

Estrutura de Dados 1º semestre de 2021

Professor Mestre Fabio Pereira da Silva

O que é um Algoritmo?

- Um algoritmo pode ser definido como uma sequência finita de passos, descritos em uma ordem lógica, que atingirão um objetivo bem definido.
- **Nunca esqueça esta definição.**

O que é um Algoritmo?

- “Algoritmo é uma sequência de passos que deve ser seguida para a realização de uma tarefa.” Ascencio, 1999
- “Algoritmo é uma sequência finita de instruções ou operações cuja execução, em tempo finito, resolve um problema computacional, qualquer que seja sua instância.” Salvetti, 1999
- “Algoritmos são regras formais para a obtenção de um resultado ou da solução de um problema, englobando fórmulas de expressões aritméticas.” Manzano, 1997

Conceitos Fundamentais

- Lógica: é a técnica de encadear pensamentos para atingir determinado objetivo;
- Sequência Lógica: são passos executados até atingir um objetivo ou solução de um problema.
- Instruções: um conjunto de regras ou normas definidas para a realização ou emprego de algo. É o que indica a um computador uma ação elementar a executar.

Exemplos de Algoritmos

- Instruções para se utilizar um aparelho eletrodoméstico;
- Uma receita para preparo de algum prato;
- Guia de preenchimento para declaração do imposto de renda;
- A regra para determinação de máximos e mínimos de funções por derivadas sucessivas;
- A maneira como as contas de água, luz e telefone são calculadas mensalmente

Formas de representação - Narrativa

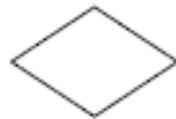
- **Narrativa:** nesta forma de representação, os algoritmos são expressos em linguagem natural
- EXEMPLO – Preparar um bolo
 - Receita de Bolo:
 - Providencie manteiga, ovos, 2 Kg de massa, etc.
 - Misture os ingredientes
 - Despeje a mistura na fôrma de bolo
 - Leve a fôrma ao forno
 - Espere 20 minutos
 - Retire a fôrma do forno
 - Deixe esfriar
 - Prove

Formas de representação - Fluxograma

- **Fluxograma:** é uma representação gráfica dos algoritmos
- Cada figura geométrica representa diferentes ações
- Facilita o entendimento das ideias contidas no algoritmo



Cálculo



Decisão



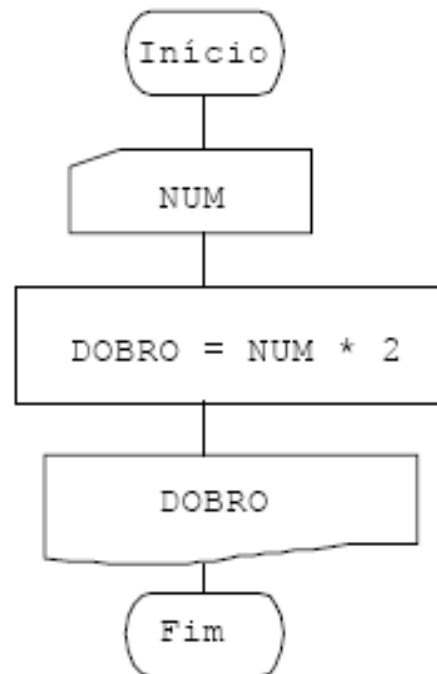
Entrada



Saída



Início/Fim



Início do algoritmo





Entrada do número

Cálculo do dobro do número

Apresentação do resultado

Fim do algoritmo

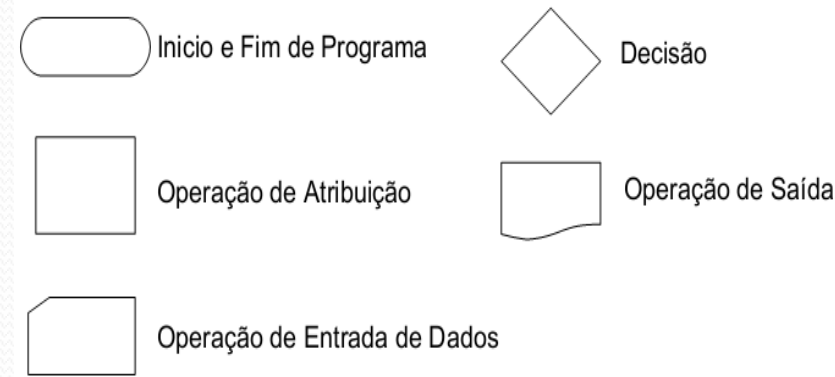
Formas de representação - Fluxograma

Símbolo	Função
 TERMINAL	Indica o INÍCIO ou FIM de um processamento Exemplo: Início do algoritmo
 PROCESSAMENTO	Processamento em geral Exemplo: Calculo de dois números
 ENTRADA DE DADO MANUAL	Indica entrada de dados através do Teclado Exemplo: Digite a nota da prova 1
 EXIBIR	Mostra informações ou resultados Exemplo: Mostre o resultado do calculo

Formas de representação - Fluxograma

- Elementos do fluxograma:

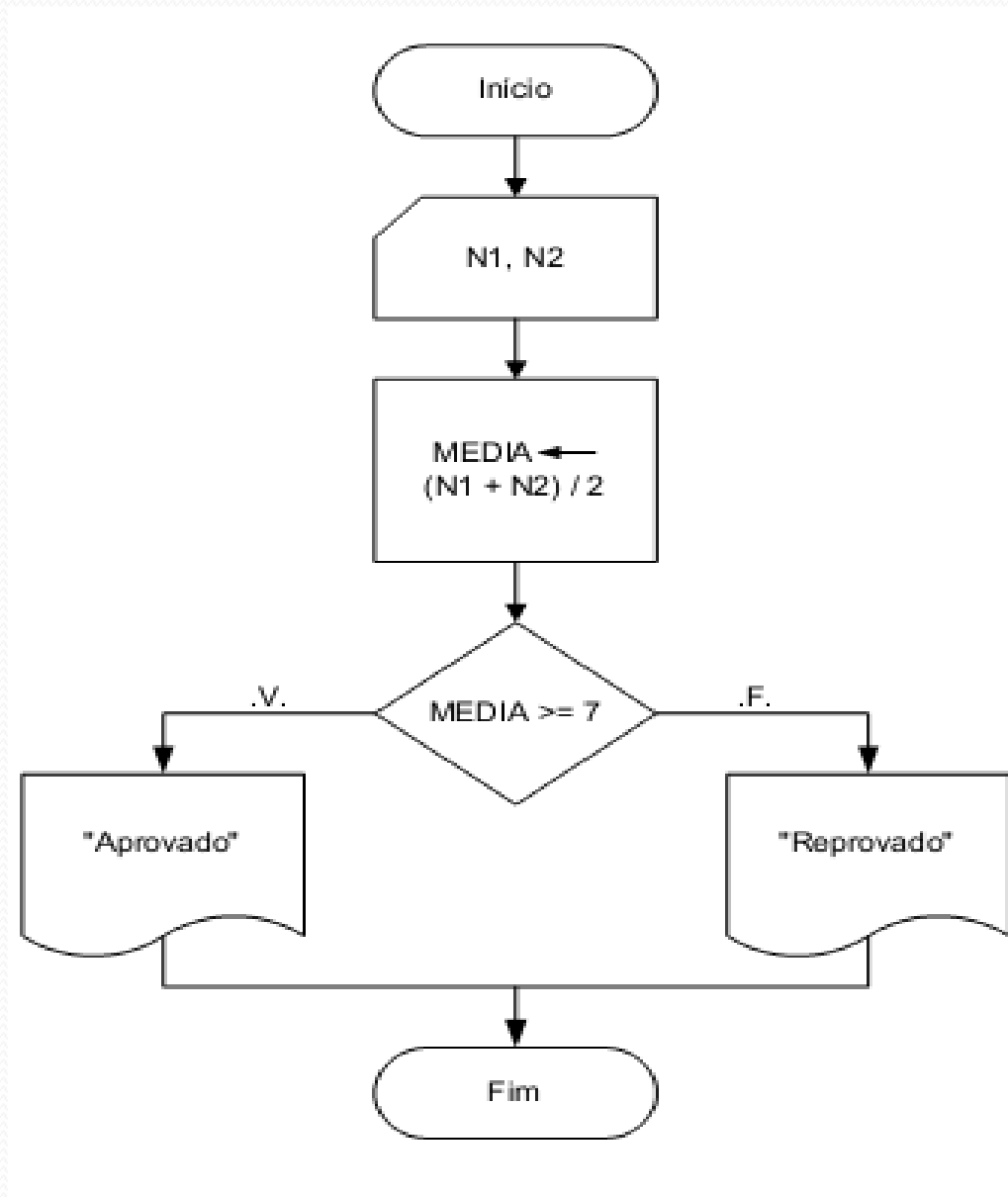
- Início e fim de programa
 - Representados por uma elipse
- Operação de Atribuição
 - Representada por um retângulo
- Operação de Entrada de Dados
 - Representada por um retângulo com um dos cantos dobrados (como em uma folha de papel)
- Decisão
 - Representada por um losango
- Operação de Saída
 - Representada por um retângulo com um dos lados recordado de maneira ondulada



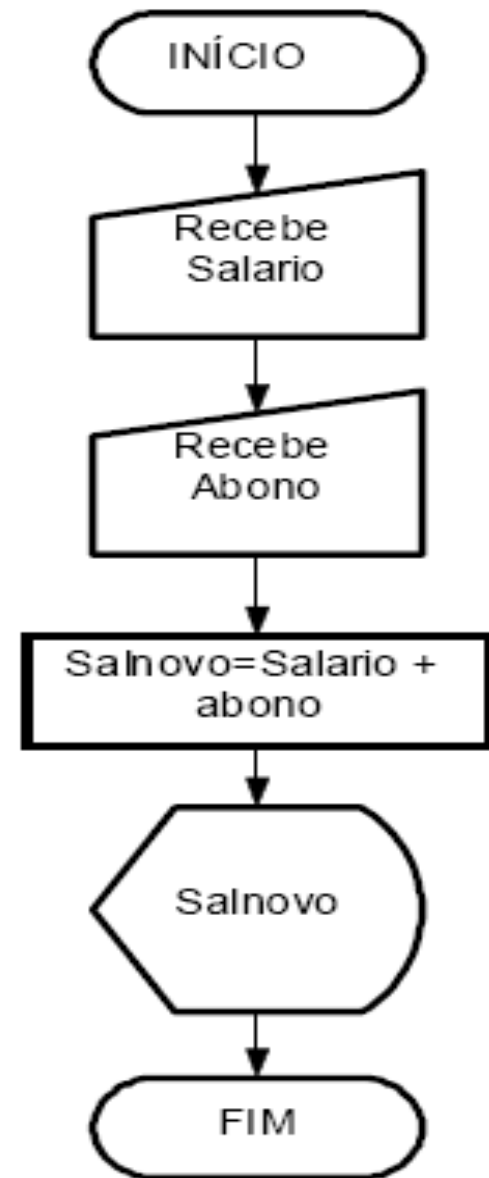
Exemplo de Fluxograma

- Início (dentro de uma elipse)
- Calcular média de duas notas (dentro de um retângulo com um dos cantos dobrados)
- A média para passar é 7 (dentro de um retângulo)
- Indicar “Aprovado” ou “Reprovado” como saída (verifica se a média é maior ou igual a 7 dentro de um losango)
- Se a média for maior ou igual a 7 imprime “Aprovado” dentro de um retângulo com um dos lados recortado de maneira ondulada
- Se a média for menor do que 7 imprime “Reprovado” dentro de um retângulo com um dos lados recortado de maneira ondulada
- Fim de programa (dentro de uma elipse)

Exemplo de Fluxograma



Exemplo de Fluxograma



Formas de representação - Pseudocódigo

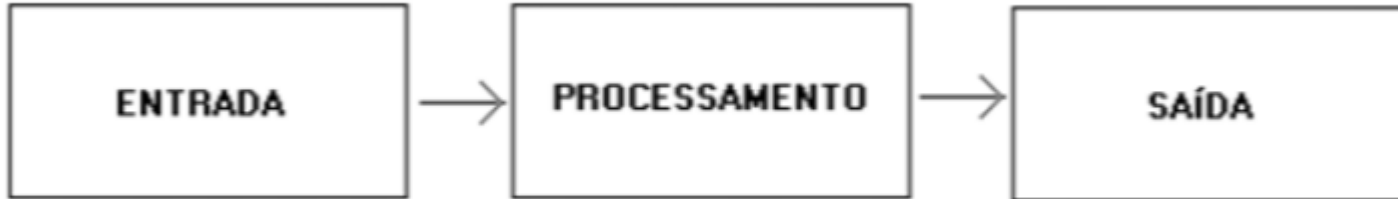
- **Pseudocódigo:** São independentes das linguagens de programação;
- Devem ser fácil de se interpretar e de codificar;
- Devem ser o intermediário entre a linguagem falada e a linguagem de programação (e.g., C, Java e Python).

Formas de representação - Pseudocódigo

- **Pseudocódigo:** forma de representação de algoritmos rica em detalhes
- É uma aproximação do código final a ser escrito em uma linguagem de programação
- Algoritmo é uma palavra que indica o início da definição de um algoritmo em forma de pseudocódigo
- <nome_do_algoritmo> é um nome simbólico dado ao algoritmo com a finalidade de distingui-los dos demais
- <declaração_de_variáveis> consiste em uma porção opcional onde são declaradas as variáveis globais usadas no algoritmo principal e, eventualmente, nos subalgoritmos
- <subalgoritmos> consiste de uma porção opcional de pseudocódigo onde são definidos os subalgoritmos
- Início e Fim são respectivamente as palavras que delimitam o início e o término do conjunto de instruções do corpo do algoritmo

Exemplo de Pseudocódigo

- Exemplo: Calcular a média aritmética dos alunos.
- Os alunos realizarão quatro provas: P₁, P₂, P₃ e P₄.
- Quais são os dados de entrada?
- Qual será o processamento a ser utilizado?
- Quais serão os dados de saída?



Exemplo de Pseudocódigo

- Receba a nota da prova 1
- Receba a nota da prova 2
- Receba a nota da prova 3
- Receba a nota da prova 4
- Some todas as notas e divida o resultado por 4
- Mostre o resultado da divisão

Exemplo de Pseudocódigo

- Algoritmo da média de duas notas em pseudocódigo

```
Algoritmo Media;  
  Var N1, N2, MEDIA: real;  
Início  
  Leia (N1, N2);  
  MEDIA ← (N1 + N2) / 2;  
  Se MEDIA >= 7 então  
    Escreva “Aprovado”;  
  Senão  
    Escreva “Reprovado”;  
Fim_se  
Fim
```

Exemplo de Pseudocódigo

- Algoritmo que realiza a divisão de dois números

Algoritmo Divisão;

Var

**n1, n2: inteiro;
resultado: real;**

Inicio

**Escreva "Digite o dividendo";
Leia (n1);
Escreva "Digite o divisor ";
Leia (n2);**

Se n2=0 então

Escreva "Impossível dividir por 0";

Senão

**resultado <- n1/n2;
Escreva "O resultado é", resultado;**

Fim_se

Fim

Tipos Básicos de Dados

- Dados Numéricos Inteiros
 - São os números positivos e negativos sem casas decimais
- Dados Numéricos Reais
 - São os números positivos e negativos que possuem casas decimais
- Dados Literais (caracteres)
 - São sequências de caracteres
- Dados Lógicos ou Booleanos
 - Podem ser verdadeiros ou Falsos, apenas

Variáveis

- O armazenamento de informações pelo computador em sua memória, se dá em uma região nomeada através de uma variável
- Uma variável possui:
 - NOME
 - TIPO
 - CONTEÚDO
- As regras para nomes de variáveis mudam de uma linguagem para outra

Variáveis

- Variáveis devem ser declaradas antes de serem utilizadas
- Ao declarar uma variável, o computador reserva um espaço na memória para ela
- A memória é constituída de bytes, que são conjuntos de 8 bits
- Cada tipo de variável ocupa um tamanho diferente na memória, isso varia para cada linguagem de programação

Estrutura de decisão

- As instruções condicionais avaliam se uma condição é verdadeira ou falsa, e em seguida especificam uma ou mais instruções a serem executadas, dependendo do resultado dessa avaliação.

Comandos de seleção:

- **if...then** / **if...then...else/elseif**

Estruturas de Repetição

Digamos que o usuário deseja escrever automaticamente uma sequência numérica de 1 a 10, com um número em cada linha. O algoritmo ficaria extenso mesmo para algo tão simples.

```
1 algoritmo "numeros"  
2 var  
3 inicio  
4 escreval('1')  
5 escreval('2')  
6 escreval('3')  
7 escreval('4')  
8 escreval('5')  
9 escreval('6')  
10 escreval('7')  
11 escreval('8')  
12 escreval('9')  
13 escreval('10')  
14 fimalgoritmo
```

Estruturas de repetição

❖ Uma estrutura de repetição obrigatoriamente possui:

❖ *Uma variável de controle.*

❖ Usada para contar quantas vezes o laço se repete.

❖ *Um incremento.*

❖ Usado para aumentar ou diminuir o valor da variável de controle. Pode ser por atribuição ou por digitação do usuário.

❖ *Um teste lógico.*

❖ Usado para verificar se a condição de parada foi atingida.

Vetores

- Vetor é uma coleção de variáveis de mesmo tipo que compartilham o mesmo nome (identificador).

Declaração de um vetor:

<tipo de dado> <identificador>[<tamanho>];

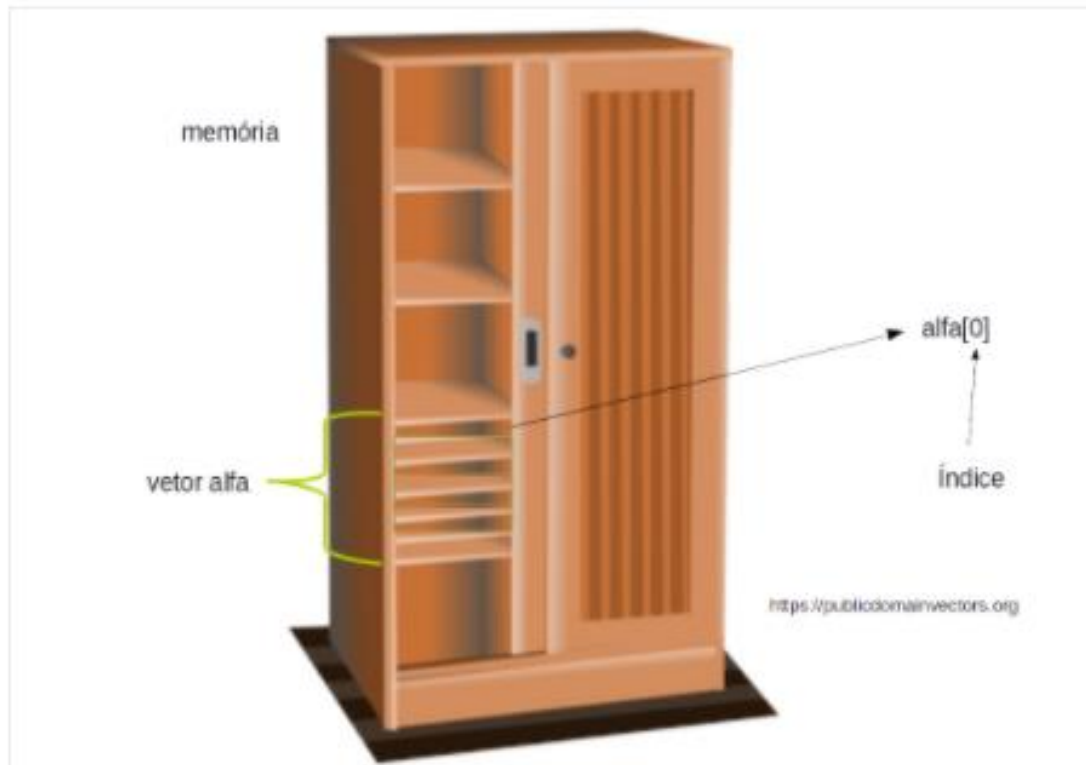
- Exemplo: Definição de um vetor do tipo inteiro com 5 posições e um vetor do tipo char com 32 posições:
 - `int x[5];`
 - `double y[50];`

Vetores

- Cada elemento do vetor é referenciado individualmente por meio de um número inteiro e positivo, entre colchetes.
 - Este número/índice representa a posição do elemento no vetor.
- Exemplo:
 - `y[10] = 'a';`

Vetores

- Um vetor pode ser visto como uma variável (uma caixa) dividida em partes menores (CAIXAS menores) acessadas por um índice (posição). Em termos mais técnicas cada elemento do vetor pode ser acessado através da indexação do vetor. Os elementos do vetor possuem um tipo único.

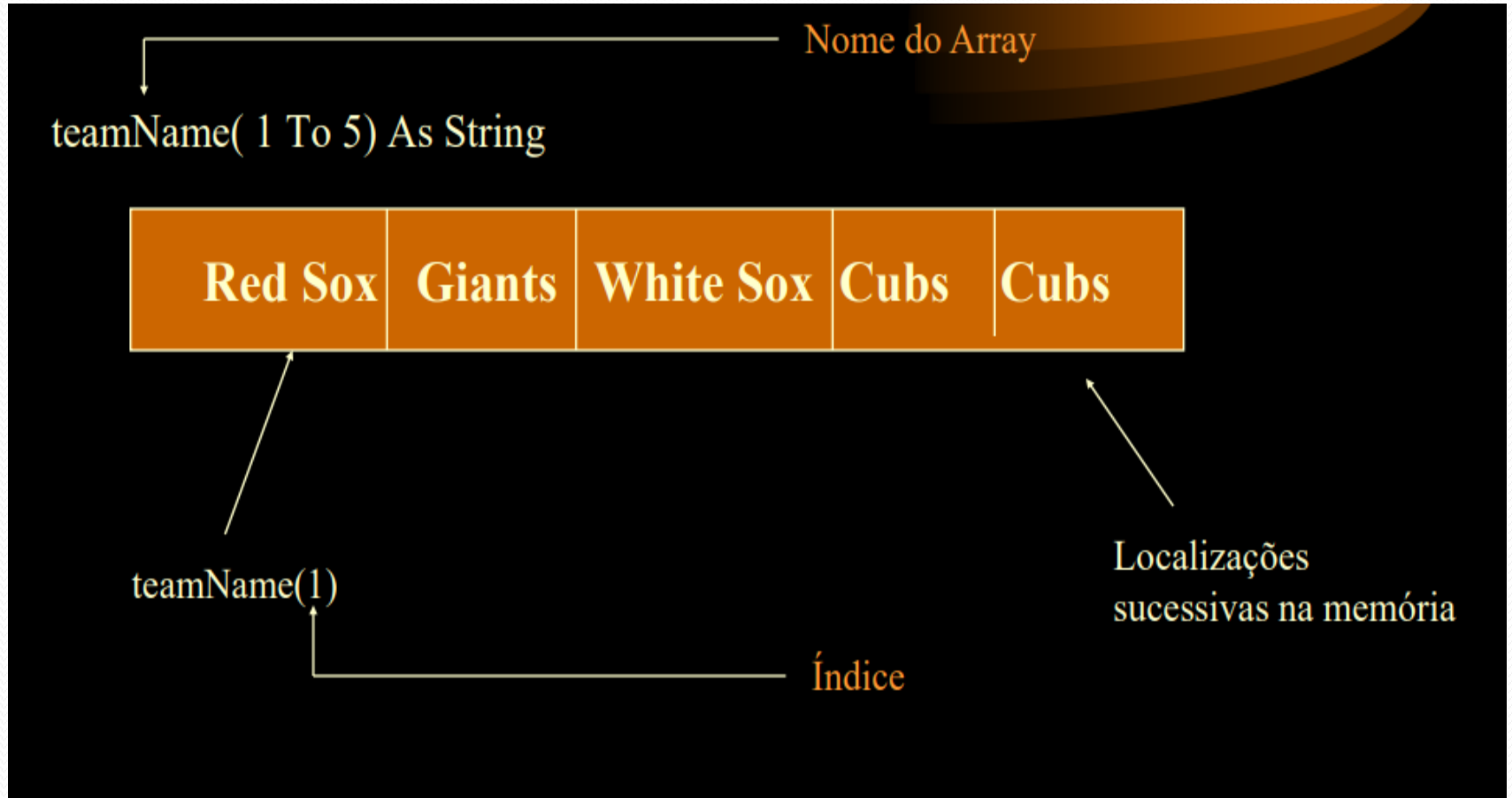


Vetores

- Uma boa analogia é comparar o vetor com uma tabela de tamanho fixo onde em cada linha pode ser armazenado um elemento.

posição	alfa	
0	3	alfa[0]
1	7	alfa[1]
2	8	alfa[2]
3	6	alfa[3]
4	1	alfa[4]

Vetores



Variáveis Compostas Homogêneas

- Interpretada como um conjunto (coleção) de valores de um mesmo tipo.
- Podem ser
 - Unidimensionais
 - Multidimensionais
- Requerem novos conceitos para serem manipuladas

Variáveis Compostas Unidimensionais

- São uma coleção caixinhas, onde cada caixinha guarda uma variável.
- Semelhante a uma coleção de gavetas do armário agrupadas.

[illegible]

Variáveis Compostas Unidimensionais

- Vetores (Arrays)
 - Tipo de dado usado para representar uma coleção de variáveis de um mesmo tipo.
 - Estrutura de dados homogênea e unidimensional.
 - Sintaxe: `tipo nome_do_vetor[tamanho];`
 - Tamanho representa o número de elementos.
 - O índice do vetor varia de 0 a (tamanho - 1)

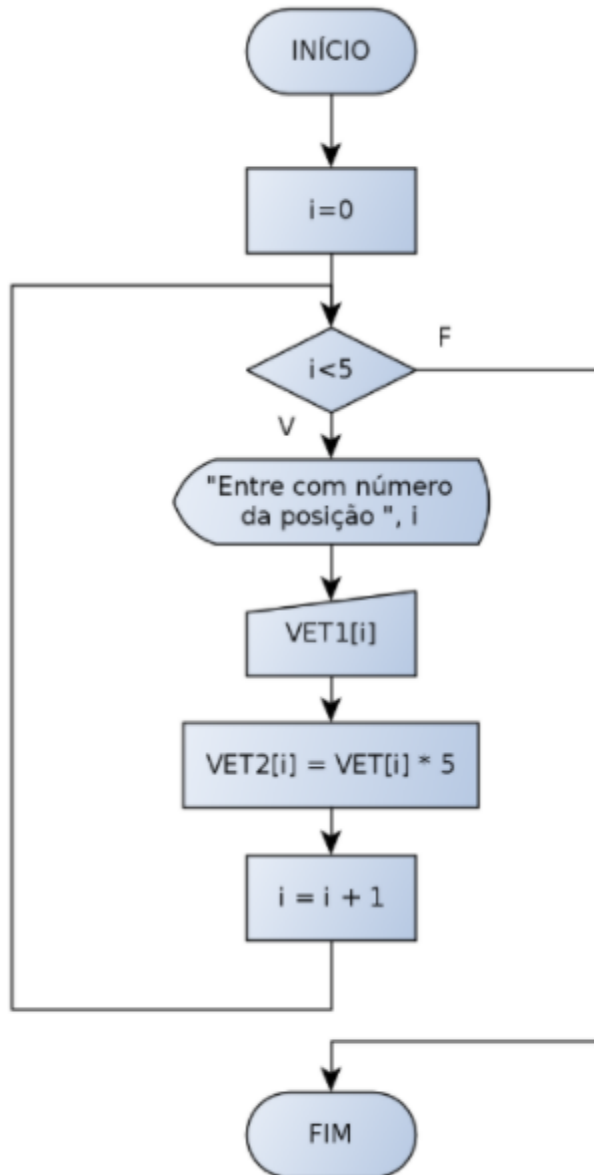
Vetores

- As variáveis são alocadas seqüencialmente na memória, onde o endereço mais baixo corresponde ao primeiro elemento (índice 0) do vetor.



x é um vetor unidimensional de n elementos.

Exemplo



i	0
VET1[0]	10
VET1[1]	2
VET1[2]	5
VET1[3]	14
VET1[4]	21
VET2[0]	50
VET2[1]	10
VET2[2]	25
VET2[3]	60
VET2[4]	105

A variável i é usada
Para indexar os vetores

$VET2[i] = VET1[i] * 5$

ou

$VET2[0] = VET1[0] * 5$

No primeiro loop

Arrays


- Exemplo
 - Nome do array ► **vet**
 - Número de elementos ► **12**

Nome do array

*Todos os elementos do array têm o mesmo nome, **vet***

Posição do elemento

Número que indica a posição do elemento no array acompanha o nome, entre colchetes



vet[0]	-45
vet[1]	6
vet[2]	0
vet[3]	72
vet[4]	1543
vet[5]	-89
vet[6]	0
vet[7]	62
vet[8]	-3
vet[9]	1
vet[10]	6453
vet[11]	78

Matrizes

- Matrizes funcionam de modo similar a vetores; porém são arrays multidimensionais.
- Declaração:
<tipo de dado> <identificador> [tamanho1,tamanho2,...]
- Exemplo:
- `int m[3][3]; /* declara uma matriz 3x3 */`
- `int n[3][4][5]; /* declara uma matriz tridimensional de tamanho 3x4x5 */`
- `double p[10][2]; /* declara uma matriz do tipo double de tamanho 10x2 */`

Matrizes

- `int a[3][3]={1,2,3,4,5,6,7,8,9};`
- `int a[3][3]={{1,2,3},{4,5,6},{7,8,9}}; /* Separa os dados de cada linha da matriz. Esta maneira é preferida. */`
- Em ambos os exemplos acima é criada a matriz:

1 2 3

4 5 6

7 8 9

Recursividade

- Se um problema pode ser resolvido facilmente
 - resolva o problema;
- Se o problema é grande,
 - elabore uma solução menor do problema,
 - relacione com o problema maior,
 - resolva o problema menor,
 - volte ao problema inicial.

Recursividade

- Um objeto é dito recursivo se ele consistir parcialmente ou for definido em termos de si mesmo.
- Uma função recursiva é uma função que faz uma chamada a si mesma.
- Uma função recursiva é definida em termos dela mesma
- Exemplos
 - Números naturais, Função fatorial, Árvore

Condição de parada

- Nenhum programa nem função pode ser exclusivamente definido por si
 - Um programa seria um loop infinito
 - Uma função teria definição circular
- Condição de parada
 - Permite que o procedimento pare de se executar
 - $F(x) > 0$ onde x é decrescente

Definições

- Listas: Uma estrutura que armazena elementos de forma alinhada, ou seja, com elementos dispostos um após o outro.
- Pilha: Trabalham com o algoritmo LIFO (last-in first-out), ou seja o último elemento inserido, será o primeiro retirado.
- Fila: Trabalham com o algoritmo FIFO (first-in-first-out) , ou seja, o primeiro elemento inserido é o primeiro retirado.
- Árvores: Diferente das listas encadeadas, em que os **dados** se encontram numa sequência, nas **árvores** os **dados** estão dispostos de forma hierárquica.

Contatos

- Email: fabio.silva321@fatec.sp.gov.br
- LinkedIn: <https://br.linkedin.com/in/b41a5269>
- Facebook: <https://www.facebook.com/fabio.silva.56211>

Contatos

- Email: fabio.silva321@fatec.sp.gov.br
- LinkedIn: <https://br.linkedin.com/in/b41a5269>
- Facebook: <https://www.facebook.com/fabio.silva.56211>