalho, da Faculdades Integradas de Taquara, Rio Grande do Sul, Brasil.

… leia a Apostila de Lógica de Programação, do Professor Paulo Sérgio de Moraes, da Universidade Estadual de Campinas, São Paulo, Brasil.

… leia a Apostila de Programação I, do Professor Edílson de Aguiar et al, da Universidade Federal do Espírito Santo, Espírito Santo, Brasil.

… leia a Apostila de Introdução à Linguagem Pascal, do Prof. Dalton Vinicius Kozak, da Pontifícia Universidade Católica do Paraná, Brasil.

Agradecimentos

Aos colegas:

•Prof. Me. Gabriel Paniz Patzer, pela ajuda com os códigos em C e Java;

•Prof. Esp. Cristhiano E. C. de Souza Hatanaka, pela ajuda com os códigos em C++, C# e VB.NET.

Módulo 7

Estruturas de decisão

Estruturas de decisão

Estruturas de decisão, também conhecidas como estruturas condicionais ou de seleção, são usadas para direcionar o fluxo de um programa e permitir que algumas ações sejam tomadas de acordo com determinadas condições.

Estruturas de decisão

As estruturas de decisão podem ser do tipo:

- Estrutura de decisão simples:

  se..entao ou if..then

- Estrutura de decisão composta:

  se..entao..senao ou if..then..else

- Estrutura de seleção múltipla:

  escolha..caso ou case..of

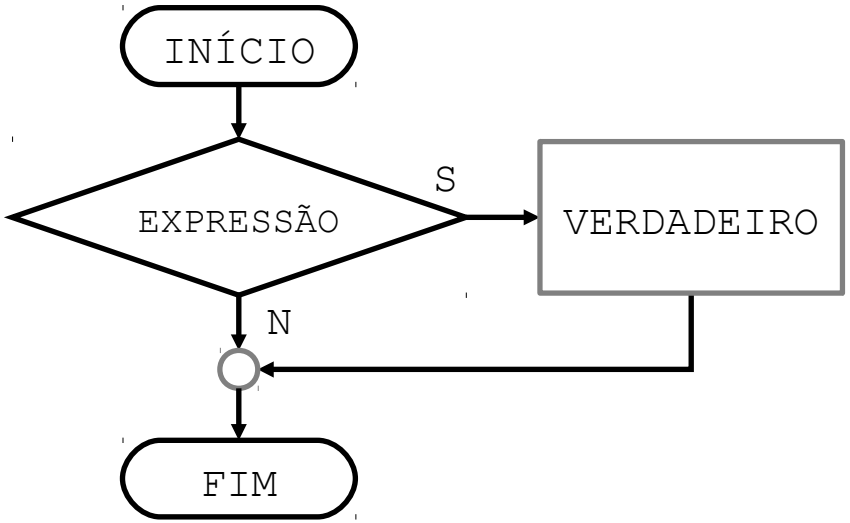
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estrutura de decisão simples

se..entao

Sintaxe:

```
se <expressão> entao
   <verdadeiro>
fimse
```



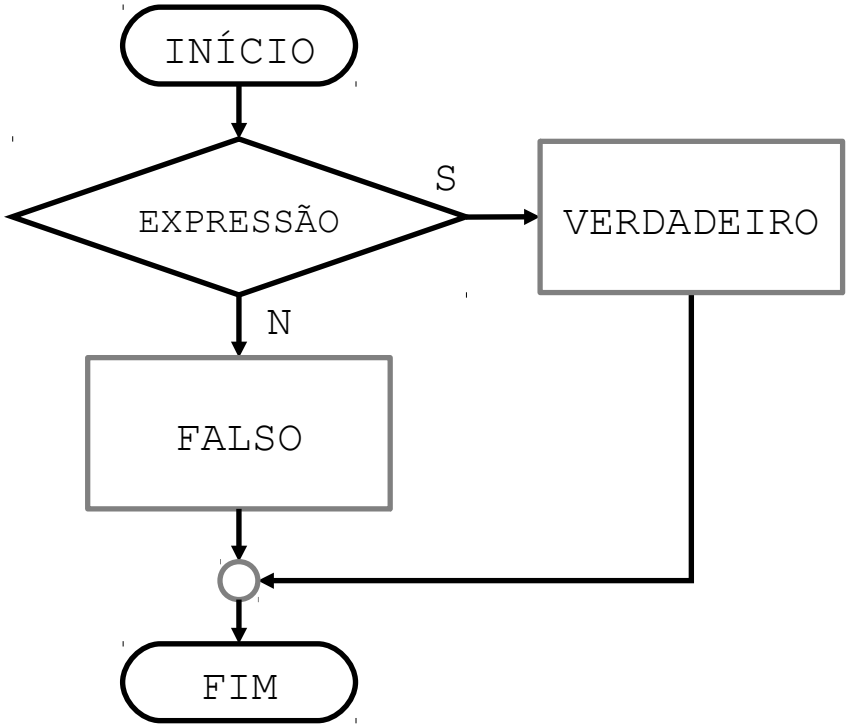
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estrutura de decisão composta

se..entao..senao

Sintaxe:

```
se <expressão> entao
   <verdadeiro>
senao
   <falso>
fimse
```



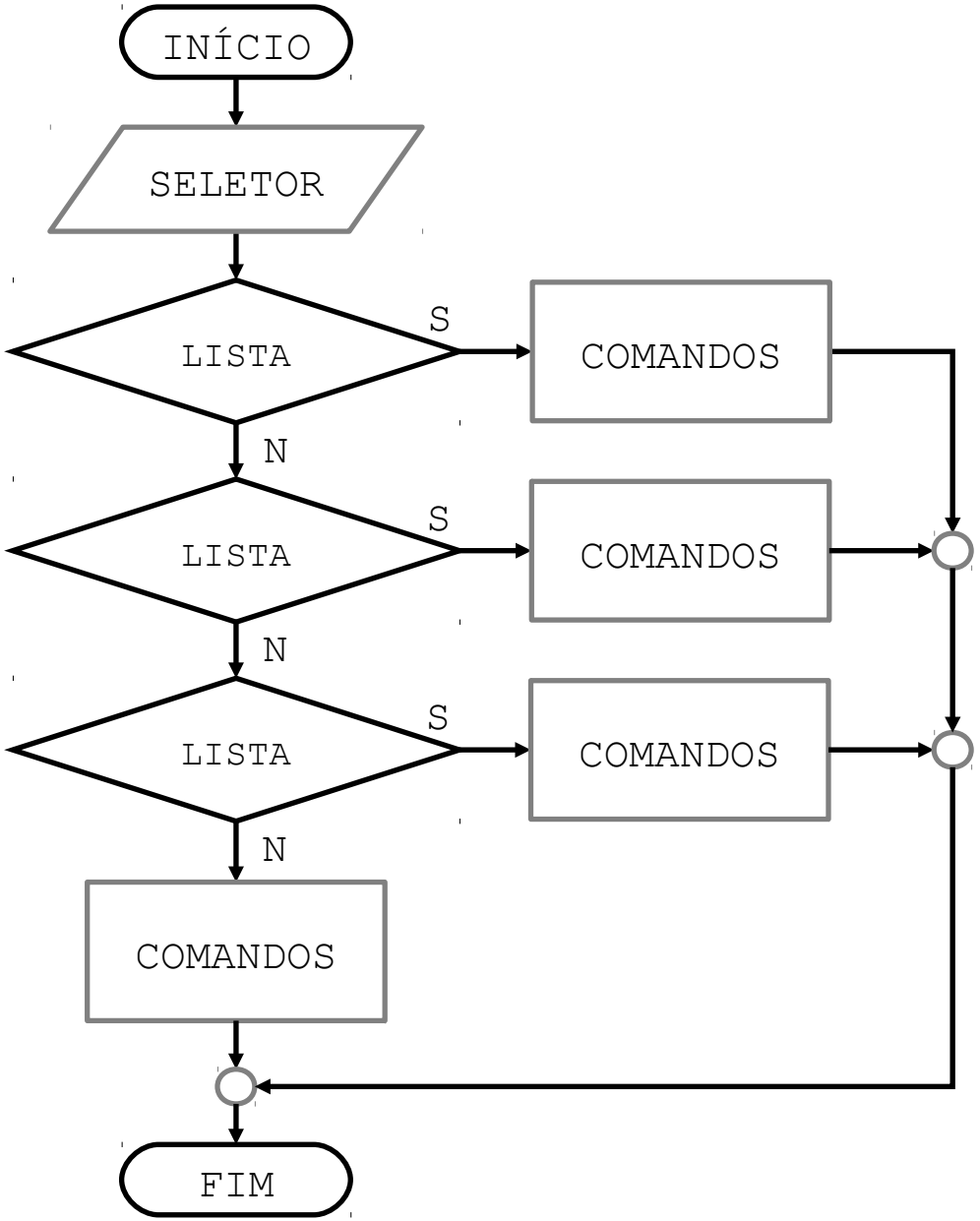
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estrutura de seleção múltipla

escolha..caso

Sintaxe:

```
escolha <seletor>
    caso <lista>
        <comandos>
    caso <lista>
        <comandos>
    ...
    caso <lista>
        <comandos>
    outrocaso
        <comandos>
fimescolha
```



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Combinando estruturas de decisão

Uma estrutura de decisão típica sempre terá duas saídas. Se uma determinada condição for verdadeira, o programa executará uma sequência de comandos ou ações, caso contrário, se a condição for falsa, o programa executará uma outra sequência de comandos ou ações.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Combinando estruturas de decisão

Quando é necessário tomar decisões que podem ter como resultado mais de duas possibilidades ou saídas, devemos combinar mais de uma estrutura de decisão. Estas combinações poderão ser do tipo Concatenada ou Aninhada (Encadeada).

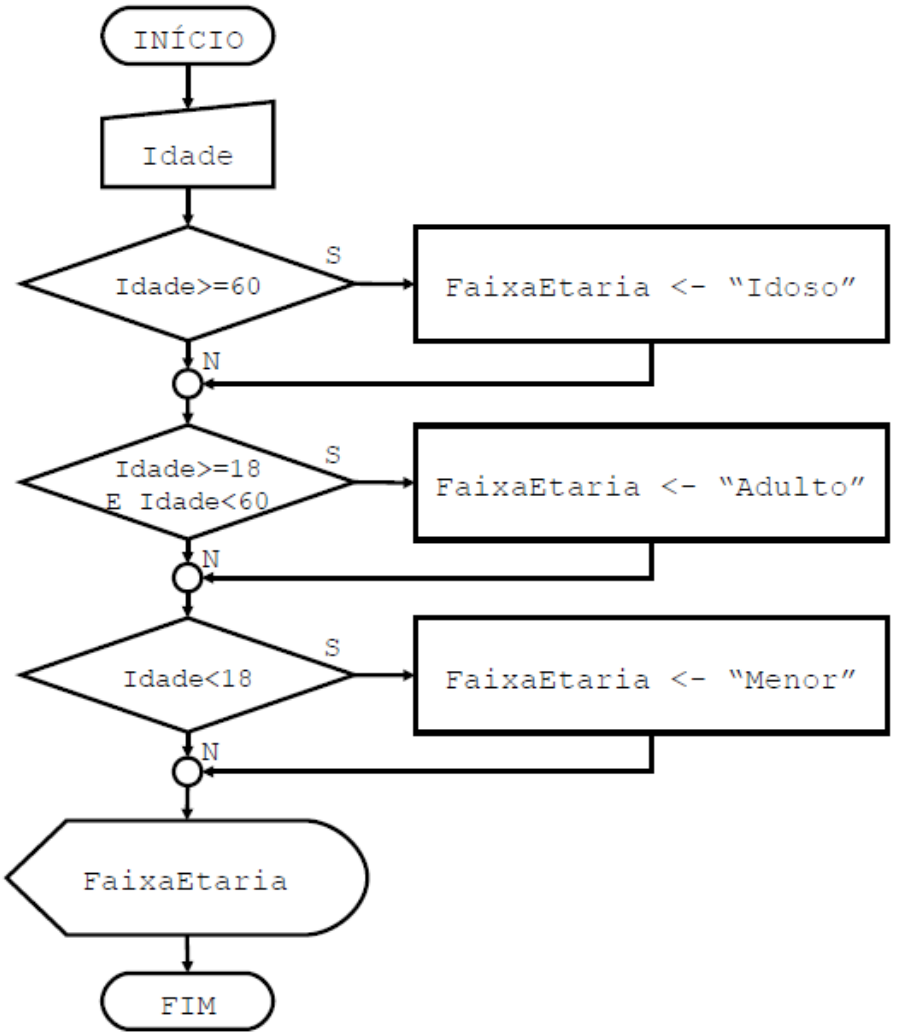
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

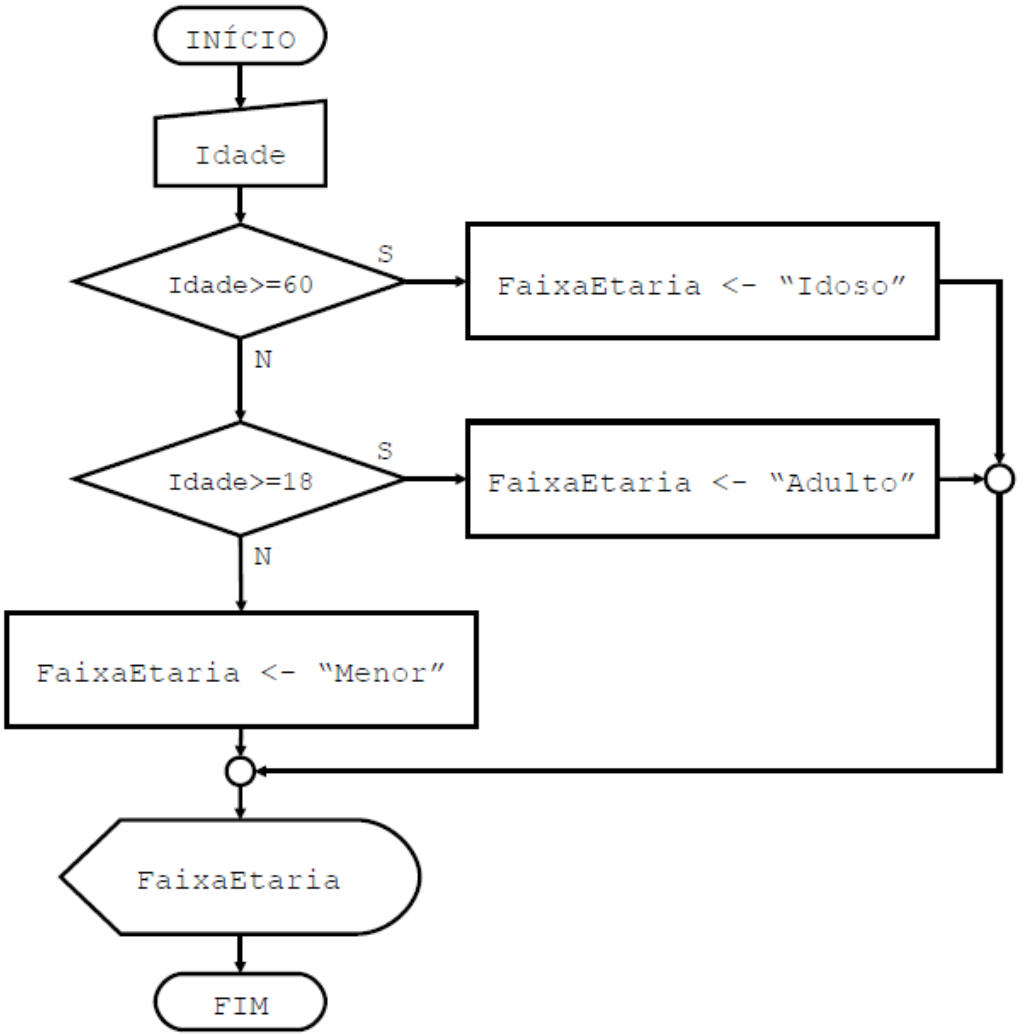
Exemplo de aplicação

Uma clínica médica deseja mostrar, a partir da idade de uma pessoa dada em anos, qual a sua faixa etária, ou seja, se ela é idosa (maior ou igual a 60 anos), adulta (maior ou igual a 18 anos e menor de 60 anos) ou se é menor de idade (menor de 18 anos).

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Concatenada vs. Aninhada - Fluxograma





Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Concatenada vs. Aninhada - Código

```
algoritmo "Idade"
var
   Idade: inteiro
   FaixaEtaria: caracter
inicio
   escreva("Digite a idade da pessoa: ")
   leia(Idade)
   se Idade >= 60 entao
      FaixaEtaria <- "Idosa"
   fimse
   se (Idade >= 18) E (Idade < 60) entao
      FaixaEtaria <- "Adulta"
   fimse
   se Idade < 18 então
      FaixaEtaria <- "Menor"
   fimse
   escreva("A pessoa eh ", FaixaEtaria)
fimalgoritmo
```

```
algoritmo "Idade"
var
   Idade: inteiro
   FaixaEtaria: caracter
inicio
   escreva("Digite a idade da pessoa: ")
   leia(Idade)
   se Idade >= 60 entao
      FaixaEtaria <- "Idosa"
   senao
      se Idade >= 18 entao
         FaixaEtaria <- "Adulta"
      senao
         FaixaEtaria <- "Menor"
      fimse
   fimse
   escreva("A pessoa é ", FaixaEtaria)
fimalgoritmo
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Concatenada vs. Aninhada - Diferenças

Ainda que as duas formas de combinar estruturas de decisão produzam o mesmo resultado, devemos destacar o seguinte:

O código concatenado geralmente é mais fácil de entender, mas no momento da execução, todas as expressões deverão ser testadas.

Por outro lado, apesar do código aninhado não ser fácil de entender num primeiro momento, ele não necessariamente terá que avaliar todas as expressões.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Concatenada vs. Aninhada - Conclusão

Programas que usam estruturas de decisão aninhadas são executados de forma mais rápida que programas que usam estruturas de decisão concatenadas.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Decisão vs. Seleção

Estruturas de decisão podem usar em suas expressões operadores relacionais e lógicos.

Por sua vez, estruturas de seleção não podem usar operadores relacionais e lógicos, somente listas para comparação.

Assim, toda vez que for necessário comparar faixas de valores para tomar uma decisão, a melhor opção são as estruturas de decisão.

Mas quando se tem uma lista de valores discretos a se comparar, as estruturas de seleção são a melhor opção.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Para saber mais...

… leia a Apostila de Introdução a Algoritmos e Programação, dos Professores Fabricio Ferrari e Cristian Cechinel, da Universidade Federal do Pampa, Rio Grande do Sul, Brasil.

… leia a Apostila de Lógica de Programação, da Professora Flávia Pereira de Carvalho, da Faculdades Integradas de Taquara, Rio Grande do Sul, Brasil.

… leia a Apostila de Lógica de Programação, do Professor Paulo Sérgio de Moraes, da Universidade Estadual de Campinas, São Paulo, Brasil.

… leia a Apostila de Programação I, do Professor Edílson de Aguiar et al, da Universidade Federal do Espírito Santo, Espírito Santo, Brasil.

… leia a Apostila de Introdução à Linguagem Pascal, do Prof. Dalton Vinicius Kozak, da Pontificia Universidade Católica do Paraná, Brasil.

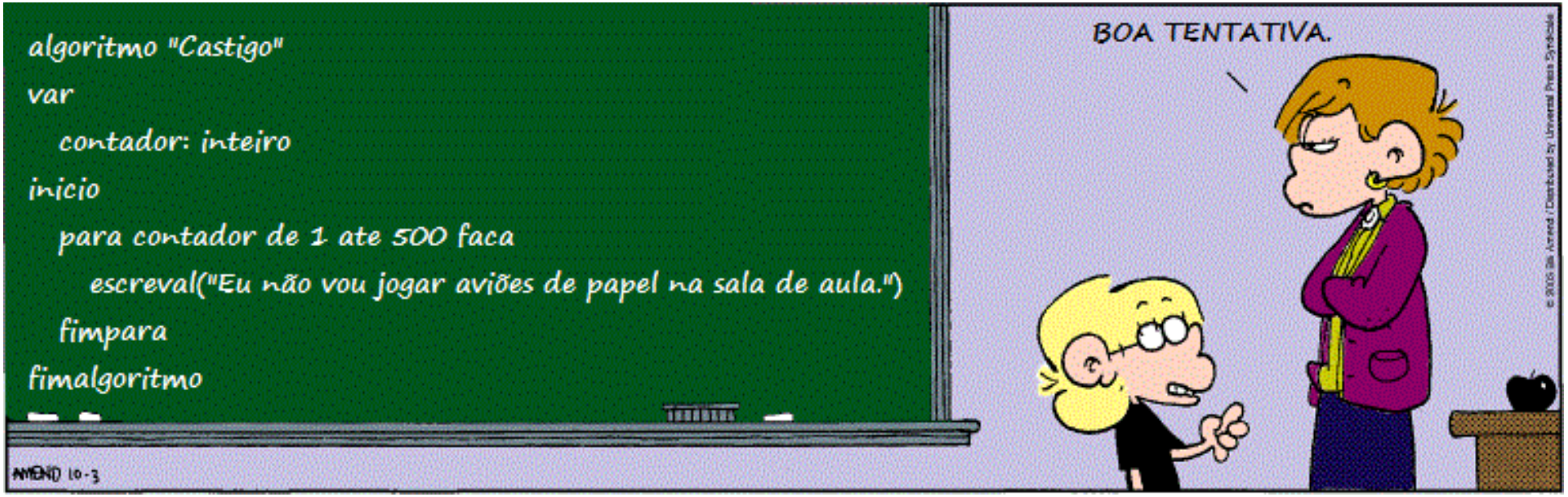
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Módulo 8

Estruturas de repetição

Introdução

Estruturas de repetição são usadas para permitir que um mesmo trecho de código possa ser executado várias vezes, de acordo com condições preestabelecidas ou não.



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estruturas de repetição

As estruturas de repetição podem ser do tipo:

- Estrutura de repetição com variável de controle:

  para..ate..faca ou for..to..do

- Estrutura de repetição com teste no início:

  enquanto..faca ou while..do

- Estrutura de repetição com teste no final:

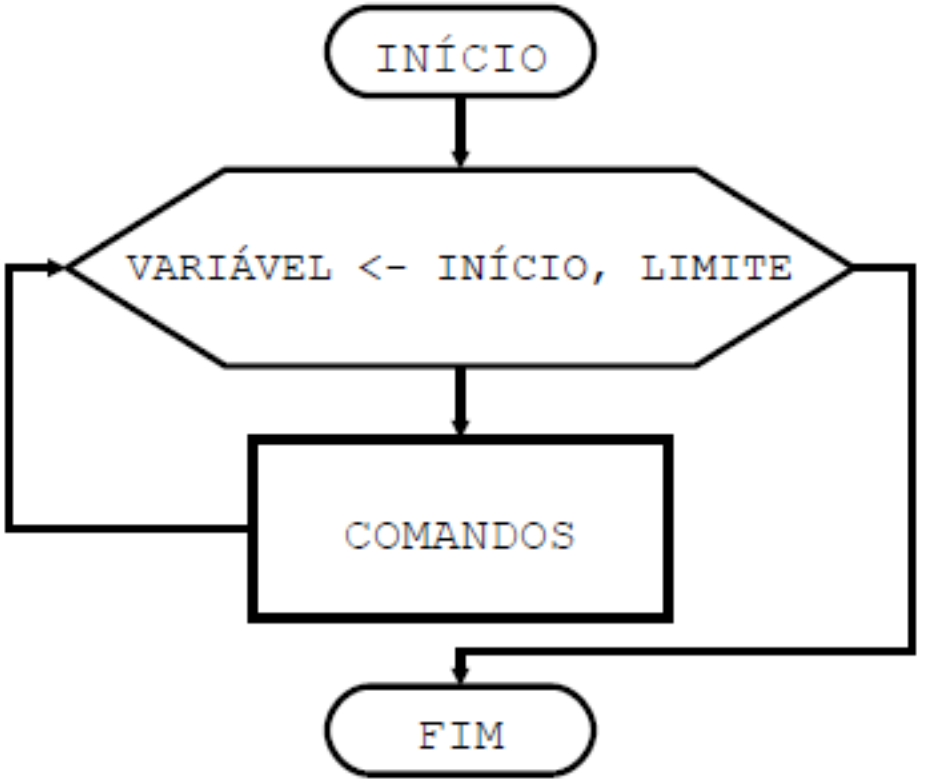
  repita..ate ou repeat..until

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estrutura de repetição com variável de controle *para..ate..faca*

Sintaxe (simplificada):

```
para <variável> de <início> ate <limite> faca
    <comandos>
fimpara
```

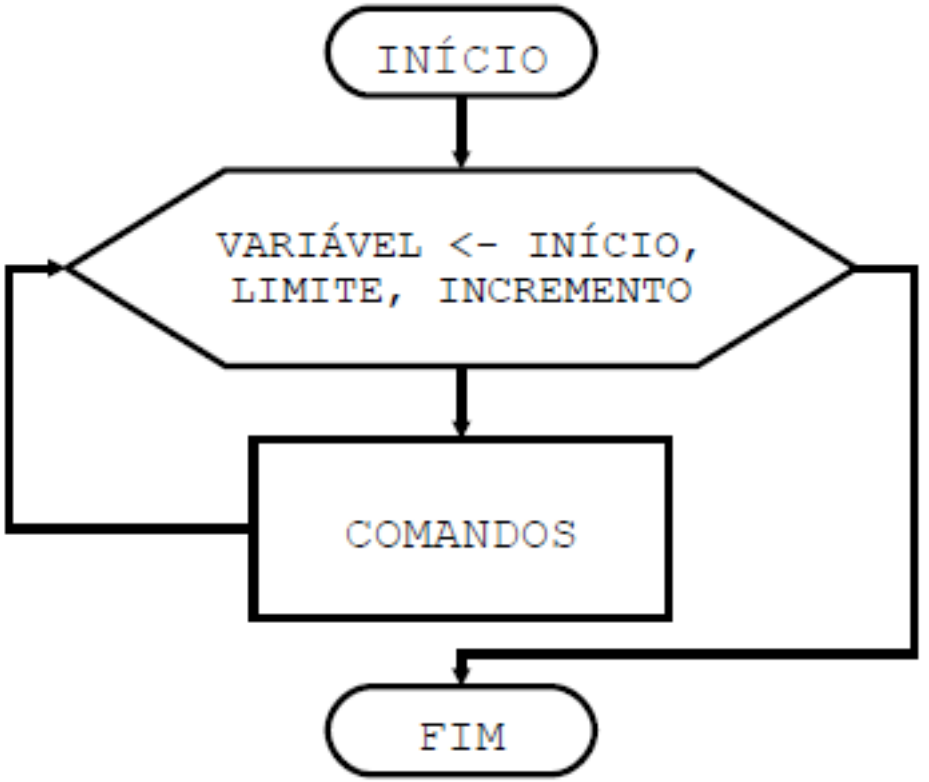


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estrutura de repetição com variável de controle *para..ate..faca*

Sintaxe (completa):

```
para <variável> de <início> ate <limite> [passo <incremento>] faca
    <comandos>
fimpara
```



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

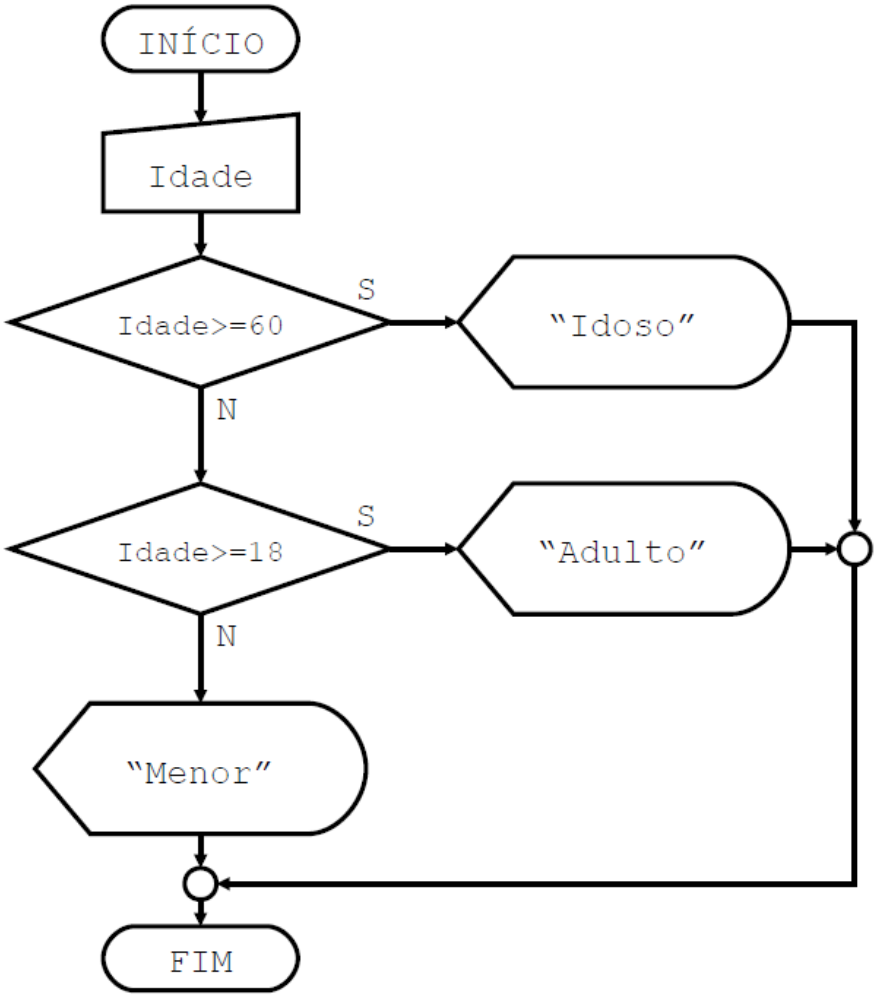
Exemplo de aplicação

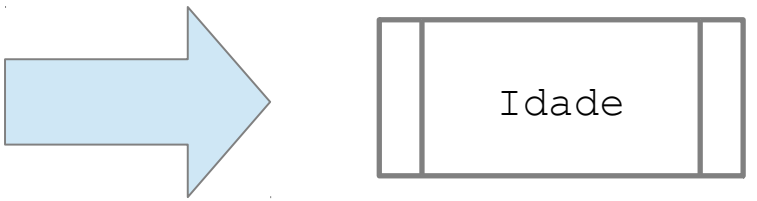
Uma clínica médica deseja mostrar, a partir da idade de uma pessoa dada em anos, se ela é idosa (maior ou igual a 60 anos), adulta (maior ou igual a 18 anos e menor de 60 anos) ou se é menor de idade (menor de 18 anos).

O programa deverá ser executado 10 vezes.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

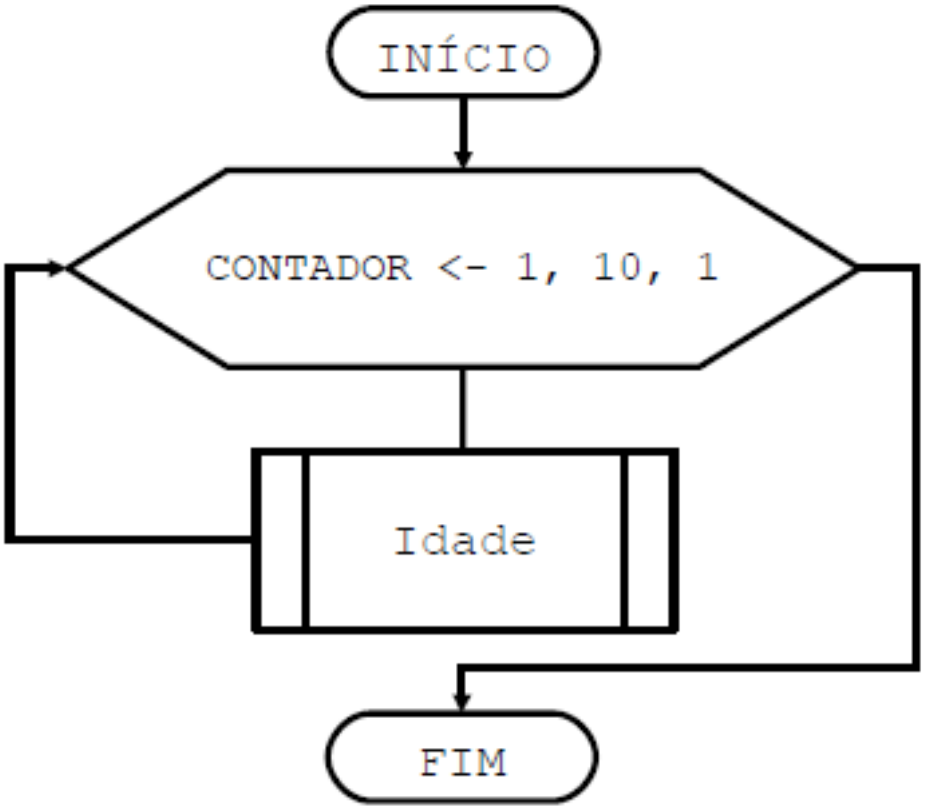
Exemplo de aplicação
Fluxograma com abstração do procedimento





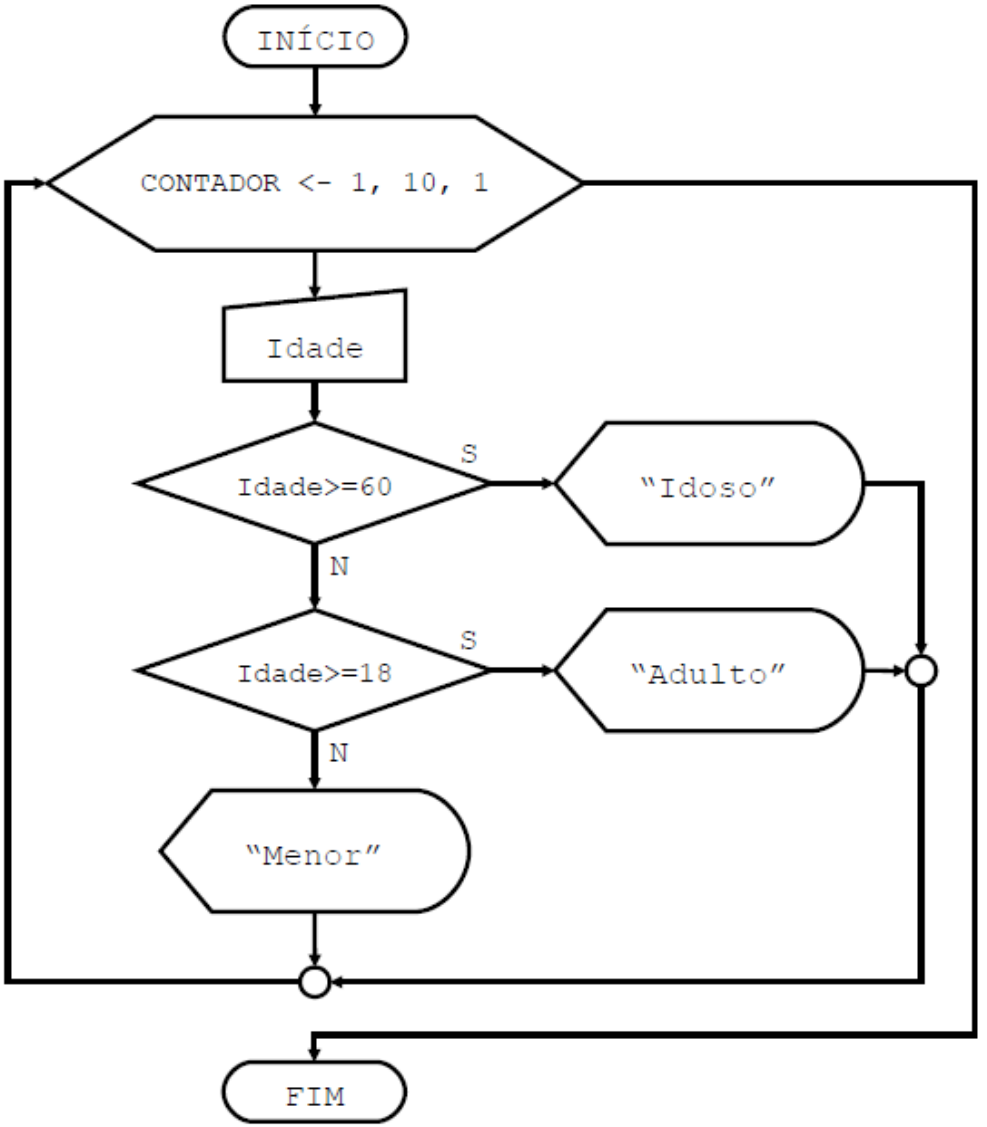
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Fluxograma com estrutura de repetição



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Fluxograma expandido



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Código sem repetição e com repetição

```
algoritmo "Idade"
var
   Idade: inteiro
inicio
   escreva("Digite a idade da pessoa: ")
   leia(Idade)
   se Idade >= 60 entao
      escreval("A pessoa eh Idosa.")
   senao
      se Idade >= 18 entao
         escreval("A pessoa eh Adulta.")
      senao
         escreval("A pessoa eh Menor.")
      fimse
   fimse
fimalgoritmo
```

```
algoritmo "Idade"
var
   Contador, Idade: inteiro
inicio
   para Contador de 1 ate 10 faca
      escreva("Digite a idade da pessoa: ")
      leia(Idade)
      se Idade >= 60 entao
         escreval("A pessoa eh Idosa.")
      senao
         se Idade >= 18 entao
            escreval("A pessoa eh Adulta.")
         senao
            escreval("A pessoa eh Menor.")
         fimse
      fimse
   fimpara
fimalgoritmo
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Algoritmo

1. Início;

2. Repetir dez vezes;

2.1. Obter a idade da pessoa;

2.2. Se a idade da pessoa for maior ou igual a 60 anos, mostrar que a pessoa é idosa;

2.3. Caso contrário, se a idade da pessoa for maior ou igual a 18 anos, mostrar que a pessoa é adulta;

2.4. Caso contrário, mostrar que a pessoa é menor de idade;

3. Fim.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estrutura de repetição com variável de controle
Observações

Estruturas de repetição com variável de controle devem ser usadas quando se conhece de antemão a quantidade de vezes que um trecho de código de programa deve ser executado.

No entanto, se for necessário forçar a saída de uma estrutura de repetição como esta, pode-se usar o comando `interrompa`.

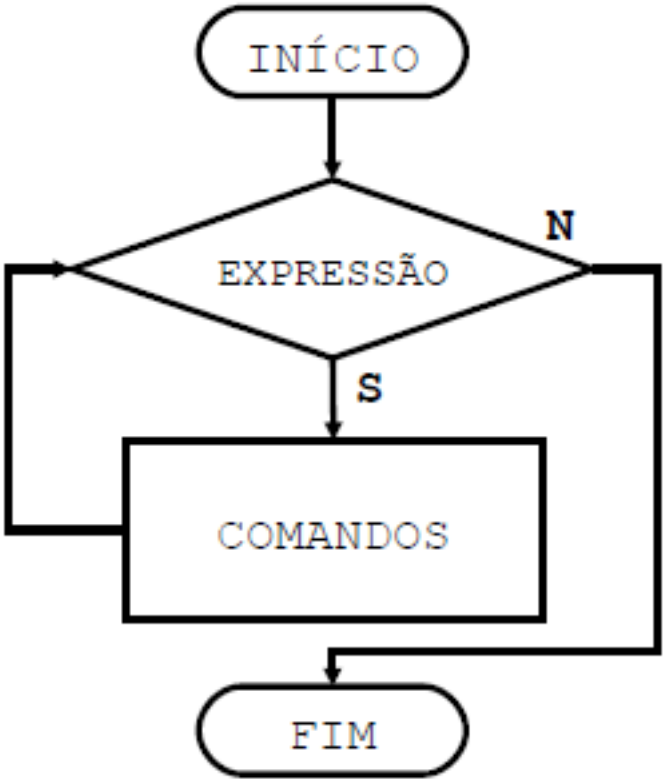
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estrutura de repetição com teste no início

enquanto..faca

Sintaxe:

```
enquanto <expressão> faca
    <comandos>
fimenquanto
```



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

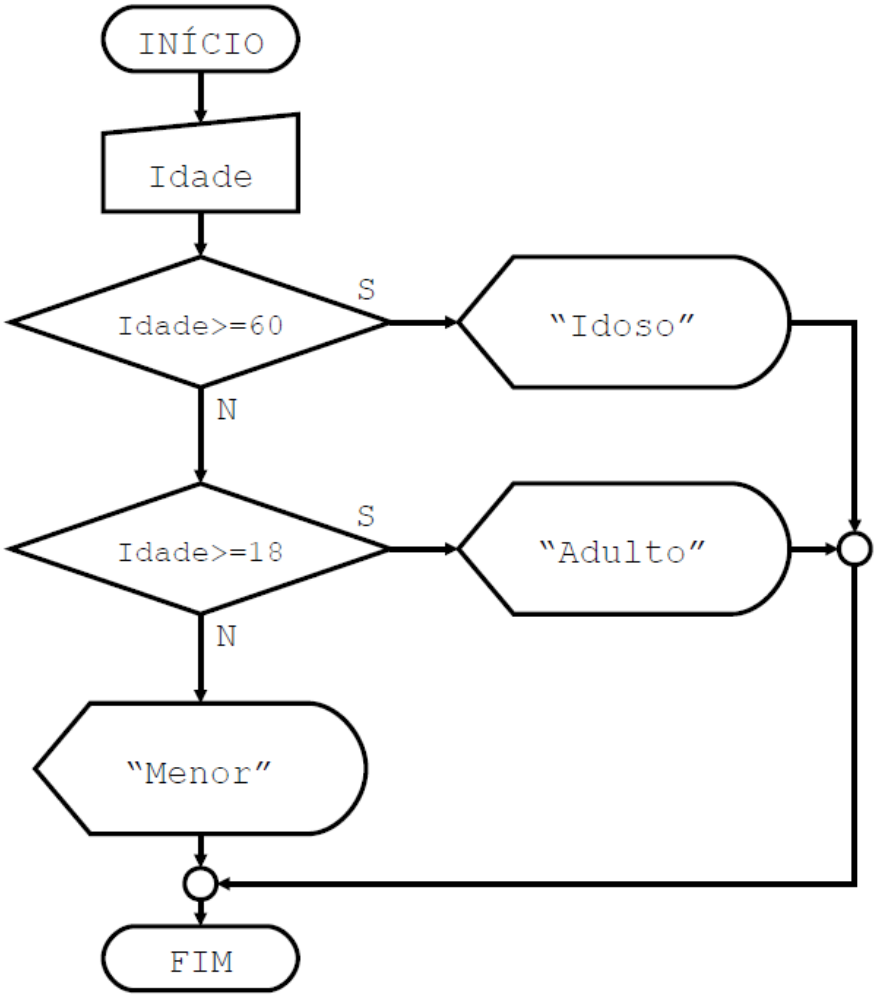
Exemplo de aplicação

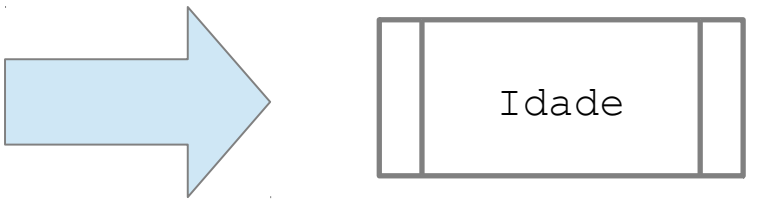
Uma clínica médica deseja mostrar, a partir da idade de uma pessoa dada em anos, se ela é idosa (maior ou igual a 60 anos), adulta (maior ou igual a 18 anos e menor de 60 anos) ou se é menor de idade (menor de 18 anos).

O usuário deverá indicar se deseja que o programa seja ou não executado mais de uma vez.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

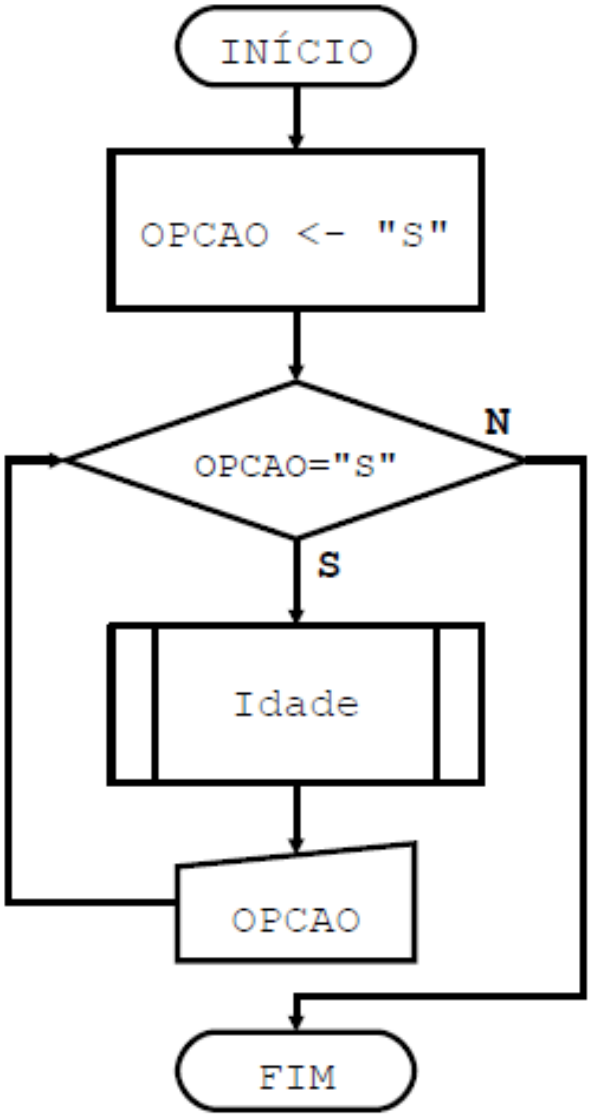
Exemplo de aplicação
Fluxograma com abstração do procedimento





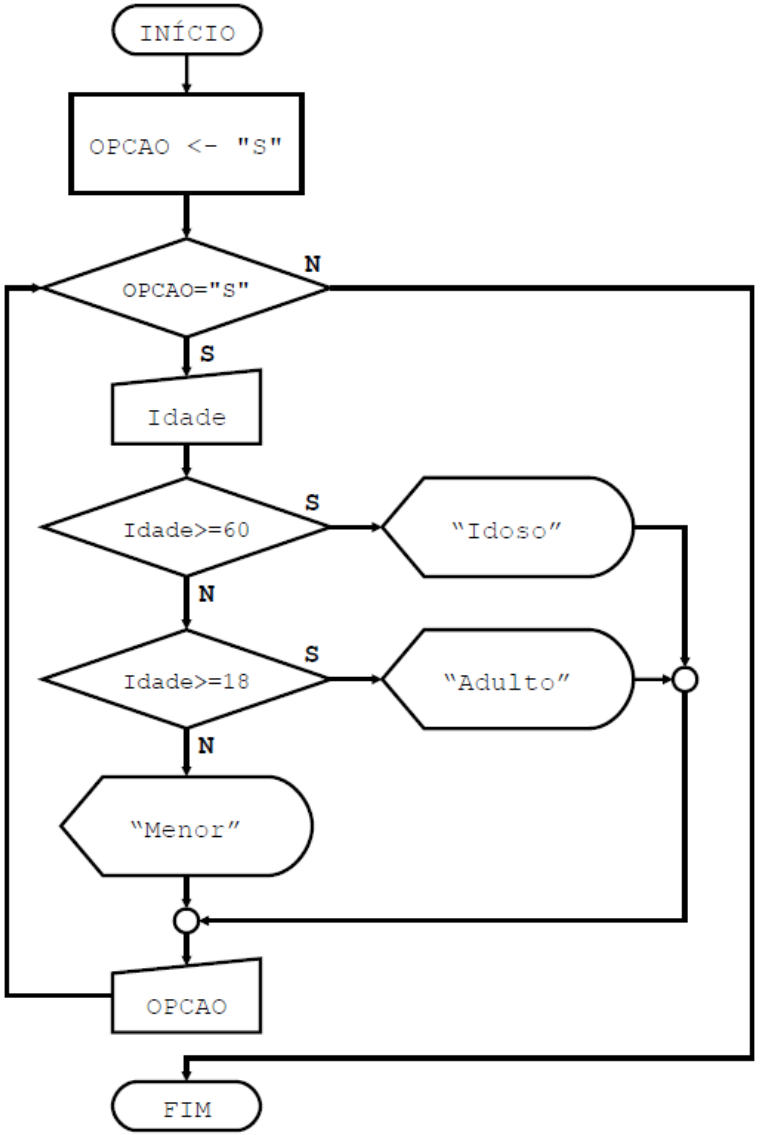
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Fluxograma com estrutura de repetição



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Fluxograma expandido



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Código sem repetição e com repetição

```
algoritmo "Idade"
var
   Idade: inteiro
inicio
   escreva("Digite a idade da pessoa: ")
   leia(Idade)
   se Idade >= 60 entao
      escreval("A pessoa eh Idosa.")
   fimse
   se (Idade >= 18) E (Idade < 60) entao
      escreval("A pessoa eh Adulta.")
   fimse
   se Idade < 18 entao
      escreval("A pessoa eh Menor.")
   fimse
fimalgoritmo
```

```
algoritmo "Idade"
var
   Opcao: caracter
   Idade: inteiro
inicio
   Opcao <- "s"
   enquanto (Opcao = "s") ou (Opcao = "S") faca
      escreva("Digite a idade da pessoa: ")
      leia(Idade)
      se Idade >= 60 entao
         escreval("A pessoa eh Idosa.")
      senao
         se Idade >= 18 entao
            escreval("A pessoa eh Adulta.")
         senao
            escreval("A pessoa eh Menor.")
         fimse
      fimse
      escreval("Deseja continuar? <s/n>: ")
      leia(Opcao)
   fimenquanto
fimalgoritmo
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Algoritmo

1. Início;

2. Opção igual a “s”;

3. Enquanto Opção for igual a “s” ou “S”;

3.1. Obter a idade da pessoa;

3.2. Se a idade da pessoa for maior ou igual a 60 anos, mostrar que a pessoa é idosa;

3.3. Caso contrário, se a idade da pessoa for maior ou igual a 18 anos, mostrar que a pessoa é adulta;

3.4. Caso contrário, mostrar que a pessoa é menor de idade;

3.5. Obter Opção;

4. Fim.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estrutura de repetição com teste no início
Observações

Estruturas de repetição com teste no início permitem que o usuário controle a quantidade de vezes que um trecho de código de programa deve ser executado.

Como o teste é feito no início, pode acontecer do trecho de código de programa ser executado nenhuma, uma ou várias vezes.

No entanto, se for necessário forçar a saída de uma estrutura de repetição como esta, pode-se usar o comando `interrompa`.

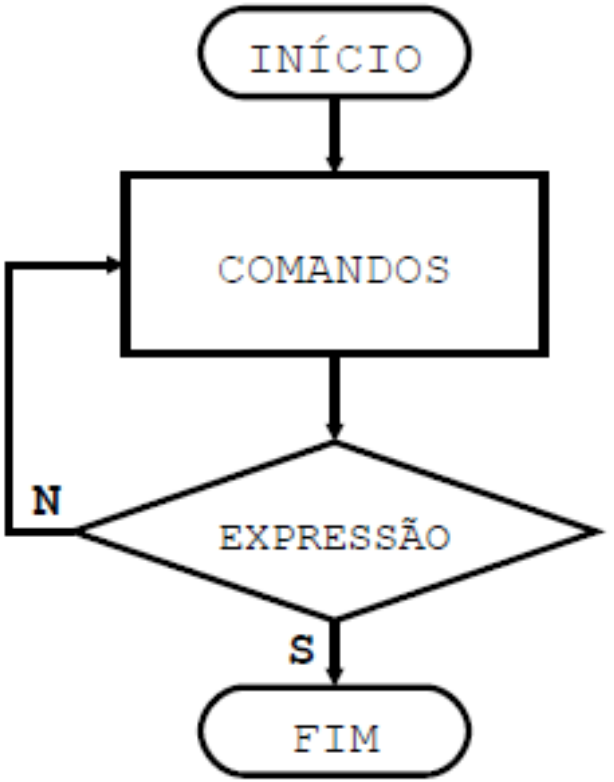
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estrutura de repetição com teste no final

repita..ate

Sintaxe:

```
repita
    <comandos>
ate <expressão>
```



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

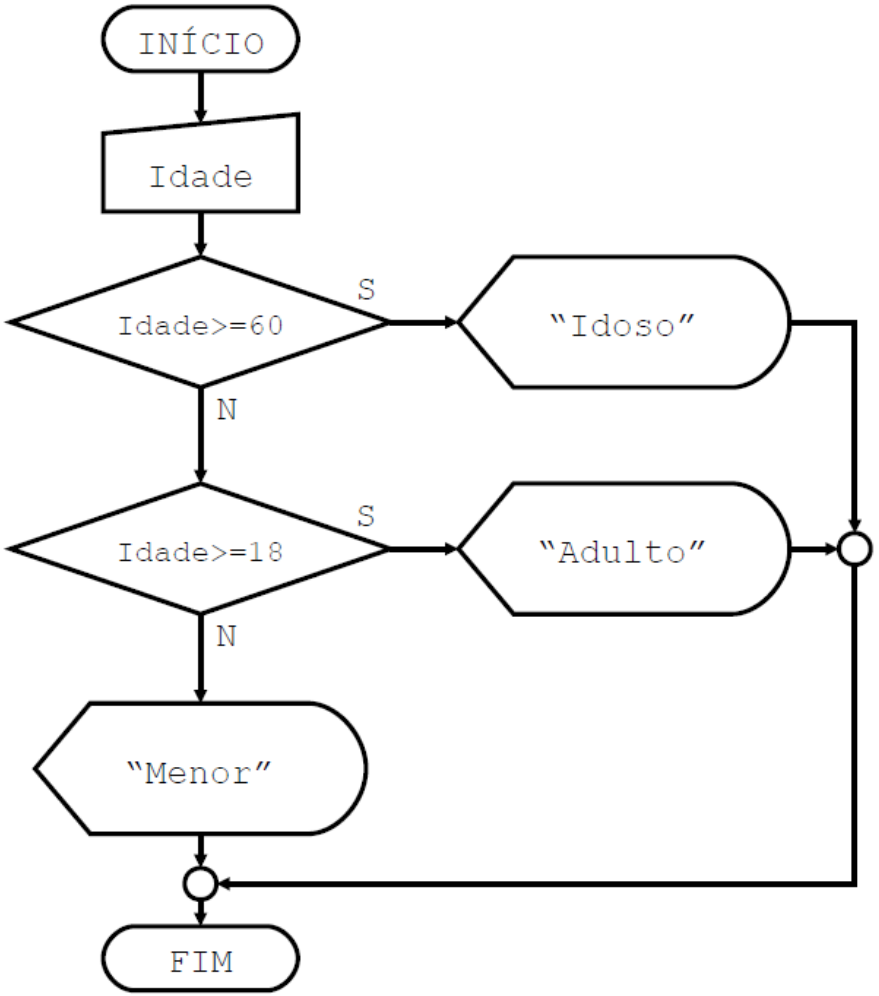
Exemplo de aplicação

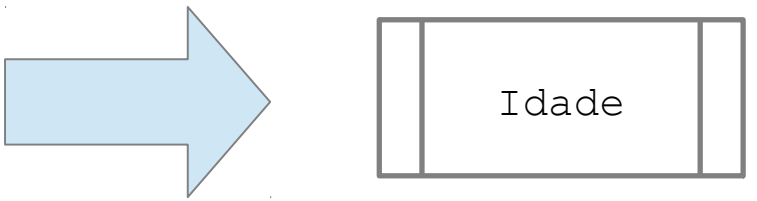
Uma clínica médica deseja mostrar, a partir da idade de uma pessoa dada em anos, se ela é idosa (maior ou igual a 60 anos), adulta (maior ou igual a 18 anos e menor de 60 anos) ou se é menor de idade (menor de 18 anos).

O usuário deverá indicar se deseja que o programa seja ou não executado mais de uma vez.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

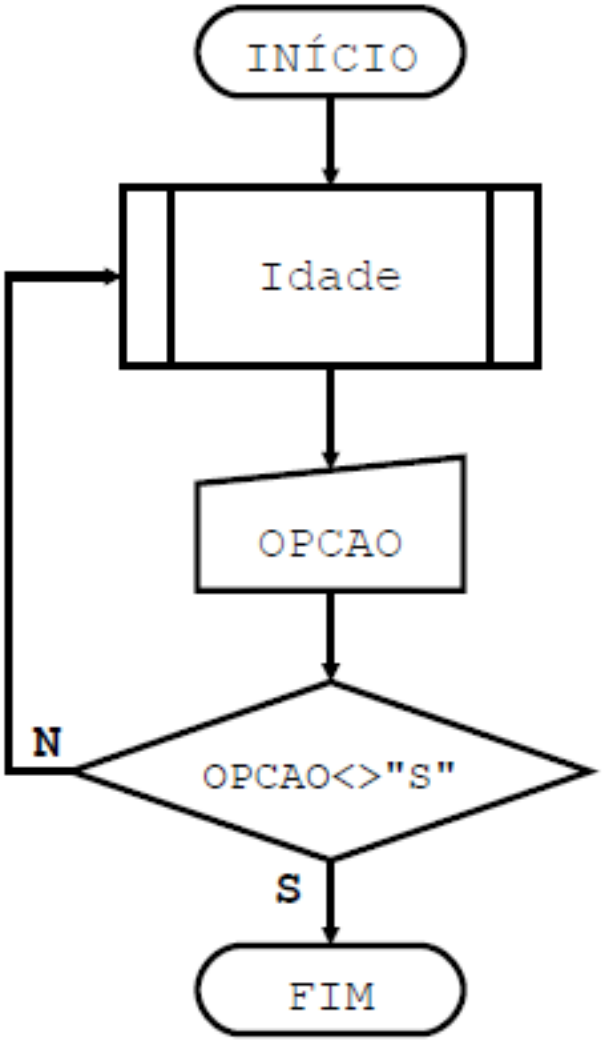
Exemplo de aplicação
Fluxograma com abstração do procedimento





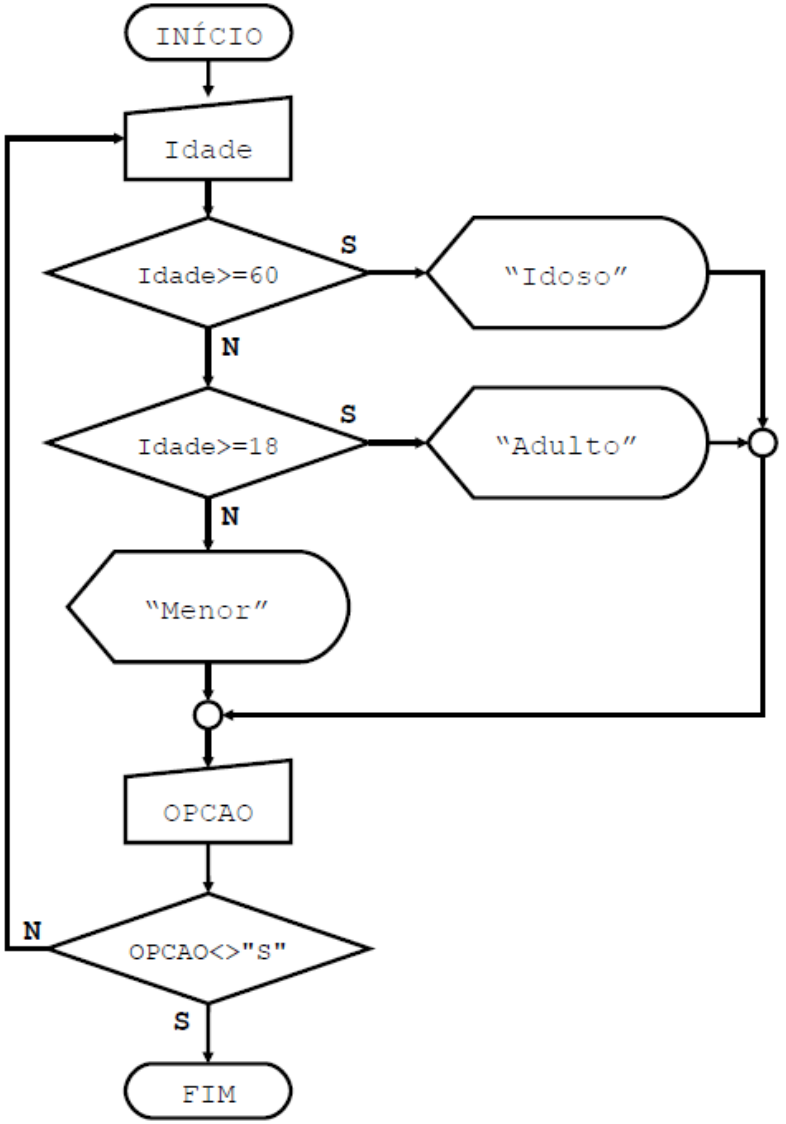
Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Fluxograma com estrutura de repetição



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Fluxograma expandido



Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Código sem repetição e com repetição

```
algoritmo "Idade"
var
   Idade: inteiro
inicio
   escreva("Digite a idade da pessoa: ")
   leia(Idade)
   se Idade >= 60 entao
      escreval("A pessoa eh Idosa.")
   fimse
   se (Idade >= 18) E (Idade < 60) entao
      escreval("A pessoa eh Adulta.")
   fimse
   se Idade < 18 entao
      escreval("A pessoa eh Menor.")
   fimse
fimalgoritmo
```

```
algoritmo "Idade"
var
   Opcao: caracter
   Idade: inteiro
inicio
   repita
      escreva("Digite a idade da pessoa: ")
      leia(Idade)
      se Idade >= 60 entao
         escreval("A pessoa eh Idosa.")
      senao
         se Idade >= 18 entao
            escreval("A pessoa eh Adulta.")
         senao
            escreval("A pessoa eh Menor.")
         fimse
      fimse
      escreval("Deseja continuar? <s/n>: ")
      leia(Opcao)
   ate (Opcao <> "s") ou (Opcao <> "S")
fimalgoritmo
```


Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Exemplo de aplicação
Algoritmo

1. Início;

2. Repetir;

2.1. Obter a idade da pessoa;

2.2. Se a idade da pessoa for maior ou igual a 60 anos, mostrar que a pessoa é idosa;

2.3. Caso contrário, se a idade da pessoa for maior ou igual a 18 anos, mostrar que a pessoa é adulta;

2.4. Caso contrário, mostrar que a pessoa é menor de idade;

2.5. Obter Opção;

2.6. Até que Opção seja diferente de "s" ou "S";

4. Fim.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Estrutura de repetição com teste no final
Observações

Estruturas de repetição com teste no final permitem que o usuário controle a quantidade de vezes que um trecho de código de programa deve ser executado.

Como o teste é feito no final, o trecho de código de programa será executado pelo menos uma vez.

No entanto, se for necessário forçar a saída de uma estrutura de repetição como esta, pode-se usar o comando `interrompa`.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

Para saber mais...

… leia a Apostila de Introdução à Linguagem Pascal, do Prof. Dalton Vinicius Kozak, da Pontificia Universidade Católica do Paraná, Brasil.

… leia a Apostila de Introdução a Algoritmos e Programação, dos Professores Fabricio Ferrari e Cristian Cechinel, da Universidade Federal do Pampa, Rio Grande do Sul, Brasil.

… leia a Apostila de Lógica de Programação, da Professora Flávia Pereira de Carvalho, da Faculdades Integradas de Taquara, Rio Grande do Sul, Brasil.

… leia a Apostila de Lógica de Programação, do Professor Paulo Sérgio de Moraes, da Universidade Estadual de Campinas, São Paulo, Brasil.

… leia a Apostila de Programação I, do Professor Edílson de Aguiar et al, da Universidade Federal do Espírito Santo, Espírito Santo, Brasil.

Módulo 9

Vetores e matrizes

Para saber mais...

… acesse o material online sobre Protocolo de Roteamento Dinâmico OSPF, de Júlio Battisti.

… acesse o material online sobre Protocolo de Roteamento Dinâmico OSPF, de Aaron Balchunas.

Prof. Me. Wallace Rodrigues de Santana
www.neutronica.com.br

FIM