

Knowledge Leveling

Programming Concepts

ALGORITHMS

Introduction

What is an Algorithm?



An algorithm is a set of ordered steps that allows you to **solve a problem with a specific objective**. An analogy is often made between Algorithms and recipes.

Characteristics:

- **Precise:** It must clearly indicate what is to be done.
- **Finite:** It must have a limited number of steps.
- **Defined:** The same result must be obtained for the same input conditions.

Algorithms try to solve problems of reality (cooking a recipe, finding a percentage, evaluating conditions, etc).

They are an orderly and finite set of instructions that lead to the solution of the problem.

They can be classified into:

- **Computational algorithm:** It can be solved by a computer or device. To this end, it must be expressed in a programming language. Algorithms expressed in a programming language are called "Programs."
- **Non-computational algorithm:** It cannot be solved by a computer or device (such as assembling a piece of furniture, baking a cake, etc.).

Every algorithm can be broken down into three parts:

- **Data Entry**
- **Processing**
- **Result output**

Let's see some examples.

Example: Recipe

**INPUT:**

2 oranges
1 egg
2 cups sugar
1 cup flour

**START:**

Incorporate all ingredients into a blender.
Pour the mixture into a buttered baking dish.
Bake for 40 minutes in a moderate oven.

END.

**OUTPUT:**

The cake is ready to serve.

This problem cannot be solved by a computer, so it is a non-computational algorithm.

We need to enter the ingredients (input), process them (mix the ingredients, put them in the baking dish and bake).

The output (result) is the cake ready to serve.

Example: Convert a number of meters to centimeters

**INPUT:**

Quantity M of meters.

**START:**Calculation of centimeters: $C = M * 100$ **END****OUTPUT:**

Quantity C of centimeters.

Let's now look at an algorithm to express in centimeters a certain amount in meters.

This is a computational algorithm (it can be solved by a computer) written in natural language. In order to be solved by a computer it must be written in a programming language, but it can also be executed manually by a person.

The input data is the quantity in meters, the processing involves the calculation of centimeters (using the corresponding calculation), and the output is the result in centimeters.

Algorithms

In **Software**, we say that algorithms **solve more general problems**.

Examples:

- **Sorting a cluttered data set**
- **Finding the verification digit of an Identification card**
- **Encrypting a password**
- **Calculating the area of a polygon**

We can say that an 'Algorithm' is an **ordered set of instructions that a computer must interpret** in order to execute a particular operation.

Algorithms

- **Specification:** Is the **High Level description of the Algorithm** to answer the question 'What does the Algorithm do?.'
- **Implementation:** Is the **set of instructions** in a programming language that allow the Algorithm to **achieve its objective set out in the Specification.**

In summary, we can say that:

Problem: is a case of reality that you want to solve.

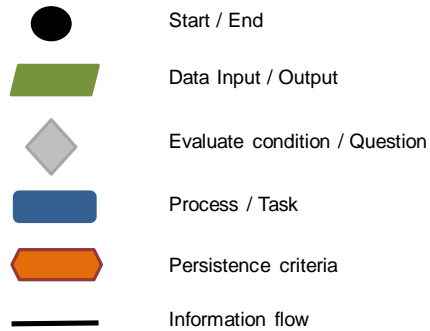
Algorithm: is the set of instructions needed to solve it.

Program: whether the algorithm can be solved by a computer or device (written in a programming language).

FLOW CHART

What is a Flow Chart?

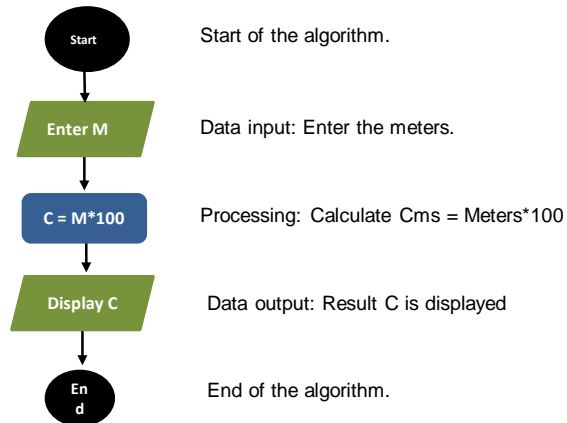
- It's a graphical representation of the algorithm.
- The steps that the algorithm goes through are represented by figures.
- It is easy to read.



The Flow Chart is a graphical representation of the algorithm. The necessary steps to solve the problem are expressed using images. It allows for easy reading.

Let's see some examples.

Example: Convert a number of meters to centimeters



Every diagram begins with the **Start** point.

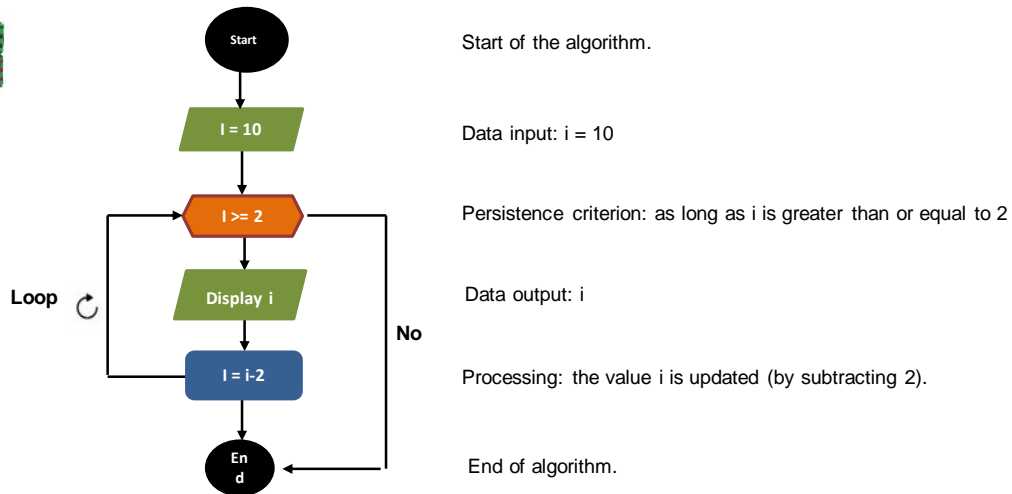
Data input: Entering the number of meters.

Processing: the calculation of meters * 100 is performed

Data output: the result C representing the quantity in centimetres is displayed.

End of chart.

Example: Show even numbers from 10 to 2



Let's look at another example of an algorithm that can be solved by a computer or device. You want to list the even numbers from 10 to 2.

We have:

Start of flow chart

Data input: variable i is entered with value 10. We will discuss the concept of "variable" later.

Processing: the persistence criterion "as long as i is greater than or equal to 2" is indicated. If this criterion (condition) is met then i is displayed, and the value is updated by subtracting two units. If the result of i is lower than 2, the persistence criterion is no longer met and the end result is reached directly.

It is important to note that unlike the "conditional diamond," the "persistence criterion" continuously assesses the condition. This means that a loop is entered.

PSEUDOCODE

What is it and what is it for?



It is an informal language that allows you to:

- Specify "in words" the solution of a problem.
- Focus on the logical aspects of the solution, without relying on the syntax of a programming language.
- Easy error detection.
- Quick interpretation.
- Freedom for the programmer. Since it isn't a formal language, each programmer reaches his or her best version.

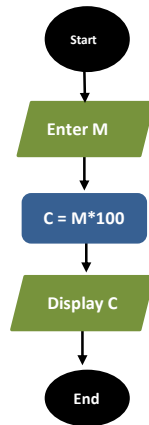
The purpose of Pseudocode is to allow the programmer to focus on the logical aspects without having to rely on a programming language.

Anyone can easily interpret the steps an algorithm goes through.

Since it isn't a formal language, the pseudocode varies from one programmer to another.

Let's see some examples.

Example: Convert a number of meters to centimeters



START

Enter quantity M of meters.

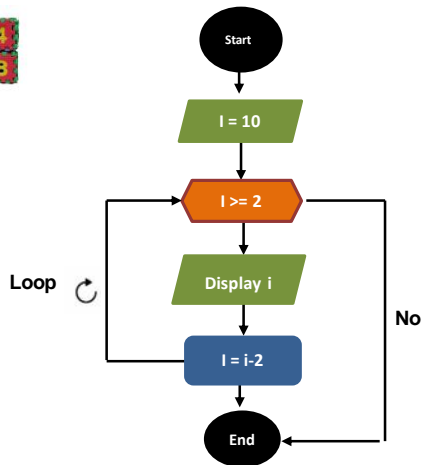
Calculation: $C = M * 100$

Display: quantity C of centimeters

END

.

Example: Show even numbers from 10 to 2



START

i = 10

As long as i >= 2

Display i

i = i-2

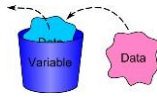
End while

END

.

VARIABLES

What are they?



- A variable is a memory space.
- It stores a value that may change during program execution.
- It has a name and a data type.

Examples:

Age - Numerical (integer)

Name - Character

Date of birth - Date

Is it green? – Boolean (True / False)

When a program needs to store data in memory, it needs a "**variable.**"

A variable is a location in the main memory that stores a value that may change during the course of the program.

Every variable has:

Name

Data type

It can then be assigned a value. Before you can use a variable, you need to declare these concepts.

Unlike the variables, all the values in the pseudocode are referred to as "**literals.**"

They may be:

- Full literals
- Real literals
- Character literals
- Date literals
- Logical literals

DATA TYPES

DATA TYPES

Simple

- Simple data types are those that represent a unique single value, such as a number, a string, or a Boolean value.

Compound or Structured

- Compound data types allow you to store a set of values, where each value is of a simple data type. Examples of structures that can be created with this data type are arrays or collections. When arrays have only one dimension, they are called vectors.

VECTORS

- A vector is a set of data of the same type, such as: Integer, Character, Boolean, etc., ordered as a row of N elements.
- Each element or data item is in one position (index). These indexes are positive integers. The index of the first element will always be 0.

	0	1	2	3	4	← Position in the Vector (Index)
myVector	23	42	12	8	10	← Vector Data

e.g.: myVector[2] has the value 12.

COLLECTIONS

- A collection is a structure that allows storing a set of elements, all of the same type.



The main difference between a vector and a collection is how it is sized. When creating a vector you have to define beforehand how many elements it will contain. In a collection, this is not necessary and elements can be added as needed by adjusting the size automatically. Collections also have methods (functions) that make it easier to work with them.

ARITHMETIC EXPRESSIONS

What are they?

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

An arithmetic expression is a combination of variables, literals and arithmetic operators.

Arithmetic operators:

Sum - $a+b$
Subtraction - $a-b$
Multiplication - $a*b$
Division - a/b

Relational operators:

Greater than - $a > b$
Greater than or equal to - $a >= b$
Lower than - $a < b$
Lower than or equal to - $a <= b$
Equal to - $a = b$

Logical operators:

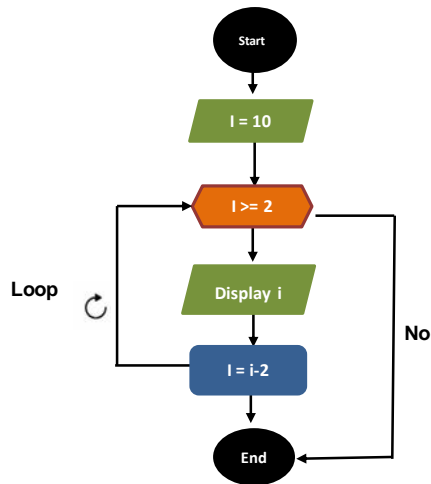
No - not a
Y - a and b
O - a or b

There are different arithmetic expressions, and different "**operators**" participate in them:

- Arithmetic operators
- Relational operators
- Logical operators

Let's see the operators that participate in the example of the list of even numbers.

Example: Show even numbers from 10 to 2

**Variables**

i - Numeric 2

Literal (fixed values)

10

2

Arithmetic Operator

-

Arithmetic Expressions

$i = i - 2$

Relational Operator

\geq

Tables of Truth

They allow you to evaluate more than one logical condition

a	b	a AND b	a	b	a OR b	a	NOT a
False	False	False	False	False	False	False	True
False	True	False	False	True	True	True	False
True	False	False	True	False	True		
True	True	True	True	True	True		

When more than one logical condition is evaluated, the "**Tables of Truth**" are available.

- **Operator "AND"**: Both conditions must be met for the result to be True.
- **Operator "OR"**: If both conditions are met, or only one is met, then the result will be True.
- **Operator "NO"**: The opposite result is obtained (the negation of the value).

ALGORITHMIC INSTRUCTIONS

What are they and what are they for?



They are expressions that allow us to approach a language that is understandable by a computer.



Data input.



Processing (body of the algorithm).



Result output.


“Algorithmic instructions” allow us to get closer to a language that can be understood by a computer.

There are different instructions depending on the stage of the algorithm:

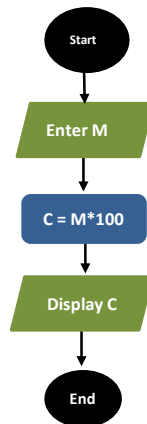
- **Data input:** The action of entering data into a variable is generally expressed by the word "read". For example, the instruction "read M" or "enter M" requests the input of a value by means of an input device (such as a keyboard), to be stored in the variable M.
- **Processing or body of the algorithm**
- **Result output:** It consists of displaying the value of a variable on an output device such as a screen. In general, the action of displaying a variable value is expressed in the pseudocode as "show M" or "display M."

Assignment

It consists of assigning the value of an expression to a variable.


Variable = Expression

They must have the same data type.



START

Enter quantity M of meters.

Calculation: $C = M * 100$

Display: quantity C of centimeters

END

The expression assigned can be a variable, a literal, or a combination of variables, literals, and operators.

It must be borne in mind that the variable and the result of the expression assigned must have the same type of data. This means that for example, a declared variable of date type cannot be assigned a numeric value or character.

Simple conditional

It allows evaluating whether a condition is met or not, and decide what action to take.

If (condition)

What is done if the condition is met

Else

What is done if the condition is not met

Endif**Example:**

```
If   Age >= 18 and Card = "Valid"
```

```
    Msg("Can travel")
```

```
Else
```

```
    Msg("Cannot travel")
```

```
Endif
```

Checking of cases

The first condition that is fulfilled is executed, and no other.

Do case

Case (Condition 1)

Case (Condition 2)

Otherwise

What is done if
no condition is met.

EndCase**Example:****Do case**

Case Score \geq 70 "Exonerated"

Case Score \geq 60 "Regular"

Case Score \geq 50 "Non-regular"

Otherwise

"The student must repeat the course."

EndCase

Iteration

It allows you to move from one value to another.

```
For (initial value) to (end value) [step 1]
```

```
-----
```

```
EndFor
```

Example:

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

```
  Display i
```

```
EndFor
```

Loop

A loop is entered when the action is performed as long as a condition is met.

Do while (condition)

EndDo

Example:

`i = 10`

Do while $i \geq 2$

`Display i`

`i = i-2`

EndDo

GeneXus™

Videos

training.genexus.com

Documentation

wiki.genexus.com

Certifications

training.genexus.com/certifications