

UD 1: Introducción

Lenguajes de programación

Notación o conjunto de símbolos y caracteres combinados entre sí de acuerdo con una sintaxis ya definida para posibilitar la transmisión de instrucciones (instrucciones máquina) a la CPU.

Lenguajes de bajo nivel (más próximos a la arquitectura de la máquina)

Lenguaje máquina

- Primer lenguaje de programación
- Único lenguaje que entiende directamente el ordenador
- Se basa en la combinación de 0's y 1's
- Propio de cada procesador

```

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
55 89 E5 83 EC 18 89 5D F8 8B 55 08 31 DB 89 75 ; U%ãfi.%]ø«U.1Û%u
FC 8B 02 31 F6 8B 00 3D 91 00 00 C0 77 43 3D 8D ; ù<.1ò<.=\'...ÀwC=]
00 00 C0 72 5B BE 01 00 00 00 C7 04 24 08 00 00 ; ..Àr[%....Ç.$....
00 31 C0 89 44 24 04 E8 84 07 00 00 83 F8 01 74 ; .1À%D$.è'....fø.t
6C 85 C0 74 2A C7 04 24 08 00 00 00 FF D0 BB FF ; l...Àt*Ç.$....ýÐ»ý
FF FF FF 89 D8 8B 75 FC 8B 5D F8 89 EC 5D C2 04 ; ýýý%Ø<uü<]ø%]Á.
00 3D 93 00 00 C0 74 BD 3D 94 00 00 C0 74 BB 89 ; .="...Àt%="...Àt»%
D8 8B 75 FC 8B 5D F8 89 EC 5D C2 04 00 8D 76 00 ; Ø<uü<]ø%]Á..l v.
3D 05 00 00 C0 75 E8 C7 04 24 0B 00 00 00 31 F6 ; =...ÀuèÇ.$....1ò
89 74 24 04 E8 27 07 00 00 83 F8 01 74 34 85 C0 ; %t$.è'....fø.t4...À
74 CD C7 04 24 0B 00 00 00 FF D0 EB A1 C7 04 24 ; tíÇ.$....ýÐè;Ç.$.
08 00 00 00 BB 01 00 00 00 89 5C 24 04 E8 FE 06 ; ....»....%)\$.èp.
00 00 85 F6 74 88 E8 35 02 00 00 BB FF FF FF FF ; .....öt^è5....»ýýýý
EB 81 C7 04 24 0B 00 00 00 B9 01 00 00 BB FF ; è[] Ç.$....'....»ý
FF FF FF 89 4C 24 04 E8 D4 06 00 00 E9 62 FF FF ; ýýý%L$.èÓ...ébyý
FF EB OD 90 90 90 90 90 90 90 90 90 90 90 90 ; ýè.[] [] [] [] [] [] [] [] [] []
55 89 E5 53 83 EC 24 8D 5D F8 C7 04 24 00 10 40 ; U%ãSfi$[]]øÇ.$...Ø
00 E8 4A 07 00 00 83 EC 04 E8 E2 01 00 00 C7 45 ; .èJ...fi.èá...ÇE
F8 00 00 00 00 B8 00 40 40 00 8D 55 F4 89 5C 24 ; ø....].ØØ.[] Uò%\$.
10 8B OD 00 20 40 00 89 44 24 04 89 54 24 08 89 ; <... Ø.%D$.%T$.%
4C 24 0C C7 04 24 04 40 40 00 E8 A1 06 00 00 A1 ; L$.Ç.$.ØØ.è;...;
10 40 40 00 85 C0 74 58 A3 10 20 40 00 8B 15 D4 ; .ØØ....ÀtX£. Ø.<.Ó
50 40 00 85 D2 0F 85 8B 00 00 00 83 FA E0 74 20 ; PØ....Ó....<...fúàt
A1 10 40 40 00 89 44 24 04 8B 1D D4 50 40 00 8B ; ;.ØØ.%D$.<..ÓPØ.<
4B 30 89 0C 24 E8 56 06 00 00 8B 15 D4 50 40 00 ; KO%. $èV...<..ÓPØ.
83 FA C0 74 1B 8B 1D 10 40 40 00 89 5C 24 04 8B ; fúàt.<...ØØ.%)\$.<
OD D4 50 40 00 8B 51 50 89 14 24 E8 30 06 00 00 ; .ÓPØ.<QP%. $èO...
E8 1B 06 00 00 8B 1D 10 20 40 00 89 18 E8 0E 01 ; è....<... Ø.%è...
00 00 83 E4 F0 E8 E6 05 00 00 8B 08 89 4C 24 08 ; ..fäðèæ...<.%L$.
8B 15 00 40 40 00 89 54 24 04 A1 04 40 40 00 89 ; <...ØØ.%T$.; .ØØ.%
04 24 E8 A9 00 00 00 89 C3 E8 B2 05 00 00 89 1C ; . $è@...%Ãè*...%.
24 E8 7A 06 00 00 89 44 24 04 8B 15 D4 50 40 00 ; $èz...%D$.<..ÓPØ.
8B 42 10 89 04 24 E8 D5 05 00 00 8B 15 D4 50 40 ; <B.%.$èÓ....<..ÓPØ
    
```

Lenguaje ensamblador

- Sustituto al Lenguaje máquina
- Lenguaje simbólico o nemotécnico
- Cada instrucción se transforma en una única instrucción máquina.
- Existencia de pseudoinstrucciones

```
.model small
.stack
.data
Cadenal DB 'Hola Manola.$'
.code

programa:
    mov ax, @data
    mov ds, ax
    mov dx, offset Cadenal
    mov ah, 9
    int 21h
end programa
```

http://es.wikipedia.org/wiki/Hola_mundo

Lenguajes de alto nivel (más próximos al usuario/programador)

COBOL

- Lenguaje común orientado a negocios
- Creado en 1960

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
MAIN SECTION.
DISPLAY "Hola Manola"
STOP RUN.
```

Pascal

- Desarrollado por un profesor suizo a finales de los 60
- Su objetivo era facilitar el aprendizaje de la programación a sus alumnos.
- Lenguaje de programación estructurado fuertemente tipificado.
 - Estructuras bien definidas, procedimientos y funciones.
 - Obligación de declarar las variables
- La asignación con “:=” y la comparación con “=”
- Asignación de variables de distinto tipo no son permitidas
- No sensitivo a mayúsculas y minúsculas

```
Program HolaManola;  
begin  
    Writeln('¡Hola Manola!');  
end.
```

<http://www.freepascal.org/>

Lenguaje C

- Creado en 1969 Ken **Thompson** y Dennis M. **Ritchie** en los Laboratorios Bell como evolución del anterior lenguaje B
- Orientado a la implementación de Sistemas Operativos, concretamente Unix.
- Apreciado por la eficacia del código
- Se dice que es un lenguaje de **medio nivel** y no de alto nivel
- Su núcleo es bastante simple, pero tiene añadidas funcionalidades adicionales en **bibliotecas** (matemáticas, manejo de ficheros, etc.)
- Manejo de memoria de bajo nivel mediante **punteros**.
- Parámetros por valor, y por referencia simulada por punteros.
- Sensitivo a mayúsculas y minúsculas

```
#include <stdio.h>  
  
int main(void)  
{  
    printf("¡Hola Manola!\n");  
    return 0;  
}
```

C++

- Creado a mediados de los 80 por Bjarne Stroustrup como extensión del lenguaje de programación C.
- Soporte para la POO y soporte de plantillas (*templates*)
- Identificación de tipos en tiempo de ejecución
- Sobrecarga de operadores

```
#include <iostream>

int main()
{
    std::cout << ";Hola Manola!" << std::endl;
    return 0;
}
```

<http://es.wikipedia.org/wiki/C%2B%2B>

C#

- Lenguaje de programación orientado a objetos desarrollado por Microsoft y estandarizado, como parte de su plataforma .NET
- Sintaxis básica de C/C++ y modelo de objetos similar a JAVA.

```
using System;

class MainClass
{
    public static void Main()
    {
        Console.WriteLine(";Hola Manola!");
    }
}
```

http://es.wikibooks.org/wiki/C_sharp.NET

Ruby

- Lenguaje de POO creado por el programador japonés Yukihiro "Matz" Matsumoto en 1993.
- Combina sintaxis de Perl y SmallTalk, así como funcionalidades de Python, Lisp, Dylan y CLU.
- Lenguaje de programación **interpretado**
- Distribuido bajo licencia de software libre.

```
puts "Hola Manola"
```

PHP

```
<?php
echo "Hola Manola";
?>
```

http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_PHP

Java

```
public class HolaManola
{
    public static void main(String[] args)
    {
        System.out.println(";Hola Manola!");
    }
}
```

Intérpretes versus Compiladores

Intérpretes

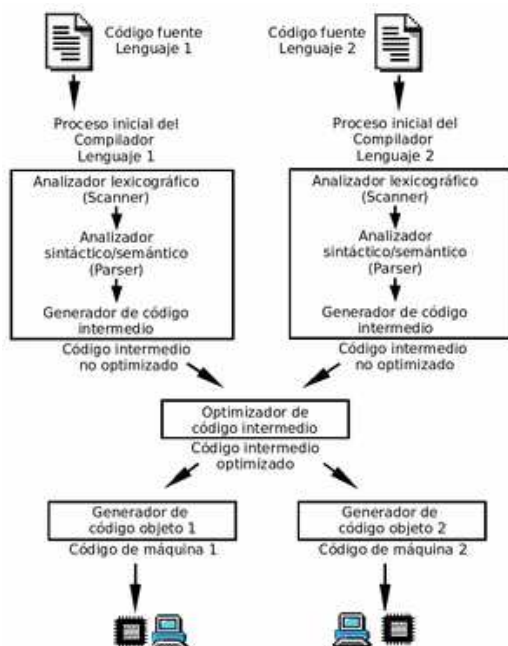
Programa encargado de procesar y traducir cada instrucción o sentencia de in programa escrito en un lenguaje de alto nivel a código máquina y después ejecutarla.

Inconveniente → Lentitud en sucesivas ejecuciones del programa

Compiladores

Traduce el código fuente a su equivalente en código máquina.

Existen diferentes etapas en el proceso de compilación:



- **Edición** del código fuente
- **Compilación** → Obtención del código objeto.
- **Enlace (link)** donde se obtiene finalmente el ejecutable.
- **Ejecución** (opcional).

Estructura de un programa

Algoritmo

Número de pasos **finitos** que debemos hacer para resolver un problema dado.

El diseño de un algoritmo debe reflejar las tres partes de un programa:

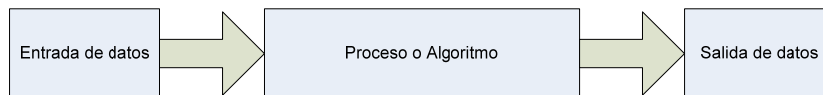


Diagrama de flujos

Se usan en el diseño de algoritmos.

Diagramas de representación gráfica

- Muestran la secuencia lógica de las operaciones o acciones que debe realizar un ordenador
- Muestran la corriente o flujo de datos en la resolución de un problema.
- Debe ser **independiente** del lenguaje de programación
- Debe tener un **diseño normalizado**
- Debe ser **intuitivo**
- Debe ser **flexible** ya que los algoritmos lo son, permitiendo futuras actualizaciones.

Organigramas (Fase de análisis)

Llamados también **Diagramas de Flujo de Sistemas** o **Diagramas de flujo de configuración**.

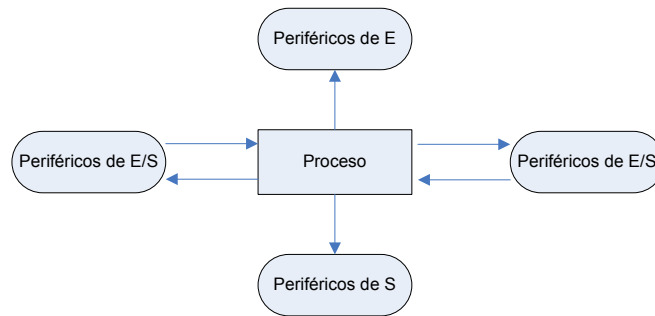
Representaciones gráficas del flujo de datos e información entre los periféricos o soportes físicos que maneja un programa.

Debe reflejar:

- a) Las distintas áreas o programas en los que se divide la solución del problema, y nombre de cada uno de ellos.
- b) E/S de cada área indicando soportes usados para almacenamiento.
- c) El flujo de datos.

Programación en Lenguajes Estructurados

d) Debe seguir esta estructura:



Su simbología es:

a) *Símbolos de soporte de información o dispositivos físicos:*

Símbolo	Denominación	Tipo de dispositivo
	TECLADO	Entrada
	SOPORTE MAGNÉTICO	Entrada
	PANTALLA/CRT	Salida
	IMPRESORA	Salida
	TARJETA PERFORADA	Entrada/Salida
	CINTA DE PAPEL	Entrada/Salida
	DISCO MAGNÉTICO	Entrada/Salida
	DISCO MAGNÉTICO	Entrada/Salida
	CINTA MAGNÉTICA	Entrada/Salida
	CINTA ENCAPSULADA	Entrada/Salida
	DISCO FLEXIBLE	Entrada/Salida
	TAMBOR MAGNÉTICO	Entrada/Salida

b) *Símbolos de proceso:*

Símbolo	Función
	Proceso u operación.
	Clasificación u ordenación de datos en un fichero.
	Fusión o mezcla de dos o más ficheros en uno sólo.
	Partición o extracción de datos de un fichero.
	Manipulación de uno o varios ficheros (<i>intercalación</i>).

c) *Líneas de flujo de datos:*

Símbolo	Función
	Dirección del proceso o flujo de datos.
	Líneas de teleproceso (<i>transmisión de datos</i>).
	Línea conectora. Permite la unión entre unidades o elementos de información.

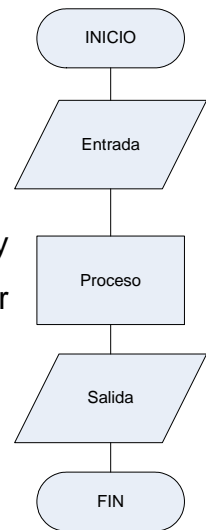
Ordinogramas (Fase de diseño)

Llamados también como **Diagramas de flujo de programas**.

Muestran de manera gráfica la secuencia lógica y detallada de las operaciones que se van a realizar en la ejecución de un programa.

Debe reflejar:

- Un **INICIO**.
- Secuencia de operaciones, lo más detallada posible y siguiendo siempre el orden en el que se deberán ejecutar (arriba-abajo e izquierda-derecha).
- Un **FIN**.
- No se pueden cruzar las líneas de conexión.



Su simbología es:

a) Símbolos de operación o proceso:

Símbolo	Función
	Terminal (marca el inicio, final o una parada necesaria realizada en la ejecución del programa).
	Operación de E/S en general (utilizada para mostrar la introducción de datos desde un periférico a la memoria del ordenador y la salida de resultados desde la memoria del ordenador a un periférico).
	Proceso u operación en general (utilizado para mostrar cualquier tipo de operación durante el proceso de elaboración de los datos depositados en la memoria).
	Subprograma o subrutina (utilizado para realizar una llamada a un subprograma o proceso, es decir, un módulo independiente cuyo objetivo es realizar una tarea y devolver el control de ejecución del programa al módulo principal).

b) Símbolos de decisión:

Símbolo	Función
	Decisión de dos salidas (indica operaciones lógicas o comparativas seleccionando en función del resultado entre dos caminos alternativos que se pueden seguir).
	Decisión múltiple con "n" salidas (indica el camino que se puede seguir entre varias posibilidades según el resultado de la operación lógica o comparación establecida).
	Bucle definido, empleado para modificar una instrucción o bloque de instrucciones que a su vez producen una alteración o modificación en el comportamiento del programa.

c) Líneas de flujo:

Símbolo	Función
	Flechas indicadoras de la dirección del flujo de datos.
	Línea conectora, también llamada línea de flujo de datos (permite la conexión entre los diferentes símbolos utilizados en el diseño).

e) Símbolo de comentarios:

Símbolo	Función
	Permite escribir comentarios a lo largo del diseño realizado.

d) Símbolos de conexión:

Símbolo	Función
	Conector (este símbolo es utilizado para el reagrupamiento de líneas de flujo).
	Conector de líneas de flujo en la misma página (utilizado para enlazar dos partes cualesquiera del diseño a través de un conector de salida y un conector de entrada).
	Conector de líneas de flujo en distintas páginas (utilizado para enlazar dos partes cualesquiera del diseño a través de un conector de salida y un conector de entrada).

Pseudocódigo

Estructura general

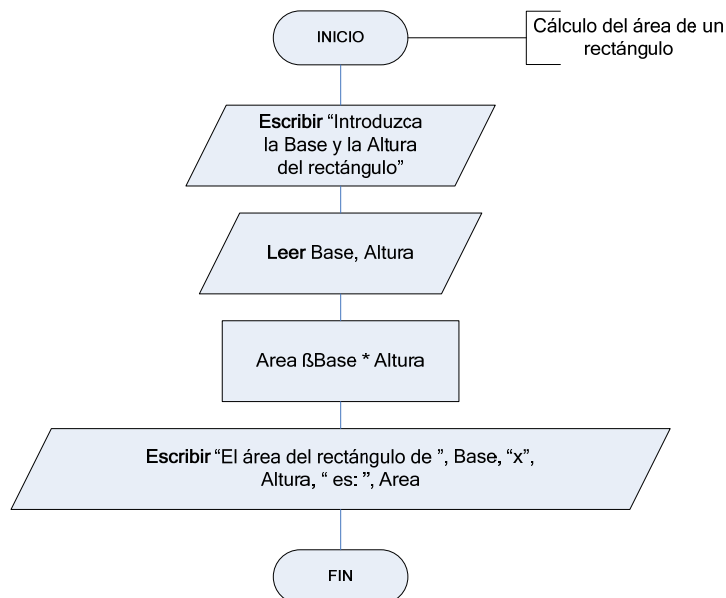


Ejemplo

```

algoritmo Area_del_rectangulo    {Cálculo del área de un rectángulo}
módulo Principal
    variables reales Area, Base, Altura: numérico real

    inicio
        Escribir "Introduzca la Base y la Altura del rectángulo"
        Leer Base, Altura
        Area ← Base * Altura
        Escribir "El área del rectángulo de ", Base, "x", Altura, " es: ", Area
    fin algoritmo
    
```



Ejercicios propuestos

- 1) Escribir un programa que lea el valor en millas marinas y las convierta en metros. Expresando su valor en pantalla. Nota: Una milla marina equivale a 1852 metros.
- 2) Realizar un programa que calcule y escriba el porcentaje descontado en una compra sabiendo el precio original y el pagado finalmente.
- 3) Algoritmo que lee dos valores numéricos y determina si son iguales o, en caso contrario, indica cual es mayor de los dos.
- 4) Programa que lee un número entero positivo y determina el número de dígitos decimales necesarios para la representación de dicho valor. Ejemplo: El valor 358 necesita 3 dígitos, el 9863 necesita 4 dígitos, etc.
- 5) Algoritmo que lee 5 valores numéricos y calcula su producto.
- 6) Diseño y algoritmo que calcule el resultado de una potencia dada su base y su exponente.

Bibliografía

- **Programación en Lenguajes Estructurados** (Desarrollo de Aplicaciones Informáticas). *Enrique Quero Catalinas*. Editorial Paraninfo. 2001.
- **Enciclopedia Libre Wikipedia** – <http://es.wikipedia.org>