Nivelamento

Conceitos de Programação

ALGORITMOS

Introdução

O que é um Algoritmo?

Um algoritmo é um conjunto de etapas ordenadas que permite **resolver um problema com um obejtivo concreto**. Uma analogia feita geralmente é entre algoritmos e receitas culinárias.

Características:

- **Preciso**: Deve indicar claramente o que deve ser feito.
- **Finito**: Deve ter um número limitado de passos.
- **Definido**: Deve obter o mesmo resultado para as mesmas condições de entrada

Os algoritmos tentam resolver os problemas da realidade (preparar uma receita, calcular uma porcentagem, avaliar as condições, etc.).

Eles são um conjunto ordenado e finito de instruções que levam à solução do problema.

Podem ser classificados como :

- **Algoritmo computacional**: Pode ser resolvido por um computador ou dispositivo. Para isso, deve ser expresso em uma linguagem de programação. Os algoritmos expressos em uma linguagem de programação são chamados de "Programas".
- **Algoritmo não computacional:** Não pode ser resolvido por um computador ou dispositivo (como a construção de um móvel, a confecção de um bolo, etc.).

Todo algoritmo pode ser decomposto em três partes :

- **Entrada de dados**
- **Processamento**
- **Saida do resultado**

Vejamos alguns exemplos.

Exemplo: Receita de cozinha

ENTRADA:
2 laranjas
1 ovo
2 xícaras azúcar
1 xícara de farinha

INICIO:
Adicione todos os ingredientes no liquidificador.
Incorporar a mistura em uma assadeira com manteiga.
Asse por 40 minutos em forno moderado
FIM.

SAIDA:
O bolo está pronto para servir.

Este problema não pode ser resolvido por um computador, portanto, é um algoritmo não computacional.

Precisamos inserir os ingredientes (entrada de dados), processá-los (misturar os ingredientes, colocar na assadeira e assar).
Como saída (resultado), você obtém o bolo pronto para servir.

Exemplo: Converter uma quantidade de metros em centímetros

ENTRADA:
Quantidade M de metros.

INICIO:
Cálculo de centímetros: C = M * 100
FIM

SAIDA:
Quantidade C de centímetros.

Vamos ver agora um algoritmo para calcular em centímetros uma determinada quantidade de metros.

Este é um algoritmo computacional (pode ser resolvido por um computador) escrito em linguagem natural. Para que possa ser resolvido por um computador, ele deve ser escrito em uma linguagem de programação, mas também pode ser executado manualmente por uma pessoa.

A entrada de dados é a quantidade em metros, o processamento envolve o cálculo dos centímetros (por meio do cálculo correspondente) e a saída é o resultado em centímetros.

Algoritmos

Em **Software**, dizemos que os algoritmos, **resolvem problemas mais gerais**.

Exemplos:

- **Ordenar un conjunto de dados desordenados**
- **Calcular o dígito verificador de uma cédula**
- **Encriptação de uma senha**
- **Calcular a área de um polígono**

Podemos dizer que um 'Algoritmo' é un **conjunto ordenado de instruções que um computador deve interpretar** para executar una operação em particular.

GeneXus

Algoritmos

- **Especificação:** É a descrição do **Algoritmo em Alto Nível** respondendo à pergunta 'O que o Algoritmo faz?'.

- **Implementação:** É o **conjunto de instruções** em uma linguagem de programação que permite que o Algoritmo **cumpra seu objetivo declarado na Especificação**.

Para resumir, podemos dizer que:

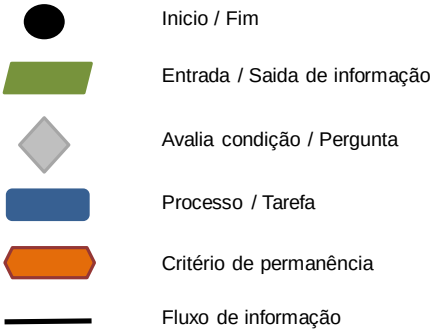
Problema: é um caso da realidade que você quer resolver.
Algoritmo: é o conjunto de instruções necessárias para resolvê-lo.
Programa: Se o algoritmo puder ser resolvido por um computador ou dispositivo (escrito em uma linguagem de programação)

DIAGRAMA DE FLUXO

GeneXus

O que é um Diagrama de fluxo?

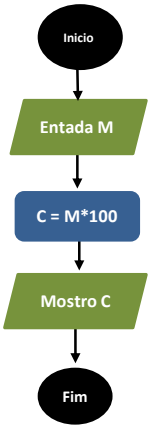
- É uma representação gráfica do algoritmo.
- Os passos pelos quais o algoritmo passa são representados por figuras.
- Permite uma fácil leitura.



O Diagrama de Fulxo é uma representação gráfica do algoritmo. Ele expressa com números os passos necessários para resolver o problema e permite uma leitura fácil.

Vamos ver alguns exemplos.

Exemplo: Converter uma quantidade de metros em centímetros



Inicio do algoritmo.

Entrada de dados: Entrada dos metros.

Processamento: cálculo Cms = Metros*100

Saida de dados: é mostrado o resultado C

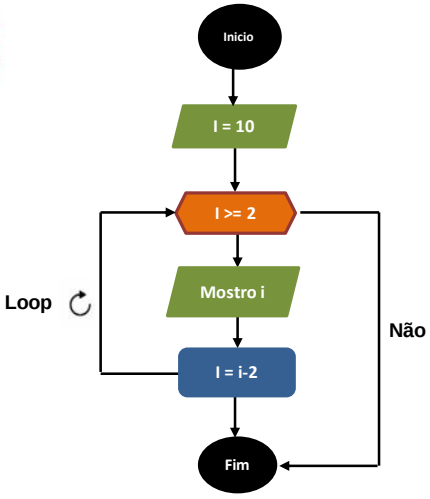
Fim do algoritmo.

Cada diagrama começa com o ponto de **Inicio**.

Entrada de dados: insira a quantidade em metros.
Processamento: o cálculo é feito * 100 metros
Saída de dados: o resultado C que representa a quantidade em centímetros é mostrado.

Fim do diagrama

Exemplo: Mostrar os números pares do 10 ao 2



Inicio do algoritmo.

Entrada de dados: i = 10

Criterio de permanência: enquanto for maior ou igual a 2

Saida de dados: mostra i

Processamento: atualiza o valor de i (se resta 2)

Fim del algoritmo.

Vamos ver outro exemplo de um algoritmo que pode ser resolvido por um computador ou dispositivo. Você deseja listar os números pares de 10 a 2

Nós temos então:
Início do diagrama de fluxo
Entrada de dados: insira a variável I com o valor 10. Posteriormente falaremos sobre o conceito de "variável".
Processamento: o critério de permanência é indicado "contanto que seja maior ou igual a 2". Se este critério (condição) for atendido, então I é mostrado e o valor das duas unidades restantes é atualizado. Se o resultado de I for menor que 2, o critério de permanência não será mais atendido e o fim será atingido imediatamente.

É importante mencionar que, diferentemente do “evento condicional", o "critério de permanência" avalia continuamente a condição. Isso significa que você define "loop".

PSEUDOCODIGO

GeneXus

O que é e para que serve?

É uma linguagem informal que permite:

- Especifique "em palavras" a solução para um problema.
- Concentra-se nos aspectos lógicos da solução, sem depender da sintaxe de uma linguagem de programação.
- Fácil detecção de erros.
- Rápida interpretação
- Liberdade para o programador. Por não ser uma linguagem formal, cada programador atinge sua melhor versão.

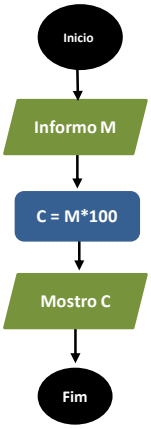
O objetivo do Pseudocódigo é permitir que o programador se concentre nos aspectos lógicos sem depender de uma linguagem de programação.

Qualquer um pode facilmente interpretar as etapas pelas quais um algoritmo passa.

Como não é uma linguagem formal, o pseudocódigo varia de um programador para outro

Vamos ver alguns exemplos.

Exemplo: Converter uma quantidade de metros em centímetros



INICIO

Informo quantidade M de metros.

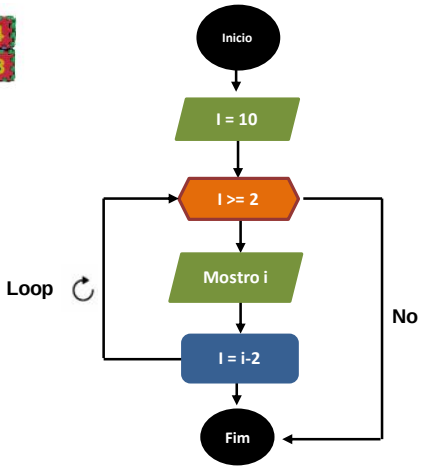
Cálculo: C = M * 100

Mostrar: quantidade C de centímetros

FIM

.

Ejemplo: Mostrar os números pares do 10 ao 2



INICIO

i = 10

Enquanto i >= 2

Mostro i
i = i-2

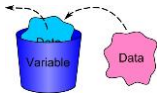
Fim enquanto

FIM

.

VARIÁVEIS

O que são?



- Uma variável é um espaço na memória.
- Armazena um valor que pode mudar durante a execução do programa.
- Tem um nome e um tipo de dados.

Exemplos:.

Idade	–	Numérico (inteiro)
Nome	–	Caracter
Data nascimento	–	Data
É verde?	–	Boolean (Verdadeiro / Falso)

Quando um programa precisa armazenar um dado na memória, ele precisa de uma "variável".

Uma variável é uma localização na memória principal que armazena um valor que pode mudar durante o curso do programa.

Cada variável tem:
Nome
Tipo de dados

E depois você pode atribuir um valor a ela. Antes de poder usar uma variável, é necessário declarar esses conceitos.

Ao contrário das variáveis, todos esses valores que aparecem no pseudocódigo são chamados de "**literais**".

Pode ser:
Literais inteiros
Literais reais
Caracter literal
Data literal
Literais lógicos

TIPOS DE DADOS

TIPOS DE DADOS

Simples

- Os tipos de dados simples são aqueles que representam um valor único e individual como, por exemplo, um número, uma string ou um valor booleano.

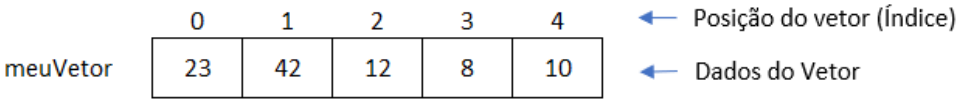
Compostos o Estruturados

- Os tipos de dados compostos permitem armazenar um conjunto de valores, onde cada valor é de um tipo de dados simples.

Exemplos de estruturas que podem ser criadas com este tipo de dados são os arrays ou as coleções. Quando os arrays têm apenas uma dimensão, são chamados vetores.

VETORES

- Um vetor é um conjunto de dados do mesmo tipo, como: Inteiros, Caractere, Booleano, etc., ordenados como uma linha de N elementos.
- Cada elemento ou dado está em uma posição (índice). Estes índices são números inteiros positivos. O índice do primeiro elemento sempre será 0.



Ex: meuVetor[2] tem o valor 12

COLEÇÕES

- Uma coleção é uma estrutura que permite armazenar um conjunto de elementos, todos do mesmo tipo.

A principal diferença entre um vetor e uma coleção é como atribuímos seu tamanho. Enquanto ao criar um vetor, temos que definir antecipadamente
quantos elementos irá conter, em uma coleção isto não é necessário e podem ser adicionados elementos quando necessário, ajustando-se automaticamente o tamanho. As coleções também possuem métodos (funções) que permitem trabalhar com elas mais facilmente.

EXPRESSÕES ARITMÉTICAS

GeneXus

O que são?

$$z = \frac{3x}{2w} - \frac{8x^2 - w}{3}$$

Uma expressão aritmética é uma combinação de variáveis, literais e operadores aritméticos.

Operadores aritméticos:

Soma	- a+b
Resto	- a-b
Multiplicação	- a*b
Divisão	- a/b

Operadores relacionais:

Maior que	- a > b
Maior ou igual que	- a >= b
Menor que	- a < b
Menor ou igual que	- a <= b
Igual a	- a = b

Operadores lógicos:

No	- not a
Y	- a and b
Ó	- a or b

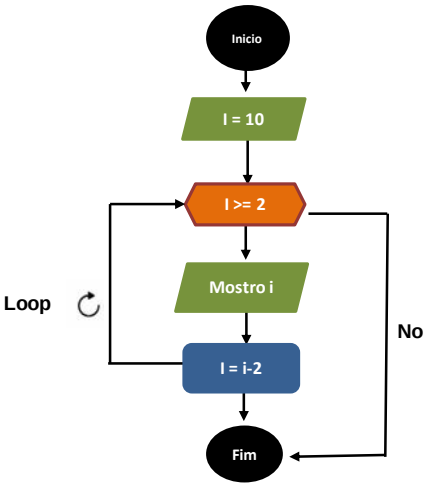
Existem diferentes expressões aritméticas, e nelas podem ser usados diferentes "**operadores**":

Operadores Airtméticos
Operadores Relacionais
Operadores Lógicos

Vamos ver os operadores envolvidos no exemplo da lista de números pares.

GeneXus

Exemplo: Mostrar os números pares de 10 a 2



Variáveis
I – Numérico 2

Literal (valores fixos)
10
2

Operador Aritmético
-

Operador Relacional
>=

Expressões Aritméticas
I = I – 2

GeneXus

Tabelas da verdade

Permitem avaliar mais de uma condição lógica

a	b	a AND b
Falso	Falso	**Falso**
Falso	Verdadeiro	**Falso**
Verdadeiro	Falso	**Falso**
Verdadeiro	Verdadeiro	**Verdadeiro**

a	b	a OR b
Falso	Falso	**Falso**
Falso	Verdadeiro	**Verdadeiro**
Verdadeiro	Falso	**Verdadeiro**
Verdadeiro	Verdadeiro	**Verdadeiro**

a	NOT a
Falso	**Verdadeiro**
Verdadeiro	**Falso**

Quando você avalia mais de uma condição lógica, você tem "tabelas da verdade".

Operador "Y" (AND): Ambas as condições devem ser atendidas para que o resultado seja True.
Operador "O" (OR): Se ambas as condições forem atendidas ou apenas uma condição for atendida, o resultado será True.
Operador "NO" (NOT): O resultado oposto é obtido (a negação do valor).

INSTRUÇÕES ALGORÍTMICAS

O que são e para que servem?

São expressões que nos permitem abordar uma linguagem compreensível por um computador.

Entrada de dados.

Processamento (corpo do algoritmo).

Saida de resultado.

As **"Instruções algorítmicas"** permitem nos aproximarmos cada vez mais de uma linguagem compreensível por um computador.

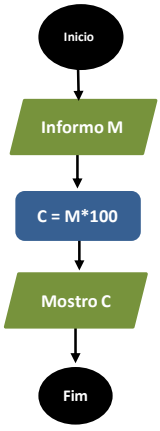
Existem diferentes instruções dependendo da etapa do algoritmo:
Entrada de dados: A ação de inserir dados em uma variável é geralmente expressa pela palavra "read". Por exemplo, a instrução "read M" ou "enter M" solicita a entrada de um valor por meio de um dispositivo de entrada (como o teclado), para salvá-lo na variável M.
Processamento ou corpo do algoritmo
Saída do resultado: consiste em mostrar o valor de uma variável em um dispositivo de saída, como uma tela. Em geral, a ação de exibir um valor de uma variável é expressa no pseudocódigo como "show M" ou "display M".

Atribuição

Consiste em atribuir o valor de uma expressão a uma variável.

Variável = Expressão

Deben tener el mismo tipo de dato.



INICIO

Informo quantidade M de metros.

Cálculo: C = M * 100

Mostrar: quantidade C de centímetros

FIM

.

A expressão designada pode ser uma variável, um literal ou uma combinação de variáveis, literais e operadores.

Deve-se levar em conta que a variável e o resultado da expressão designada devem ter o mesmo tipo de dados. Isso significa que, por exemplo, uma variável declarada do tipo date não pode ser atribuída a um valor numérico ou caracter.

Condicional simples

Permite avaliar se uma condição é cumprida ou não, e decidir a ação a ser tomada.

```
If (condição)
   O que é feito se a condição é cumprida
Else
   O que é feito se a condição não é cumprida
Endif
```

Ejemplo:

```
If  Idade >= 18 and Passaporte = "Válido"
      Msg("Pode viajar")
Else
      Msg("Não pode viajar")
Endif
```

Verificação de casos

É executada a primeira condição que é cumprida e nenhuma outra.

Do case

Case (Condição 1)
Case (Condição 2)

Otherwise

O que é feito se nenhuma condição é cumprida.

EndCase

Exemplo:

Do case

Case Nota >= 70 “Exonerado”
Case Nota >= 60 “Regulamentado”
Case Nota >= 50 “Livre”

Otherwise

“Deve repetir o curso”

EndCase

Iteração

Permite avançar de um valor para outro.

```
For (valor inicial) to (valor final) [step 1]

        ----------------------

EndFor
```

Exemplo:

```
For i=2 to 10 [step 2]

    Mostrar i

EndFor
```

While (Loop)

Entra-se em um loop quando a ação é executada enquanto uma condição é atendida.

```
Do while (condição)

----------

EndDo
```

Exemplo:

```
I = 10

Do while i >= 2

    Mostro i
    i = i-2

EndDo
```

GeneXus™

Videos	training.genexus.com
Documentation	wiki.genexus.com
Certificactions	training.genexus.com/certifications