## PROCESO COMPUTACIONAL

El computador es una máquina cuya función básica es llevar a cabo operaciones de cómputo (cálculo) sobre elementos de datos. Como todas las máquinas, el computador recibe una materia prima (datos de entrada), la transforma a través de un proceso y devuelve un producto (datos de salida). Sin embargo, el computador posee una característica primordial: el proceso que define la transformación efectuada sobre los datos de entrada puede ser especificado por el usuario; es por ello que se dice que el computador es programable.


## ALGORITMO

Un algoritmo se define como los pasos ordenados que se deben efectuar para realizar un trabajo o tarea específica. Por ejemplo, la realización de una receta de cocina requiere de unos pasos ordenados, así como el armado de un mueble modular.

**Ejemplo:** Diseñe un algoritmo para preparar un litro de limonada

*INICIO*
Llenar una jarra con un litro agua
Exprimir el jugo de 5 limones
Añadir el jugo de los limones al agua
Agregar 5 cucharadas de azúcar
Revolver el agua hasta que el azúcar se disuelva completamente
*FIN*

**Ejemplo:** Diseñe un algoritmo que permita hallar la suma y el promedio de tres números

*INICIO*
*Leer* numero1, numero2, numero3
*Hacer* suma = numero1 + numero2 + numero3
*Hacer* promedio = suma / 3
*Imprimir* suma, promedio
*FIN*

El científico de computación Donald Knuth ofreció una lista de cinco propiedades, que son ampliamente aceptadas como requisitos para un algoritmo:

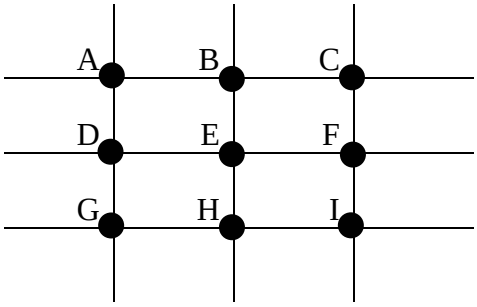
- Carácter finito: Un algoritmo siempre debe terminar después de un número finito de

pasos.

- Precisión: Cada paso de un algoritmo debe estar precisamente definido; las operaciones a llevar a cabo deben ser especificadas de manera rigurosa y no ambigua.
- Entrada: Un algoritmo tiene cero o más entradas: cantidades que le son dadas antes de que el algoritmo comience, o dinámicamente mientras el algoritmo corre. Estas entradas son tomadas de conjuntos específicos de objetos (tienen un universo definido).
- Salida: Un algoritmo tiene una o más salidas.
- Eficacia: Se espera que un algoritmo sea eficaz, en el sentido de que todas las operaciones a realizar en un algoritmo deben ser suficientemente básicas como para que en principio puedan ser hechas de manera exacta y en un tiempo finito por un hombre usando lápiz y papel.

Importante: Un algoritmo define *una manera* de resolver un problema, pero no necesariamente la única. Esto significa que diferentes personas pueden idearse diferentes maneras para resolver un mismo problema. En este caso puede ser que todas las soluciones sean válidas pero que algunas sean más eficientes que otras, entendiendo por eficiencia la cantidad de recursos (tiempo, espacio de memoria, etc.) necesarios para llevarlos a cabo.

**Ejemplo:** Diseñe un algoritmo para desplazarse desde el punto D al B, moviéndose de a un punto por la cuadrícula.



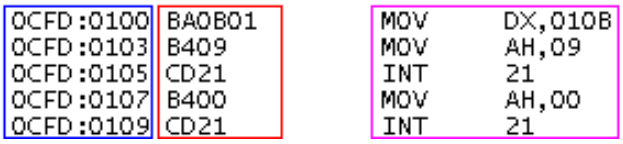

| Solución 1: | Solución 2: | Solución 3: |
|---|---|---|
| *INICIO*<br>Moverse al punto E<br>Moverse al punto B<br>*FIN* | *INICIO*<br>Moverse al punto A<br>Moverse al punto B<br>*FIN* | *INICIO*<br>Moverse al punto G<br>Moverse al punto H<br>Moverse al punto I<br>Moverse al punto F<br>Moverse al punto C<br>Moverse al punto B<br>*FIN* |

## LENGUAJES DE PROGRAMACIÓN

Se usan para poder dar a un computador las diferentes órdenes que llegan a componer un algoritmo. Según su nivel de abstracción los lenguajes de programación se pueden clasificar en:

Lenguajes de bajo nivel

Los lenguajes de bajo nivel son lenguajes de programación que se acercan al funcionamiento de una computadora. El lenguaje de más bajo nivel es, por excelencia, el código máquina. A éste le sigue el lenguaje ensamblador, ya que al programar en ensamblador se trabajan con los registros de memoria de la computadora de forma directa.



```
0CFD:0100 BA0B01
0CFD:0103 B409
0CFD:0105 CD21
0CFD:0107 B400
0CFD:0109 CD21
```

```
MOV     DX,010B
MOV     AH,09
INT     21
MOV     AH,00
INT     21
```

Lenguajes de alto nivel

Los lenguajes de alto nivel son normalmente fáciles de aprender porque están formados por elementos de lenguajes naturales, como el inglés. En BASIC, el lenguaje de alto nivel más conocido, los comandos como "IF CONTADOR = 10 THEN STOP" pueden utilizarse para pedir a la computadora que pare si CONTADOR es igual a 10. Algunos ejemplos de este tipo de lenguaje son Fortran, C++, Visual Basic, Java, etc.

Lenguajes de medio nivel

Hay lenguajes de programación que son considerados por algunos expertos como lenguajes de medio nivel al tener ciertas características que los acercan a los lenguajes de bajo nivel pero teniendo, al mismo tiempo, ciertas cualidades que lo hacen un lenguaje más cercano al humano y, por tanto, de alto nivel.

Un listado de algunos de los lenguajes de programación más conocidos se presenta en la siguiente tabla.

| | | | | |
|---|---|---|---|---|
| ABAP | CORAL | J | Object REXX | Scriptol |
| ABC | D | Java | Objective-C | Seed7 |
| ActionScript | Delphi | JavaScript | Ocaml | Self |
| Ada | DIV | Joy | Occam | Sh |
| Afnix | Dylan | KWC | Oz | Simula |
| ALGOL | Eiffel | Ladder | Pascal | Smalltalk |
| APL | Erlang | Letra | Parlog | Snobol |
| ASP | Ensamblador | Lexico | Perl | SPARK |
| ASP.NET | Extended ML | Lingo | PHP | Squeak |
| AWK | Euphoria | Lisp | PL/1 | SR |
| B | Fénix | Logo | Plankalkül | Standard ML |
| BASIC | Flow-Matic | Lua | PostScript | TI-Basic |
| BCPL | Forth | MAGIC | PowerBuilder | TCL |
| Befunge | FORTRAN | Mainsail | Prolog | VBA |
| Boo | FP | Mesa | Python | Visual Basic |
| C | Gambas | Miranda | Rapid | Visual C++ |
| C++ | GML | ML | REXX | Visual DialogScript |
| C# | GRAFCET | Modula | RPN | Visual Foxpro |
| Caml | Haskell | Modula-2 | RPG | XBase++ |

| | | | | |
|---|---|---|---|---|
| Clipper | Icon | Modula-3 | Ruby | Yurix |
| CLIPS | Inform | Natural | Sail | ZPL |
| CLU | INTERCAL | NetREXX | Sather | |
| COBOL | ISWIM | Oberon | Scheme | |

Para conocer la cronología de los lenguajes de programación, así como sus principales características puede consultarse la página http://www.levenez.com/lang/

## PROGRAMAS DE COMPUTADOR

Los términos "algoritmo" y "programa" tienden a confundirse y en algunos casos a utilizarse indiferentemente. Recordemos que el término "algoritmo" se refiere a la secuencia de pasos para resolver un problema, pero independiente del lenguaje de programación que se utilice, mientras que "programa" se refiere propiamente a la codificación de un algoritmo en algún lenguaje de programación.

Algunos ejemplos de programas son aquellos creados para problemas específicos como la elaboración de facturas, contabilidades, nominas, cálculos de ingeniería, etc. Así como otros de uso más general como Word, Excel, etc.

## OBJETIVO DEL CURSO

La principal razón para que las personas aprendan lenguajes de programación es utilizar el computador como una herramienta para la solución de problemas. Este proceso implica dos pasos:

- Fase de resolución del problema: diseño del algoritmo de manera formal (pseudocódigo, diagrama de caja, diagrama de flujo, etc.)
- Fase de implementación: codificación del algoritmo en algún lenguaje de programación (C++, Visual Basic, etc.)

La fase de resolución del problema incluye tres pasos:

- Análisis del problema: Es necesario examinar cuidadosamente el problema con el fin de tener una idea precisa de lo que se solicita. En este paso de define "que hará" el algoritmo, determinando claramente cuales son las entradas, cuales las salidas y que es lo que se desea que el algoritmo haga.
- Diseño del algoritmo: Consiste en identificar las tareas más importantes para resolver el problema y disponerlas en el orden en el que han de ser desarrolladas. En este paso se debe definir de manera explicita como y cuando se obtienen los datos de entrada, cuales son las operaciones o cálculos necesarios para resolver el problema, y cuales son las respuestas que debe entregar el algoritmo.
- Verificación del algoritmo: Este paso sirve para determinar si el algoritmo es correcto, es decir, si el resultado que produce ante una entrada dada es el esperado. El modo más común de verificación es mediante una prueba manual (conocida

como prueba de escritorio) que consiste en darle diferentes datos de entrada al algoritmo, seguir la secuencia de pasos y determinar si la salida resultante es la que se espera.

La fase de implementación consiste en la codificación (traducción) el algoritmo en un lenguaje de programación determinado (en este caso C++). Esta fase se complementa con la ejecución del programa y su verificación.

**Ejemplo:** Suponga que se necesita un algoritmo para realizar la factura en un almacen que vende televisores (suponiendo que de un solo tipo). La factura debe mostrar el valor total a pagar el valor de los impuestos. El costo por unidad de los televisores es de $400.000 y el impuesto aplicado es del 16%.

**Análisis del problema:**

Datos de entrada:
Numero de unidades compradas

Datos de salida:
Valor total a pagar
Valor de los impuestos

**Diseño del algoritmo:**

*INICIO*
1. *Leer* numero de unidades
2. *Hacer* Impuestos = numero de unidades * $400.000 * 0.16
3. *Hacer* Total = numero de unidades * 400.000 + Impuestos
4. *Imprimir* Total, Impuestos

*FIN*

**Verificación del algoritmo:**

*INICIO*
1. Si numero de unidades = 8
2. Impuestos = 8 * $400.000 * 0.16 = $512.000
3. Total = 8 * $400.000 + $512.000 = $3’712.000
4. $3’712.000, $512.000

*FIN*