Nivelación

Conceptos de Programación

GeneXus

ALGORITMOS

Introducción

¿Qué es un Algoritmo?

Un algoritmo es un conjunto de pasos ordenados que permite **resolver un problema con un objetivo concreto**. Suele realizarse una analogía entre los Algoritmos y las recetas de cocina.

Características:

- **Preciso**: Debe indicar claramente lo que se debe hacer.
- **Finito**: Debe tener un número limitado de pasos.
- **Definido**: Se debe obtener el mismo resultado para las mismas condiciones de entrada

Los Algoritmos intentan resolver problemas de la realidad (cocinar una receta, resolver un porcentaje, evaluar condiciones, etc).

Son un conjunto ordenado y finito de instrucciones que conducen a la solución del problema.

Se pueden clasificar en:

- **Algoritmo computacional**: Puede ser resuelto por una computadora o dispositivo. Para eso debe ser expresado en un lenguaje de programación. Los algoritmos expresados en un lenguaje de programación se denominan “Programas”.
- **Algoritmo no computacional:** No puede ser resuelto por una computadora o dispositivo (como por ejemplo armar un mueble, cocinar una torta, etc).

Todo algoritmo se puede descomponer en tres partes:

- **Entrada de datos**
- **Procesamiento**
- **Salida de resultado**

Veamos algunos ejemplos.

Ejemplo: Receta de cocina

ENTRADA:
2 naranjas
1 huevo
2 tazas azúcar
1 taza harina

INICIO:
Incorporar todos los ingredients en una licuadora.
Incorporar la mezcla en una asadera enmantecada.
Hornear por 40 minutos a horno moderado
FIN.

SALIDA:
La torta está lista para servir.

Este problema no puede ser resuelto por una computadora, por lo tanto se trata de un algoritmo no computacional.

Necesitamos ingresar los ingredientes (entrada de datos), procesarlos (mezclar los ingredientes, poner en la asadera y hornear).
Como salida (resultado) se obtiene la torta pronta para servir.

Ejemplo: Convertir una cantidad de metros a centímetros

ENTRADA:
Cantidad M de metros.

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

SALIDA:
Cantidad C de centímetros.

Veamos ahora un algoritmo para expresar en centímetros una determinada cantidad en metros.

Este es un algoritmo computacional (puede ser resuelto por una computadora) escrito en lenguaje natural. Para que pueda ser resuelto por una computadora debe ser escrito en un lenguaje de programación, pero también puede ser ejecutado manualmente por una persona.

La entrada de datos es la cantidad en metros, el procesamiento implica el cálculo de los centímetros (mediante el cálculo correspondiente), y la salida es el resultado en centímetros.

Algoritmos

En **Software**, decimos que los algoritmos, r**esuelven problemas más generales**.

Ejemplos:

- **Ordenar un conjunto de datos desordenados**
- **Calcular el dígito verificador de la cédula**
- **Encriptación de una contraseña**
- **Calcular el área de un polígono**

Podemos decir que un 'Algoritmo', es un **conjunto ordenado de instrucciones que debe interpretar un computador** con el fin de ejecutar una operación en particular.

Algoritmos

- **Especificación:** Es la descripción en **Alto Nivel del Algoritmo** respondiendo a la pregunta '¿Qué hace el Algoritmo?'.

- **Implementación:** Es el **conjunto de instrucciones** en un lenguaje de programación que permiten al Algoritmo **lograr su objetivo planteado en la Especificación**.

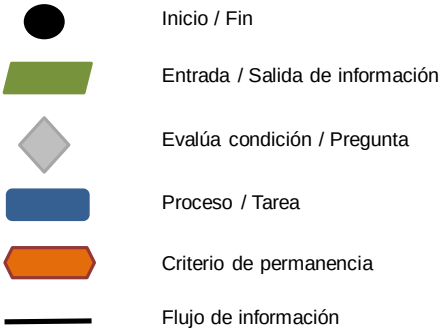
A modo de resumen, podemos decir que :

Problema: es un caso de la realidad que se quiere resolver.
Algoritmo: es el conjunto de instrucciones necesarias para resolverlo.
Programa: Si el algoritmo puede ser resuelto por una computadora o dispositivo (escrito en un lenguaje de programación)

DIAGRAMA DE FLUJO

¿Qué es un Diagrama de flujo?

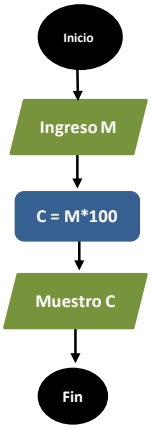
- Es una representación gráfica del algoritmo.
- Los pasos por los que pasa el algoritmo se representan con figuras.
- Permite una fácil lectura.



El Diagrama de Flujo es una representación gráfica del algoritmo. Se expresa con figuras los pasos necesarios para resolver el problema, y permite una fácil lectura.

Veamos algunos ejemplos.

Ejemplo: Convertir una cantidad de metros a centímetros



Inicio del algoritmo.

Entrada de datos: Ingreso de los metros.

Procesamiento: cálculo Cms = Metros*100

Salida de datos: se muestra el resultado C

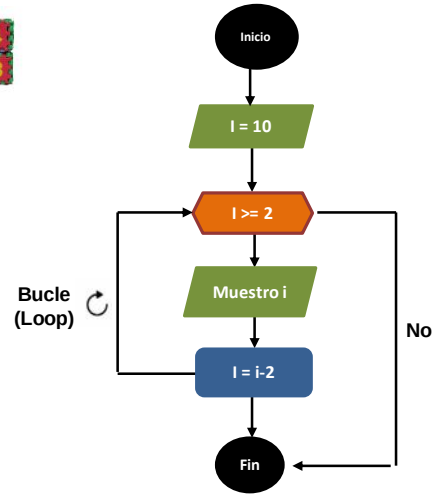
Fin del algoritmo.

Todo diagrama comienza con el punto de **Inicio**.

Entrada de datos: Ingreso de la cantidad en metros.
Procesamiento :se realiza el cálculo metros * 100
Salida de datos: se muestra el resultado C que representa la cantidad en centímetros.

Fin del diagrama.

Ejemplo: Mostrar los números pares del 10 al 2



Inicio del algoritmo.

Entrada de datos: i = 10

Criterio de permanencia: mientras i sea mayor o igual a 2

Salida de datos: se muestra i

Procesamiento: se actualiza el valor de i (se resta 2)

Fin del algoritmo.

Veamos otro ejemplo de un algoritmo que puede ser resuelto por una computadora o dispositivo. Se quieren listar los números pares desde el 10 hasta el 2

Tenemos entonces:
Inicio del diagrama de flujo
Entrada de datos: se ingresa la variable i con el valor 10. Más adelante hablaremos del concepto de "variable".
Procesamiento: se indica el criterio de permanencia "mientras i sea mayor o igual a 2". Si este criterio (condición) se cumple entonces se muestra i, y se actualiza el valor restándole dos unidades. Si el resultado de i es menor que 2 entonces ya no se cumple el criterio de permanencia y se llega al final directamente.

Es importante mencionar que a diferencia del "rombo condicional", el "criterio de permanencia" evalúa continuamente la condición. Esto significa que se entra en un "bucle" o "loop".

PSEUDOCODIGO

GeneXus

¿Qué es y para qué sirve?

Es un lenguaje informal que permite:

- Especificar “en palabras” la solución de un problema.
- Concentrarse en los aspectos lógicos de la solución, sin depender de la sintaxis de un lenguaje de programación.
- Fácil detección de errores.
- Rápida interpretación
- Libertad para el programador. Al no ser un lenguaje formal, cada programador llega a su mejor versión.

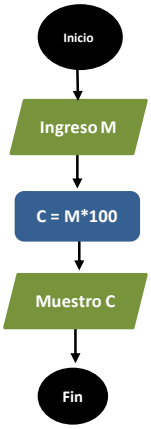
El objetivo del Pseudocódigo es permitir que el programador se concentre en los aspectos lógicos sin tener que depender de un lenguaje de programación.

Cualquier persona puede interpreter fácilmente los pasos por los que transcurre un algoritmo.

Al no tratarse de un lenguaje formal, el pseudocódigo varía de un programador a otro

Veamos algunos ejemplos.

Ejemplo: Convertir una cantidad de metros a centímetros



INICIO

Ingreso cantidad M de metros.

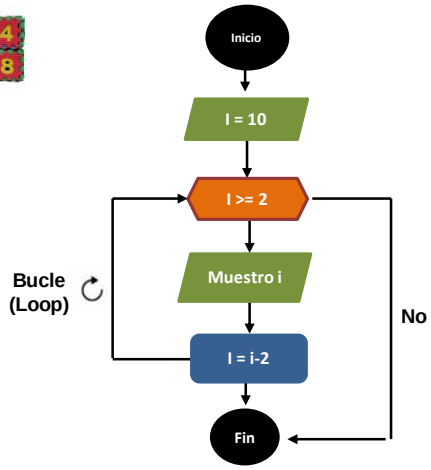
Cálculo: C = M * 100

Desplegar: cantidad C de centímetros

FIN

.

Ejemplo: Mostrar los números pares del 10 al 2



INICIO

i = 10

Mientras i >= 2

Muestro i
i = i-2

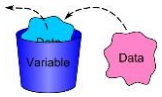
Fin mientras

FIN

.

VARIABLES

¿Qué son?



- Una variable es un espacio en memoria.
- Almacena un valor que puede cambiar en el transcurso de la ejecución del programa.
- Tiene un nombre y un tipo de dato.

Ejemplos:.

Edad	–	Numérico (entero)
Nombre	–	Carácter
Fecha nacimiento	–	Fecha
¿Es verde?	–	Boolean (Verdadero / Falso)

Cuando un programa necesita almacenar un dato en memoria, necesita una "**variable**".

Una variable es una localización en la memoria principal que almacena un valor que puede cambiar durante el transcurso del programa.

Toda variable tiene:
Nombre
Tipo de dato

Y posteriormente se le podrá asignar un valor. Antes de poder utilizar una variable es necesario declarar dichos conceptos.

A diferencia de las variables, se denominan "**literales**" a todos aquellos valores fijos que figuran en el pseudocódigo.

Pueden ser:
- Literales enteros
- Literales reales
- Literales character
- Literales fecha
- Literales lógicos

TIPOS DE DATOS

TIPOS DE DATOS

Simples

- Los tipos de datos **simples**, son aquellos que representan un valor único e individual, como por ejemplo un número, una cadena o un valor booleano.

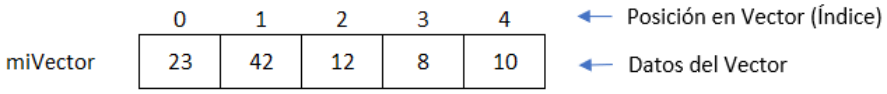
Compuestos o Estructurados

- Los tipos de datos compuestos permiten almacenar un conjunto de valores, donde cada valor es de un tipo de datos simple.

Ejemplos de estructuras que pueden crearse con este tipo de datos son los arrays (arreglos) o las colecciones. Cuando los arrays tienen una sola dimensión, se llaman vectores.

VECTORES

- Un vector es un conjunto de datos del mismo tipo como: Enteros, Character, Booleano, etc. Ordenados como una fila de N elementos.
- Cada elemento o dato, se encuentra en una posición (índice). Estos índices son números enteros positivos. El índice del primer elemento siempre será 0



Ejemplo: miVector[2] tiene el valor 12

COLECCIONES

- Una colección es una estructura que permite almacenar un conjunto de elementos, todos del mismo tipo.

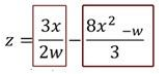
La principal diferencia entre un vector y una colección es cómo asignamos su tamaño. Mientras que al crear un vector tenemos que definir de antemano cuántos elementos va a contener, en una colección esto no es necesario y se pueden agregar elementos cuando se necesite ajustándose el tamaño automáticamente.

Las colecciones además disponen de métodos (funciones) que permiten trabajar con ellas en forma más sencilla.

EXPRESIONES ARITMETICAS

GeneXus

¿Qué son?



Una expresión aritmética es una combinación de variables, literales y operadores aritméticos.

Operadores aritméticos:

Suma	-	a+b
Resta	-	a-b
Multiplicación	-	a*b
División	-	a/b

Operadores relacionales:

Mayor que	-	a > b
Mayor o igual que	-	a >= b
Menor que	-	a < b
Menor o igual que	-	a <= b
Igual a	-	a = b

Operadores lógicos:

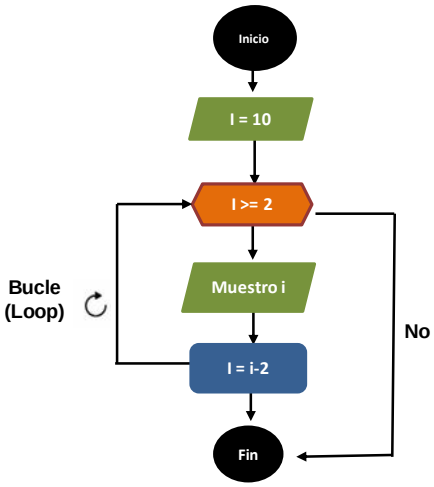
No	-	not a
Y	-	a and b
Ó	-	a or b

Existen diferentes expresiones aritméticas, y en ellas participan diferentes "**operadores**":

- Operadores Airtméticos
- Operadores Relacionales
- Operadores Lógicos

Veamos los operadores que participan en el ejemplo del listado de números pares.

Ejemplo: Mostrar los números pares del 10 al 2



Variables
I – Numérico 2

Literal (valores fijos)
10
2

Operador Aritmético
-

Operador Relacional
>=

Expresiones Aritméticas
I = I – 2

GeneXus

Tablas de verdad

Permiten evaluar más de una condición lógica

a	b	a AND b
Falso	Falso	**Falso**
Falso	Verdadero	**Falso**
Verdadero	Falso	**Falso**
Verdadero	Verdadero	**Verdadero**

a	b	a OR b
Falso	Falso	**Falso**
Falso	Verdadero	**Verdadero**
Verdadero	Falso	**Verdadero**
Verdadero	Verdadero	**Verdadero**

a	NOT a
Falso	**Verdadero**
Verdadero	**Falso**

Cuando se evalúa más de una condición lógica se cuenta con "**Tablas de verdad**".

- **Operador "Y" (AND):** Ambas condiciones deben cumplirse para que el resultado sea Verdadero.
- **Operador "O" (OR):** Si ambas condiciones se cumplen, o se cumple una sola, entonces el resultado será Verdadero.
- **Operador "NO" (NOT):** Se obtiene el resultado contrario (la negación del valor).

INSTRUCCIONES ALGORITMICAS

¿Qué son y para qué sirven?

Son expresiones que permiten acercarnos a un lenguaje comprensible por una computadora.

Entrada de datos.

Procesamiento (cuerpo del algoritmo).

Salida de resultado.

Las “**instrucciones algorítmicas**” permiten que nos acerquemos cada vez más a un lenguaje comprensible por una computadora.

Hay diferentes instrucciones dependiendo de la etapa del algoritmo:

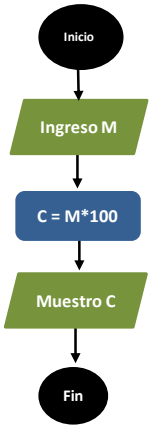
- **Entrada de datos:** La acción de ingresar un dato a una variable generalmente se expresa mediante la palabra “leer”. Por ejemplo la instrucción “leer M” o “ingresar M” solicita el ingreso de un valor mediante un dispositivo de entrada (como puede ser el teclado), para guardarlo en la variable M.
- **Procesamiento o cuerpo del algoritmo**
- **Salida de resultado:** Consiste en mostrar el valor de una variable en un dispositivo de salida como puede ser una pantalla. En general la acción de mostrar un valor de una variable se expresa en el pseudocódigo como “mostrar M” o “desplegar M”.

Asignación

Consiste en asignar a una variable el valor de una expresión.

Variable = Expresión

Deben tener el mismo tipo de dato.



INICIO

Ingreso cantidad M de metros.

Cálculo: C = M * 100

Desplegar: cantidad C de centímetros

FIN

.

La expression que se asigna puede ser una variable, un literal, o una combinación de varialbes, literales y operadores.

Se debe tener en cuenta que la variable y el resultado de la expression que se asigna deben tener el mismo tipo de dato. Esto significa que por ejemplo a una variable declarada de tipo fecha no se le puede asignar un valor numérico o character.

Condicional simple

Permite evaluar si una condición se cumple o no, y decidir la acción a tomar.

```
If (condición)
   Lo que se hace si la condición se cumple
Else
   Lo que se hace si la condición no se cumple
Endif
```

Ejemplo:

```
If  Edad >= 18 and Cédula = “Vigente”
      Msg(“Puede viajar”)
Else
      Msg(“No puede viajar”)
Endif
```

Chequeo de casos

Se ejecuta la primera condición que se cumpla, y ninguna otra.

```
Do case

    Case (Condición 1)
    Case (Condición 2)
    ---------

Otherwise

   Lo que se hace si no se cumple
    ninguna condición.

EndCase
```

Ejemplo:

```
Do case

    Case  Nota >= 70   “Exonerado”
    Case  Nota >= 60   “Reglamentado”
    Case  Nota >= 50   “Libre”

Otherwise

    “Debe repetir el curso”

EndCase
```

Iteración

Permite avanzar desde un valor hacia otro.

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

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

EndFor
```

Ejemplo:

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

    Mostrar i

EndFor
```

Bucle (Loop)

Se entra en un bucle cuando la acción se realiza mientras se cumpla una condición.

```
Do while (condición)

----------

EndDo
```

Ejemplo:

```
I = 10

Do while i >= 2

    Muestro i
    i = i-2

EndDo
```

GeneXus™

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