Lectura motivadora

Algoritmos y programas

En ocasiones se ha comparado la escritura de programas de ordenador con la literatura. Durante los primeros años de las ciencias de la computación, todo el proceso era artesanal y dependía más de destellos de inspiración y de factores como la experiencia o la brillantez de un programador, que de un método fiable y contrastado.

Sin embargo, construir programas por el método de prueba y error es comparable con tratar de levantar un edificio sin tener conocimientos de arquitectura. A lo largo del siglo XX se han llevado a cabo un gran número de investigaciones para tratar de resolver todo tipo de problemas de computación. Fruto de estas investigaciones surgieron diversos lenguajes de programación, estructuras abstractas de datos y bases matemáticas que permiten desarrollar los algoritmos que resolverán los problemas planteados.

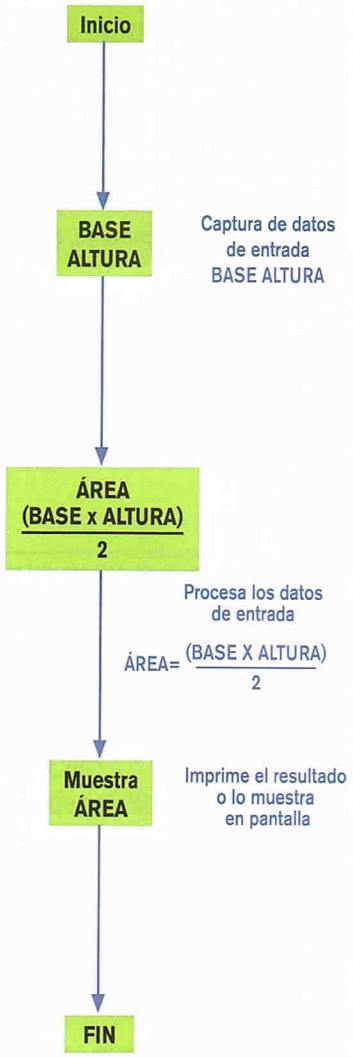
De estos tres elementos imprescindibles a la hora de desarrollar programas, los lenguajes de programación, las estructuras de datos y los algoritmos, el más importante es este último. Al fin y al cabo, una estructura de datos no es más que una forma de organizar la información; por otro lado los lenguajes son un conjunto de órdenes y operadores que nos permiten comunicar al ordenador lo que debe hacer; mientras que los algoritmos son los verdaderos «encargados» de resolver los problemas.

Un algoritmo es una secuencia ordenada de reglas o acciones que, aplicadas sobre un conjunto de datos de entrada, **produce a la salida una solución a un problema concreto.** Sus propiedades más relevantes son:

- Independiente del computador y del lenguaje de programación en que se exprese.
- Definido. Si se aplica varias veces sobre el mismo conjunto de datos de entrada debe producir siempre la misma solución a la salida.
- Preciso. La definición no debe dar lugar a ambigüedades.
- Finito. El algoritmo debe terminar en algún momento.

Así, aunque hablemos de los algoritmos como algo reciente y totalmente relacionado con los ordenadores, lo cierto es que nacieron mucho antes que la informática.

Gráfico 1.
Ejemplo de algoritmo para el cálculo del área de un triángulo.



Preguntas sobre el texto

1. ¿Sabrías decirnos la diferencia entre un algoritmo y un programa de ordenador?
2. ¿Conoces algún algoritmo matemático?
3. ¿Qué otras propiedades podría tener un algoritmo?
4. ¿Qué consideras que es una estructura de datos?

Ideas previas

1. ¿Crees que configurar la grabación de programas en un grabador de TDT es programar un sistema?
2. ¿Has oído hablar de algún lenguaje de programación?
3. ¿Sabrías definir qué es un algoritmo?
4. ¿Conoces algún juguete que se pueda programar?

1 Historia de los lenguajes de programación

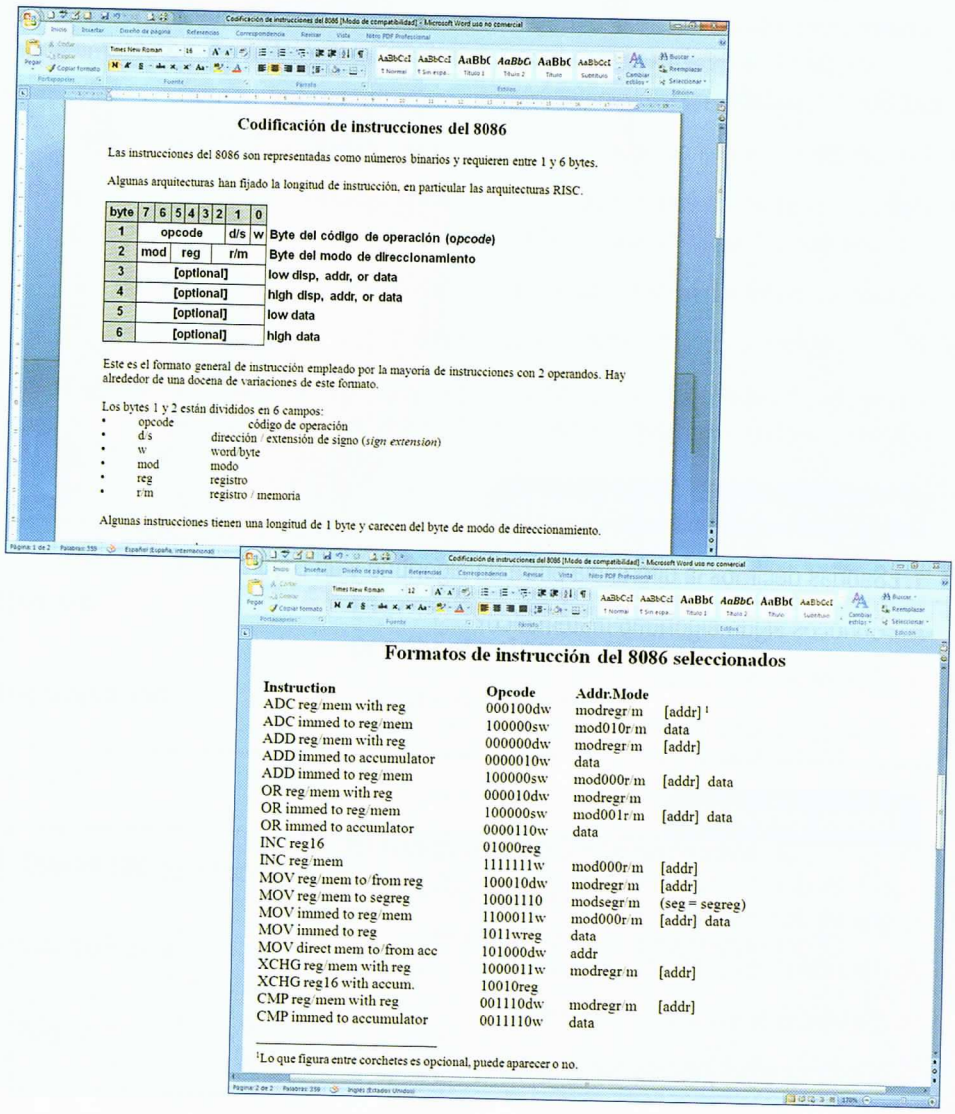
VOCABULARIO

Programa: conjunto de instrucciones que indica a un computador cómo realizar funciones diversas, que van desde la resolución de problemas matemáticos y físicos, al tratamiento de imágenes, textos, bases de datos, sistemas de comunicaciones, etc.

Las computadoras que se construyen desde el siglo XX están basadas en tecnologías eléctricas y electrónicas. Los circuitos que componen el computador funcionan con tensiones estandarizadas de 0 V o 5 V, que son interpretados como valores binarios, esto es, valores lógicos 0 y 1. Los valores lógicos se asocian a valores cualitativos como verdadero o falso, abierto o cerrado, etc. **Las máquinas solo entienden esos impulsos eléctricos obtenidos a partir de la composición de series de bits, que a la vez constituyen largos números binarios difíciles de recordar para las personas.** Aun utilizando otros sistemas de numeración, aparentemente más sencillos, seguimos enfrentándonos a listas de números a las que resulta complicado dar una estructura y sentido para trabajar directamente con ellas.

Los primeros intentos de convertir el código de la máquina en algo más familiar a los humanos consistieron en asignar letras a los códigos de operación. Así, la instrucción de suma era la letra A, de *add* (suma en inglés). Esto resultó poco operativo, porque el número de instrucciones que eran capaces de ejecutar las máquinas fue creciendo rápidamente. De este modo nacieron los lenguajes ensambladores, que asignaban mnemónicos de tres o cuatro letras a los códigos de operación binarios.

Codificación de las instrucciones del microprocesador 8086.



Codificación de instrucciones del 8086

Las instrucciones del 8086 son representadas como números binarios y requieren entre 1 y 6 bytes.

Algunas arquitecturas han fijado la longitud de instrucción, en particular las arquitecturas RISC.

byte	7	6	5	4	3	2	1	0	
1	opcode						d/s	w	Byte del código de operación (*opcode*)
2	mod		reg		r/m				Byte del modo de direccionamiento
3	[optional]								low disp, addr, or data
4	[optional]								high disp, addr, or data
5	[optional]								low data
6	[optional]								high data

Este es el formato general de instrucción empleado por la mayoría de instrucciones con 2 operandos. Hay alrededor de una docena de variaciones de este formato.

Los bytes 1 y 2 están divididos en 6 campos:

- opcode — código de operación
- d/s — dirección / extensión de signo (*sign extension*)
- w — word/byte
- mod — modo
- reg — registro
- r/m — registro / memoria

Algunas instrucciones tienen una longitud de 1 byte y carecen del byte de modo de direccionamiento.

Formatos de instrucción del 8086 seleccionados

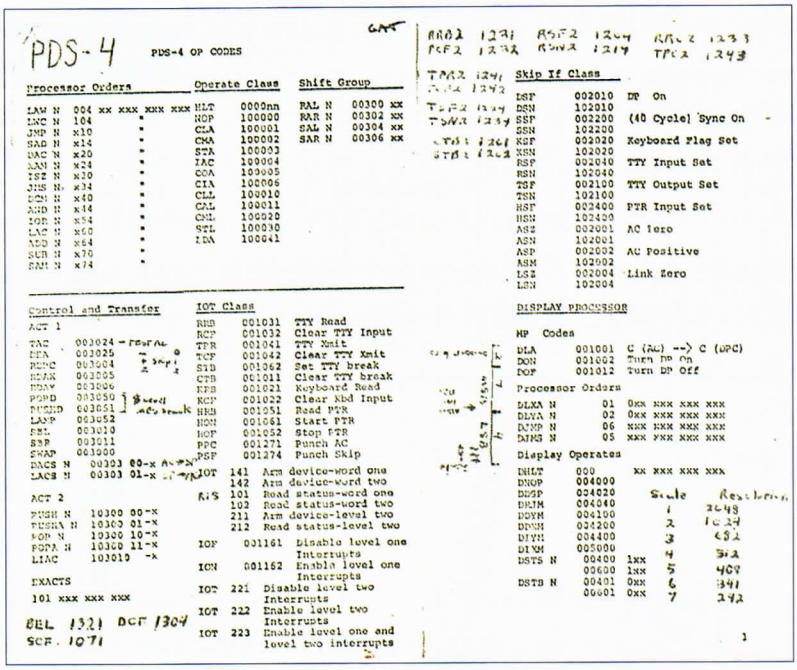
Instruction	Opcode	Addr:Mode	
ADC reg/mem with reg	000100dw	modregr/m	[addr] [1]
ADC immed to reg/mem	100000sw	mod010r/m	data
ADD reg/mem with reg	000000dw	modregr/m	[addr]
ADD immed to accumulator	0000010w	data	
ADD immed to reg/mem	100000sw	mod000r/m	[addr] data
OR reg/mem with reg	000010dw	modregr/m	
OR immed to reg/mem	100000sw	mod001r/m	[addr] data
OR immed to accumulator	0000110w	data	
INC reg16	01000reg		
INC reg/mem	1111111w	mod000r/m	[addr]
MOV reg/mem to/from reg	100010dw	modregr/m	[addr]
MOV reg/mem to segreg	10001110	modsegr/m	(seg = segreg)
MOV immed to reg/mem	1100011w	mod000r/m	[addr] data
MOV immed to reg	1011wreg	data	
MOV direct mem to/from acc	101000dw	addr	
XCHG reg/mem with reg	1000011w	modregr/m	[addr]
XCHG reg16 with accum.	10010reg		
CMP reg/mem with reg	001110dw	modregr/m	[addr]
CMP immed to accumulator	0011110w	data	

[1]Lo que figura entre corchetes es opcional, puede aparecer o no.

TEN EN CUENTA

Código de operación

Cada instrucción que es capaz de ejecutar un microprocesador tiene asignado un código binario concreto. Por ejemplo, en el microprocesador Intel 8086 y compatibles, la instrucción HLT, que hace entrar al procesador en estado de parada, tiene un *opcode* (código de operación) de 11110100.



PDS-4

PDS-4 OP CODES

Processor Orders

LAW N	004 xx xxx xxx xxx	HLT	0000nn
INC N	104	NOP	100000
JMP N	x10	CLA	100001
DAC N	x14	CMA	100002
LAC N	x20	STA	100003
AAM N	x24	IAC	100004
ISZ N	x30	COA	100005
JMS N	x34	CIA	100006
DZM N	x40	CLL	100010
AND N	x44	CAL	100011
IOR N	x54	CML	100020
XOR N	x60	STL	100030
ADD N	x64	IDA	100041
SUB N	x70		
SAM N	x74		

Operate Class

RAL N	00300 xx
RAR N	00302 xx
SAL N	00304 xx
SAR N	00306 xx

Shift Group

Skip If Class

DSF	002010	DP On
DSN	102010	
SSF	002200	(40 Cycle) Sync On
SSN	102200	
KSF	002020	Keyboard Flag Set
KSN	102020	
RSF	002040	TTY Input Set
RSN	102040	
TSF	002100	TTY Output Set
TSN	102100	
HSF	002400	PTR Input Set
HSN	102400	
ASZ	002001	AC Zero
ASN	102001	
ASP	002002	AC Positive
ASM	102002	
LSZ	002004	Link Zero
LSN	102004	

Control and Transfer

ACT 1

TAC	003024	
IDA	003025	
RUNC	003026	
RDAY	003006	
PSPD	003050	
PSPDN	003051	
HLTP	003052	
SBL	003010	
SBR	003011	
SKAP	003000	
SACS N	00303 00-x	
LACS N	00303 01-x	

ACT 2

PUSH N	10300 00-x	
PUSHA N	10300 01-x	
POP N	10300 10-x	
POPA N	10300 11-x	
LIAC	103019 -x	

EXACTS

101 xxx xxx xxx

IOT Class

RRB	001031	TTY Read
RCP	001032	Clear TTY Input
TPR	001041	TTY Xmit
TCF	001042	Clear TTY Xmit
STB	001062	Set TTY break
CTB	001011	Clear TTY break
KRB	001021	Keyboard Read
KCF	001022	Clear Kbd Input
HRB	001051	Read PTR
HSK	001061	Start PTR
HOF	001052	Stop PTR
PPC	001271	Punch AC
PSF	001274	Punch Skip
IOT	141	Arm device-word one
	142	Arm device-word two
RiS	101	Read status-word one
	102	Read status-word two
	211	Arm device-level two
	212	Read status-level two
IOF	001161	Disable level one Interrupts
ION	001162	Enable level one Interrupts
IOT	221	Disable level two Interrupts
IOT	222	Enable level two Interrupts
IOT	223	Enable level one and level two interrupts

DISPLAY PROCESSOR

MP Codes

DLA	001001	C (AC) --> C (DPC)
DON	001002	Turn DP On
DOF	001012	Turn DP Off

Processor Orders

DLXA N	01	0xx xxx xxx xxx
DLYA N	02	0xx xxx xxx xxx
DJMP N	06	xxx xxx xxx xxx
DJMS N	05	xxx xxx xxx xxx

Display Operates

DHLT	000	xx xxx xxx xxx
DNOP	004000	
DDSP	004020	
DRJM	004040	
DPYM	004100	
DPXM	004200	
DIYM	004400	
DIXM	005000	
DSTS N	00400	1xx
	00600	1xx
DSTB N	00401	0xx
	00601	0xx

Scale / Resolution

Scale	Resolution
1	2048
2	1024
3	682
4	512
5	409
6	341
7	292

Tarjeta de referencia de lenguaje ensamblador.

Actividades

1. Busca las instrucciones de un microprocesador sencillo como el 8086 y agrúpalas por funciones: aritméticas y lógicas, de movimiento de datos, etc.
2. Investiga qué son los modos de direccionamiento de un microprocesador y para qué sirven. ¿Permiten todas las instrucciones todos los tipos de direccionamiento? ¿Por qué crees que esto es así?
3. Investiga qué es la ortogonalidad referida a los juegos de instrucciones de microprocesadores. ¿Existen o han existido microprocesadores con juegos de instrucciones ortogonales? Señala las ventajas e inconvenientes de este tipo de máquinas.

No obstante, el lenguaje ensamblador con sus códigos mnemónicos sigue siendo un modo complicado de programar computadores, sujeto a muchos errores por estar demasiado próximo al lenguaje de la máquina. Por esta razón se pensó en crear **métodos de codificación que estuvieran en un nivel más próximo al lenguaje humano**, los llamados **lenguajes de alto nivel.**

En 1953, John Backus, programador al servicio de IBM, puso en marcha un proyecto cuyo fin era crear un sistema de traducción automatizado, que traduciría de la notación matemática normal al lenguaje del procesador de la máquina IBM 704. Como resultado de las investigaciones de Backus y su equipo, en 1957 vio la luz la primera versión de FORTRAN (*the IBM mathematical formula translating system*).

TEN EN CUENTA

Mayúsculas y minúsculas en FORTRAN

La especificación original del lenguaje FORTRAN establece que los programas deben escribirse empleando solo letras mayúsculas. Las actuales implementaciones permiten el uso de minúsculas, lo que mejora la legibilidad.

Ejemplo:

Análisis de un programa

Ejemplo de programa en FORTRAN 90 para sumar dos números:

```
PROGRAM SUMA
REAL :: A, B, C
READ (*,*) A, B
C = A + B
WRITE (*,*) 'La suma es: ', C
END
```

El programa FORTRAN, para sumar dos números, consta de seis líneas:

1. Programa principal designado con la palabra reservada PROGRAM y nombre de programa.
2. Declaración de variables que se van a usar y tipo correspondiente.
3. Solicitud de lectura desde el teclado de dos de las variables, A y B.
4. Asignación a la variable C del valor resultante de sumar A y B.
5. Escritura en pantalla de un mensaje y el contenido de C.
6. Fin de programa.

Tras la aparición de FORTRAN, numerosos lenguajes han surgido en el panorama de la computación. Cada nuevo lenguaje se adapta mejor a un determinado tipo de aplicaciones: científicas, de gestión, inteligencia artificial, etc.

Actividades

4. Un diagrama de flujo es un método gráfico para expresar la lógica de un programa. Busca información sobre cómo crear diagramas de flujo y construye uno que corresponda con el programa SUMA del ejemplo en FORTRAN.

5. Compara el ejemplo resuelto en FORTRAN con otro similar en ensamblador. Anota tus conclusiones. ¿Cuál resulta más comprensible? ¿Qué ventajas aporta FORTRAN sobre ensamblador?

```
;programa en ensamblador que suma dos números leídos desde teclado
.model small
.stack
.data
    var1 db ?
.code
.startup

    mov ah,01h  ;lee carácter desde teclado
    int 21h             ;ejecuta la interrupción y lee primer carácter
    sub al,30h  ;hace el ajuste de carácter a número
    mov var1,al ;guarda el número en var1

    mov ah,01h  ;lee carácter desde teclado
    int 21              ;ejecuta la interrupción y lee segundo carácter
    sub al,30h  ;hace el ajuste de carácter a número
    add al,var1 ;suma los dos valores

    mov dl,al           ;pone el número a imprimir en dl
    add dl,30h  ;hace el ajuste de número a carácter
    mov ah,02h  ;función para imprimir un carácter en pantalla
    int 21h             ;muestra en pantalla

.exit
end
```

Actividades

6. Si estás trabajando con un sistema GNU/Linux, comprueba qué utilidades de programación aparecen en el menú. Investiga y anota si se tratan de lenguajes de programación o de entornos de programación, IDE *(integrated development environment)*. ¿Qué diferencia hay entre ambos conceptos?

7. Busca información sobre los lenguajes más empleados en los últimos años. Construye una línea de tiempo en la que aparezcan los desarrolladores de los lenguajes y el propósito principal para el que fueron creados.

Existen muchos lenguajes de programación: C, C++, Java, PHP, Logo, Python, C#, Ada, Algol, etc...

Un **lenguaje de programación** es un **sistema formal de comunicación entre personas y máquinas.**

Los lenguajes de programación están orientados principalmente a computadoras y robots, pero abarcan otros dispositivos como los autómatas programables o los teléfonos inteligentes.

La función de un lenguaje de programación es expresar, de forma concreta, **el modo de ejecutar los algoritmos que forman parte de un programa.**

Los lenguajes de programación están desarrollados sobre un conjunto de símbolos que constituyen un alfabeto, junto con una estructura de reglas que establece una gramática, donde se define la sintaxis que permite la construcción válida del lenguaje.

No importa qué lenguaje de programación empleemos, **el código que hayamos desarrollado debe traducirse al lenguaje de la máquina.** De esto **se encargan** unos programas diseñados para ello, son **los compiladores e intérpretes,** que trataremos en el siguiente punto.

VOCABULARIO

Algoritmo: la Real Academia Española (RAE) define, en su primera acepción, el término algoritmo como «Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema».

Programar: idear y ordenar los pasos necesarios para la resolución de problemas mediante computadores.

2 Clasificación de los lenguajes de programación

Los lenguajes de programación se pueden clasificar según los siguientes criterios:

- Por el nivel de abstracción.
- Por el modo de ejecución.
- Por el paradigma de programación.
- Por el lugar de ejecución.

2.1. Clasificación por nivel de abstracción

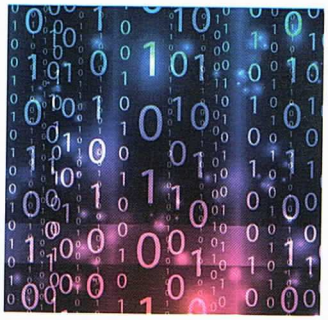
Según el nivel de abstracción, los lenguajes pueden disponer de **instrucciones que den acceso directo al *hardware* sobre el que se ejecutan los programas.** Conforme a este criterio tenemos lenguajes de bajo, medio y alto nivel.

- El **lenguaje de bajo nivel** por excelencia es el **ensamblador,** que permite acceder directamente al procesador, la memoria, los puertos y al resto del *hardware*.
- Los **lenguajes de nivel medio** proporcionan un **mayor nivel de abstracción que el ensamblador,** al tiempo que permiten un acceso más directo al *hardware*. Suelen disponer de gran cantidad de librerías que ayudan a controlar los dispositivos y realizan todo tipo de funciones. Los lenguajes C y Forth están dentro de esta categoría.
- Los **lenguajes de alto nivel** forman una capa de abstracción sobre el *hardware* y **tienen una sintaxis más cercana al lenguaje humano.** Estos lenguajes manejan distintos tipos de datos y estructuras, disponiendo de librerías de funciones que permiten usar el *hardware* sin preocuparse de los detalles que lo hacen funcionar. Existen muchos lenguajes en esta categoría, por ejemplo Ada o Java.

TEN EN CUENTA

Nivel de abstracción de los lenguajes de programación

Algunos autores sostienen que solo hay lenguajes de alto y bajo nivel, en cuyo caso, los lenguajes de nivel medio pasan a engrosar la lista de lenguajes de alto nivel.



2.2. Clasificación por el modo de ejecución

Teniendo en cuenta el modo de ejecución, los lenguajes **se clasifican en interpretados y compilados.** Cualquier lenguaje puede ser interpretado o compilado.

Un **intérprete** es una **aplicación** que se carga en la memoria del ordenador y **sobre la que se pueden escribir programas** en el lenguaje del intérprete.

Las características de los programas interpretados son:

- **Ejecución en tiempo real.** Generan el código de la máquina a medida que progresa el programa.
- En teoría, **son independientes del *hardware* y del sistema operativo.** Se requiere un intérprete adecuado a cada *hardware* y sistema operativo.
- **Facilitan la depuración de errores** de programación al ver el resultado de forma inmediata.

- **Disponemos del código fuente,** lo que facilita su modificación.
- Por el contrario, **la generación del código máquina por parte del intérprete es ineficiente,** en cuanto al número y tipo de instrucciones del procesador.
- **El tiempo** necesario **para ejecutar los programas es más elevado** que en el caso de los programas compilados.

Un **compilador es** un programa que analiza el código escrito en un lenguaje de programación y lo traduce a un **programa ejecutable** en el lenguaje de máquina.

Características de un programa compilado:

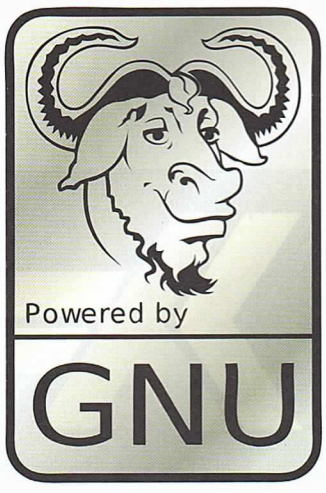
- **No se necesita un programa** adicional para su **ejecución.**
- **Dependiente del *hardware*** y del sistema operativo.
- **Si se producen errores** de ejecución hay que **contactar con el programador,** o empresa fabricante, para que los resuelva.
- En principio, **no se conoce el código fuente,** a menos que nos lo proporcionen aparte del ejecutable.
- **El compilador,** de forma automática, **trata de generar un código lo más eficiente posible,** en cuanto al número y tipo de instrucciones de máquina.
- El **tiempo de ejecución depende** directamente **del *hardware.***
- Si no se dispone del código fuente, hay que confiar en que **el programa hace solo aquello para lo que está diseñado.**

Grace Murray Hopper. Diseñadora del primer compilador.

TICO++

FSF

FSF son las siglas de Free Software Foundation, organización sin ánimo de lucro dedicada a la promoción del *software* libre. El *software* libre se define como aquel que permite al usuario cinco libertades básicas: copiarlo, distribuirlo, estudiarlo, modificarlo y mejorarlo. Es evidente que para las tres últimas es preciso disponer del código fuente.



El proyecto GNU está patrocinado por la FSF.

TICO++

Lenguajes intermedios

Existen algunos lenguajes, como Java, que necesitan una máquina virtual para ejecutar el código intermedio que generan a partir del código fuente. Este código intermedio *(bytecode)* es una versión precompilada del código fuente original, preparada para ejecutarse sobre la máquina virtual. Este código es independiente del *hardware* y del sistema operativo.

Actividades

8. Busca información y realiza una tabla comparativa en tu cuaderno con las características de programas interpretados y compilados.
9. Busca la biografía de Grace Murray Hopper y crea tu propia línea temporal con los hitos de su vida. Recoge las anécdotas más importantes y los datos que más te hayan llamado la atención. ¿Qué destacarías de su vida? Puedes hacerlo mediante alguna herramienta *online*, como Dipity, para publicar directamente tu trabajo en Internet.
10. Grace Murray Hopper participó en el desarrollo inicial del lenguaje COBOL. Este lenguaje aún se sigue usando en grandes computadores. Busca datos sobre el lenguaje COBOL y las características por las que ha resultado tan importante en la historia de la informática.

2.3. Clasificación por el paradigma de programación

El **paradigma de programación** es un marco teórico o modelo de organización que define un estilo propio de desarrollo **a través del cual resuelve los problemas.** Todos los lenguajes se aproximan más o menos a determinados paradigmas de programación. La primera gran división es entre **programación imperativa y declarativa.**

La **programación imperativa** consiste en **escribir sentencias e instrucciones precisas** que, al ejecutarse paso a paso, hacen que la máquina vaya cambiando de estado hasta solucionar el problema. El diseño de los microprocesadores sigue este paradigma, de modo que el lenguaje de máquina es también imperativo. Hay dos elementos claves que forman parte de un lenguaje imperativo:

- Las **variables,** con sentencias para **asignarles valores.**
- Las **estructuras de control,** que permiten **seleccionar alternativas o repetir la ejecución** de determinadas sentencias.

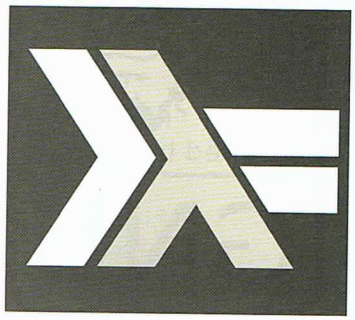
Entre los lenguajes más conocidos del paradigma imperativo están C, Pascal y Java.

La **programación declarativa** se basa en la **creación de programas por especificación de las reglas** que describen el problema y la forma de solucionarlo. Dentro de este paradigma se engloban otros más específicos como son la programación lógica y la programación funcional.

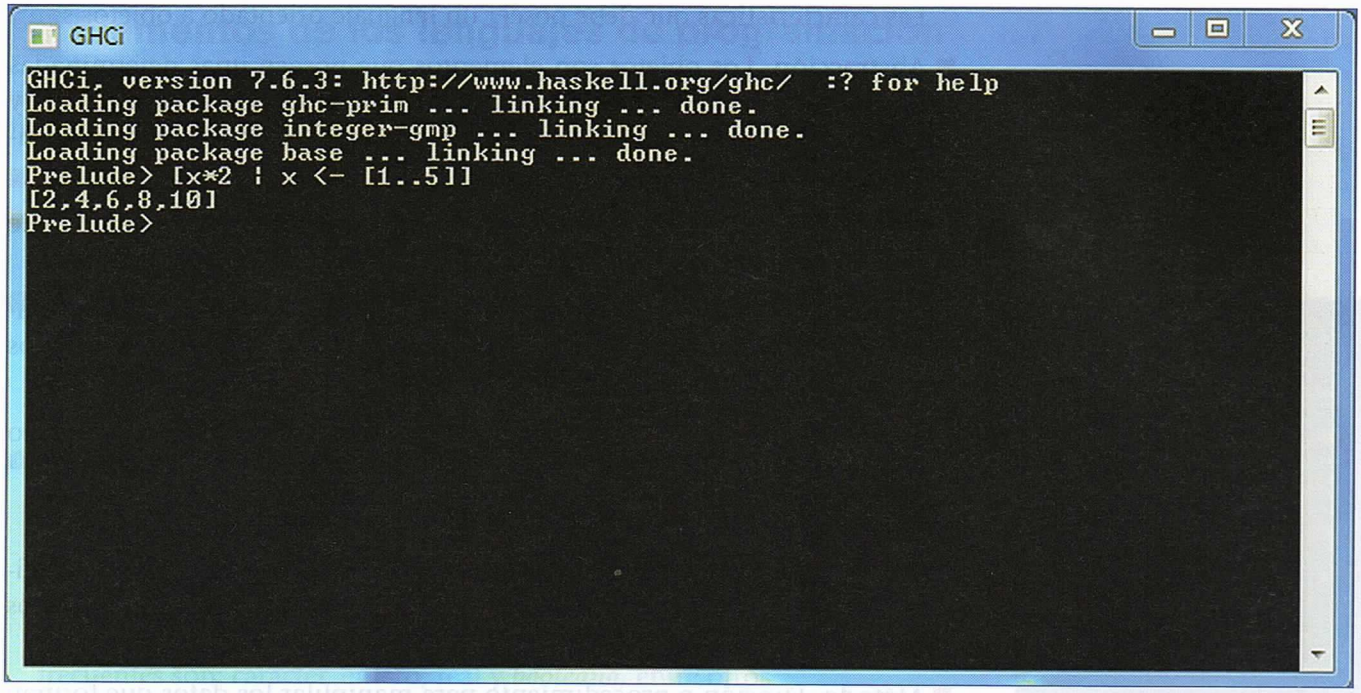
Entre los lenguajes más conocidos del paradigma declarativo están Lisp, Prolog, Haskell y SQL. Haskell, por ejemplo, tiene una sintaxis muy cercana al lenguaje matemático y permite definir de forma simple conjuntos.

Ejemplo:

Sea S el conjunto de los primeros cinco números naturales pares. En matemáticas usaríamos la notación: $s=\{x*2|x \in N, x \leq 5\}$, y el código Haskell correspondiente es `[x*2 | x <- [1..5]]`. Al ejecutarlo con un intérprete de Haskell nos daría como resultado `[2,4,6,8,10]`.



El lenguaje Haskell se encuadra en el paradigma funcional.



GHCi, intérprete de Haskell ejecutando el código para crear el conjunto del ejemplo.

Actividades

11. Existen otros paradigmas importantes, como la programación modular y la programación orientada a objetos. Busca al menos tres paradigmas y prepara una comparativa entre sus características.

12. Los dispositivos móviles también se programan. Busca qué lenguajes se emplean para hacerlo y encuádralos dentro de alguno de los paradigmas conocidos.

13. La programación declarativa produce código de mucho menor tamaño que su contraparte imperativa. Busca un ejemplo de programa sencillo escrito en Java y en Haskell que resuelva el mismo problema, por ejemplo, calcular las raíces de una ecuación de segundo grado. Compara los dos ejemplos y trata de seguir los pasos que daría la máquina para ejecutarlos. ¿Cuál resulta más fácil de comprender?, ¿por qué crees que predomina la programación imperativa sobre la declarativa?

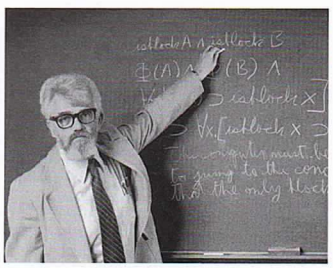
La programación orientada a objetos es otro de los paradigmas más influyentes en los lenguajes de programación de los últimos años. Tanto es así, que las revisiones y actualizaciones que se han hecho en numerosos lenguajes han incorporado este paradigma al resto de características que ya poseía el lenguaje.

Los objetos son bloques abstractos, contenedores de información que se almacenan junto con los métodos que permiten manipularlos. Antes de profundizar en la explicación, conviene conocer tanto las características como los elementos fundamentales del paradigma.

TEN EN CUENTA

Lenguajes multiparadigma

Algunos lenguajes no pueden encuadrarse estrictamente dentro de un paradigma, sino que tienen características de varios de ellos. Por ejemplo, Lisp, además de ser un lenguaje funcional y, por tanto, declarativo, también se considera orientado a objetos.



John McCarthy diseñó la especificación original de Lisp.

Las **características** que debe poseer un lenguaje orientado a objetos son:

- **Abstracción.** Los objetos son elementos que poseen unas determinadas propiedades. La abstracción consiste en generalizar estas propiedades y atributos aplicables sobre un grupo de objetos y olvidarse momentáneamente de los detalles.
- **Encapsulación.** Consiste en mantener juntos los datos y los métodos que permiten manipularlos.
- **Herencia.** Los objetos pertenecen a clases y heredan de estas sus propiedades. Hay lenguajes que permiten la herencia múltiple, esto es, los objetos pueden heredar propiedades de más de una clase.
- **Polimorfismo.** Se refiere al empleo de un mismo método para el manejo de distintos tipos de datos.

Los **elementos fundamentales** del paradigma orientado a objetos son:

- **Clase.** Bloque de código en el que se definen las propiedades y características de un objeto. A partir de la definición se crean (instancian) los objetos necesarios.
- **Método.** Función o procedimiento para manipular los datos que forman parte de un objeto. Los métodos se definen dentro de las clases y solo pueden ser aplicados a los objetos pertenecientes a esa clase.
- **Objeto.** Ente abstracto dotado de ciertas características (datos) y de los métodos para manipularlos (funciones). Todos los objetos son instancias de una clase, en el caso de lenguajes, que admiten herencia múltiple.

Algunos lenguajes orientados a objetos son C++, Java, Objective-C y PHP.

2.4. Clasificación por el lugar de ejecución

El desarrollo de la computación en la nube ha traído sustanciales cambios en el mundo de las TIC. A partir de la primera década del siglo XXI, cada vez es más común alojar nuestros datos en servidores ubicados en lugares que desconocemos, frente al almacenamiento local en nuestros propios ordenadores.
La ejecución de programas se lleva a cabo a través de los siguientes pasos:

- **Ejecución local.** Datos y programas residen en la máquina del usuario y es esta la encargada de procesarlos.
- **Ejecución remota.** Los programas que procesarán la información residen en un servidor remoto. Los datos que el usuario suministra a los programas pueden estar en su dispositivo o almacenados en la nube. En cualquier caso, los resultados se mostrarán al usuario en su dispositivo y él elegirá si se almacenan en remoto, en local o de ambos modos.

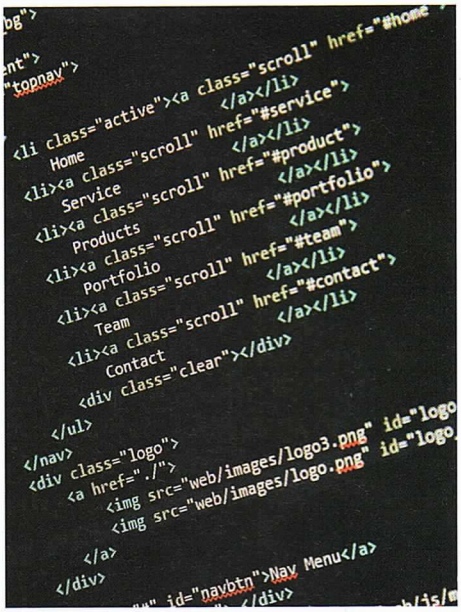
Actividades

14. Elabora un análisis de los riesgos que suponen para la privacidad el alojamiento y procesamiento de datos en la nube.
15. Cuando los programas y datos están alojados localmente, es responsabilidad del usuario hacer un mantenimiento de la máquina *(hardware* y *software).* Busca qué recomendaciones se deben seguir para mantener los datos seguros.

3 Elementos de los lenguajes de programación

Los lenguajes de programación nos sirven para definir procesos que podemos llevar a cabo en máquinas, para ello contamos con:

- **Identificadores.** Son combinaciones de letras, números y/o algún otro símbolo, como el guión (-) o el subrayado (_), que se asignan por el programador a elementos como variables o procedimientos.
- **Palabras reservadas.** Son identificadores que ya están asignados previamente a instrucciones, funciones u operadores propios del lenguaje. No se pueden utilizar para designar variables ni elementos desarrollados por el programador.
- **Tipos de datos.** Son atributos que le indican al ordenador cómo debe procesar determinada información. Los lenguajes de alto nivel nos permiten trabajar con mucho más que números, que es lo que emplea el microprocesador. Los lenguajes modernos disponen de una amplia variedad de tipos de datos predefinidos, siendo habitual que permitan al programador definir tipos nuevos a partir de los disponibles. Algunos de los tipos más frecuentes son: carácter, entero, real y ***booleano***, entre otros.
- **Variables.** Como en matemáticas, una variable es un elemento que contiene un tipo de dato. Se emplean para realizar cálculos, controlar condiciones y bucles, y almacenar información. Algunos lenguajes de programación obligan a la definición de variables en lugares concretos del programa, e incluso a la asignación del tipo de datos. Todas estas condiciones se indican en el manual del lenguaje.
- **Estructuras de control.** Son elementos que permiten decidir cómo continuará la ejecución de un programa, dependiendo del estado de una variable. Se dividen en:
  - **Estructuras de selección.**
    - IF condición, THEN bloque de sentencias.
    - IF condición, THEN primer bloque de sentencias, ELSE segundo bloque de sentencias.
    - SELECT-CASE.
  - **Estructuras de repetición.** También llamadas bucles.
- **Operadores.** Como en matemáticas, nos permiten trabajar sobre las variables, compararlas y realizar operaciones con ellas. Además de los operadores aritméticos y de comparación, tenemos otros que permiten la manipulación de los distintos tipos de datos.



Los lenguajes de programación están formados por símbolos y reglas (sintácticas y semánticas) para definir sus estructuras, elementos y expresiones.

Actividades

16. Existen diversas estructuras de control. Busca cinco de ellas y escribe ejemplos cotidianos en los que empleamos estos esquemas, como la toma de decisiones o la repetición de tareas.

17. En muchas ocasiones se emplean diagramas de flujo como introducción a la programación de computadores. Busca los esquemas asociados a las estructuras de control y dibújalos en tu cuaderno. También puedes usar el programa **Dia** para hacerlo.