

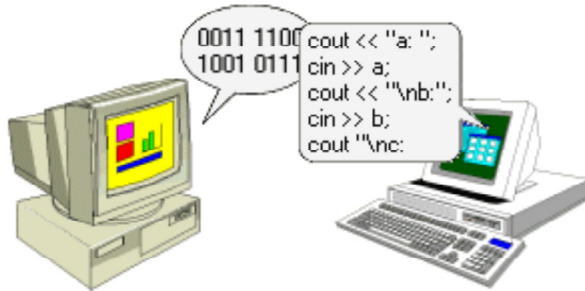
Fundamentos



9- Programación: Lenguajes



Muchos **lenguajes de computación** están disponibles para escribir programas de computadoras. Cada uno de ellos tienen ventajas para ciertas clases de tareas.



Veamos algunos ejemplos de los varios tipos de lenguajes para computadoras y analicemos para que se usan.

Dónde están:

[Jegsworks](#) > [Lecciones](#) > [Fundamentos](#)

1. Tipos de Computadoras ▶
2. Aplicaciones ▶
3. Input ▶
4. Procesamiento ▶
5. Output ▶
6. Almacenamiento ▶
7. Computadora a Computadora ▶
8. Sistemas operativos ▶
9. Programación ↴
 - Intro
 - Lenguajes
 - Creando
 - Examen

10. Lo que usted ve ▶
11. Manos a la obra ▶
12. Por las suyas ▶

[Buscar](#)
[Glosario](#)
[Apéndice](#)

Lenguaje de Máquina

Es la lengua nativa de la CPU.

Cada diseño de CPU tiene su propio **lenguaje de máquina**. Este es el juego de instrucciones que usa el chip para sí mismo. Está hecho de ceros y unos (números binarios) y resulta muy difícil para el trabajo de la gente.

10	23
11	FF
12	12
13	10
14	50
15	23
16	30
17	40
18	C0
19	00

El **lenguaje de máquina** parece compuesto de números solamente. En el segmento de un programa que vemos a la izquierda, la primera columna le dice a la computadora dónde llenar su memoria y en la segunda columna sobre una base **hexadecimal** (base 16) figuran los valores que debe poner en dichas locaciones de las memorias.

Para más información sobre números hexadecimal, vea [Aritmética de Base](#).

Otro ejemplo de **lenguaje de máquina** es el siguiente:

El segmento de código en lenguaje **Java** es:

```
int counter = 0;
```

```
counter = counter + 1;
```

podría ser trasladado en lenguaje de máquina como

```
000101000100010001000100001000101010111110
000001110101000111110000100010000010101010
```

Lenguaje Assembler (Assembly)

Son **códigos** o **abreviaturas** para las instrucciones del lenguaje de máquina.

En un **lenguaje Assembler** se le asigna un código a cada instrucción en lenguaje de máquina. De esta manera, en lugar de tener que recordar una cadena de ceros y unos, el programador sólo tendrá que recordar códigos cortos como: ADD, MOV, o JLE.

Ciertamente esto es una mejora comparado con recordar

000101000100010001000100001000101010111110!! pero tampoco resulta muy sencillo.

El **programa en Assembler** a continuación sirve para leer dos caracteres y mostrarlos sobre la pantalla que se indica. Note que el texto que figura a la derecha de los punto y comas, (;) es ignorado por la computadora. Figura esto para explicar el programa a cualquiera que mire el código. Note que cada pequeño paso debe ser codificado y que todo esto solo sirve para mostrar dos caracteres!

```
;name of the program:one.asm
;
.model small
.stack
.code
    mov AH,1h      ;Selects the 1 D.O.S. function
    int 21h        ;reads character and return ASCII
                    ; code to register AL
    mov DL,AL      ;moves the ASCII code to register DL
    sub DL,30h     ;makes the operation minus 30h to
                    ; convert 0-9 digit number
    cmp DL,9h      ;compares if digit number it was
                    ; between 0-9
    jle digit1     ;If it true gets the first number
                    ; digit (4 bits long)
    sub DL,7h      ;If it false, makes operation minus
                    ; 7h to convert letter A-F digit1:
    mov CL,4h      ;prepares to multiply by 16
    shl DL,CL      ;multiply to convert into four bits upper
    int 21h        ;gets the next character
    sub AL,30h     ;repeats the conversion operation
    cmp AL,9h      ;compares the value 9h with the content
                    ; of register AL
    jle digit2     ;If true, gets the second digit number
    sub AL,7h      ;If no, makes the minus operation 7h
                    ; digit2:
    add DL,AL      ;adds the second number digit
    mov AH,4Ch     ;21h interruption
    int 21h        ;finish the program code
    End
```

FORTRAN

= Formula Translation

El lenguaje FORTRAN fué creado cerca de 1957 para ayudar a los científicos, ingenieros y matemáticos a escribir sus programas y resolver con ellos las ecuaciones matemáticas. Todavía es un lenguaje vastamente utilizado para ese tipo de programas. Fue el primer lenguaje exitoso de **alto nivel**. Han aparecido nuevas versiones y el standard actual es el Fortran 90.

A continuación veremos un programa de ejemplo en el que se acepta el bus o línea de transmisión de datos, número 99 y muestra en pantalla el comando ♦TAKE BUS 99♦ (tome la Línea 99)

```
PROGRAM IDEXMP
  INTEGER BUS_NUM
  BUS_NUM = 99
  WRITE (*,*) ' TAKE BUS ', BUS_NUM
  END
```

COBOL

= Common Business Oriented Language

El COBOL fue escrito alrededor de 1960, teniendo presente las aplicaciones comerciales. Tiene una estructura muy inglesa, usando sentencias y párrafos, a pesar que aquí son muy diferentes de los utilizados para una novela. Esto ayuda a los hombres de negocios, que no son muy hábiles como programadores para que puedan escribir o editar un programa. Pero tiene la desventaja de su tendencia hacia programas muy largos y con muchas palabras. Es un buen lenguaje para programas simples y directos.

COBOL fue usado para crear muchos programas para las computadoras de gran porte de importantes empresas. Estos programas fueron actualizados durante las medidas llamadas Y2K, que se tomaron al llegar el año 2000. De manera que parece muy probable que todavía habrá programas en COBOL cerca nuestro por largo tiempo. Puede ver el artículo en <http://www.infogoal.com/cbd/cbdzo09.htm> para un desarrollo más extenso sobre ese tema.

El ejemplo que sigue acepta dos números a los que multiplica y los muestra junto con el resultado de la operación. Vea el punto de PROCEDURE DIVISION para notar donde se hizo el cálculo.

```
$ SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. FragmentA.
AUTHOR. Michael Coughlan.

DATA DIVISION.

WORKING-STORAGE SECTION.
01 Num1
01 Num2
01 Result
PIC 9 VALUE ZEROS.
PIC 9 VALUE ZEROS.
PIC 99 VALUE ZEROS.
```

```

PROCEDURE DIVISION.
Calc-Result.
    ACCEPT Num1.
    MULTIPLY Num1 BY Num2 GIVING Result.
    ACCEPT Num2.
    DISPLAY "Result is = ", Result.
    STOP RUN.

```

BASIC

= Beginner's All Symbolic Instruction Code

Este lenguaje fue escrito en 1964, (que para los programadores de hoy es la edad de piedra!) para que los estudiantes de la universidad aprendieran los conceptos de la programación. Originalmente el BASIC era para ser usado solamente en las clases. Pero el lenguaje ha probado ser muy útil en el mundo real. Una gran variedad de "dialectos" del BASIC fueron desarrollándose a través de los años. Ahora es muy popular el **Visual Basic** para programar aplicaciones bajo Windows. **Microsoft Visual Basic for Applications** es ejemplo de un desprendimiento del BASIC modificado para ayudar a los usuarios a escribir pequeños subprogramas llamados "**scripts**" (libretos) o "**macros**" para ser usados con las aplicaciones. Otras aplicaciones pueden también tener sus propias variaciones sobre el BASIC, por ejemplo Word Basic y Excel Basic.

Los creadores de este lenguaje deseaban un lenguaje que se pareciera más al idioma inglés normal. Aunque no se parece mucho al mismo, utiliza suficiente la sintaxis de ese idioma para darle un parecido más natural que muchos otros lenguajes de computación.

El corto programa de más abajo está escrito en BASIC. Acepta las medidas de distancia en millas, yardas, pies y pulgadas y las convierte en kilómetros, metros y centímetros. Note como el programador puede escribir ecuaciones para hacer los cálculos.

```

' English to Metric Conversion
' J. S. Quasney
' *****
PRINT : PRINT "English to Metric Conversion"
PRINT
INPUT "Miles: ", Miles
INPUT "Yards: ", Yards
INPUT "Feet: ", Feet
INPUT "Inches: ", Inches
Inches = 63360 * Miles + 36 * Yards + 12 * Feet + Inches
Meters# = Inches / 39.37#
Kilometers = INT(Meters# / 1000)
Meters# = Meters# - 1000 * Kilometers
Final.Meters = INT(Meters#)
Centimeters = Meters# - Final.Meters
Centimeters = 100 * Centimeters
Centimeters = INT((Centimeters + .005) * 100) / 100
PRINT
PRINT "Kilometers: "; Kilometers
PRINT "Meters: "; Final.Meters
PRINT "Centimeters: "; Centimeters
END

```

C



Originalmente creado para escribir software para sistemas, el lenguaje **C** ha evolucionado al **C++**. Los dos son ampliamente utilizados por programadores profesionales para toda clase de programas.

El programa siguiente está escrito en **C++**, acepta tres números y verifica si el tercero es igual a la diferencia entre los dos primeros


```
#include <iostream.h>
void main()
{
    int a, b, c;
    cout << "Please enter three numbers\n";
    cout << "a: ";
    cin >> a;
    cout << "\nb: ";
    cin >> b;
    cout << "\nc: ";
    cin >> c;

    if (c==(a-b))
    {
        cout << "a: ";
        cout << a;
        cout << " minus b: ";
        cout << b;
        cout << " equals c: ";
        cout << c << endl;
    }
    else
        cout << "a-b does not equal c:" << endl;
}
```

Java

Java es un nuevo lenguaje popular, utilizado para escribir tanto aplicaciones completas como pequeños  applets  para las páginas de Internet (web). Su objetivo es el de crear aplicaciones que corran en cualquier computadora a diferencia de otros lenguajes que no son compatibles con todas las plataformas.

Por ejemplo, MS Word para Windows no funcionará en una Apple Macintosh o una computadora UNIX. Sus sistemas operativos no son compatibles con el programa. Usted tendrá que obtener la versión de MS Word escrita para su sistema operativo particular.

 **Aclaración:** no confundir **Java** con **JavaScript** que es un lenguaje para scripts (libretos) - comúnmente usado en las páginas web. (La única cosa que comparten son sus nombres parecidos!)

El siguiente ejemplo dibuja un cuadrado en una página HTML (Hyper Text Markup Language) y cuenta el número de veces que usted ha clickeado en el box.

```
import java.applet.*;
import java.awt.*;
public class exfour extends Applet
{
    int i;
    public void init()
    {
        resize(300,300);
    }
    public void paint(Graphics g)
    {
        g.drawString("You clicked the mouse "+i+" Times",50,50);
    }
    public boolean mouseUp(Event e, int x, int y)
    {

```

```
i++;  
repaint();  
return true;  
}
```

LECCIONES FUNDAMENTOS ANTERIOR PRÓXIMA

Maestros: [Solicita permiso](#) para usar este sitio con su clase.

 [Descargar lecciones](#)
[¿Desea ayudar?](#)

Copyright © 1997-2009 [Jan Smith](#) <jegs1@jegsworks.com>
All Rights Reserved.

Traducción: Enrique Chornogubsky

~~ 1 Cor. 10:31 ...hagan cualquier otra cosa, háganlo todo para la gloria de Dios. ~~

Actualizado por última vez el día: 22 Jan 2008