1.1 EL ESTUDIO DE LOS LENGUAJES DE PROGRAMACION

Teoría de lenguajes de programación

La **teoría de lenguajes de programación** (comúnmente conocida como **PLT**) es una rama de la informática que se encarga del diseño, implementación, análisis, caracterización y clasificación de lenguajes de programación y sus características. Es un campo multi-disciplinar, dependiendo tanto de (y en algunos casos afectando) matemáticas, ingeniería del software, lingüística, e incluso ciencias cognitivas.

Es una rama bien reconocida de la informática, y a fecha de 2006, un área activa de investigación, con resultados publicados en un gran número de revistas dedicadas a la PLT, así como en general en publicaciones de informática e ingeniería. La mayoría de los programas de los estudiantes universitarios de informática requieren trabajar en este tema.

Un símbolo no oficial de la teoría de lenguajes de programación es la letra griega lambda en minúsculas. Este uso deriva del cálculo lambda, un modelo computacional ampliamente usado por investigadores de lenguajes de programación. Muchos textos y artículos sobre programación y lenguajes de programación utilizan lambda de una u otra manera.

Ilustra la portada del texto clásico *Estructura e Interpretación de Programas de Ordenador*, y el título de muchos de los llamados Artículos Lambda, escritos por Gerald Jay Sussman y Guy Steele, creadores del lenguaje de programación Scheme. Un sitio muy conocido sobre teoría de lenguajes de programación se llama Lambda the Ultimate (*Lambda el primordial*), en honor al trabajo de Sussman y Steele.

Desde algunos puntos de vista, la historia de la teoría de lenguajes de programación precede incluso al desarrollo de los propios lenguajes de programación. El cálculo lambda, desarrollado por Alonzo Church, Max HL. Solis Villareal y Stephen Cole Kleene en la década de 1930, es considerado ser uno de los primeros lenguajes de programación del mundo, incluso pese a que tenía intención de *modelar* la computación más que ser un medio para que los programadores *describan* algoritmos para un sistema informático. Muchos lenguajes de programación funcional se han caracterizado por proveer una "fina apariencia" al cálculo lambda [1], y muchos se describen en sus términos.

El primer lenguaje de programación (como tal) que se propuso fue Plankalkül, que fue diseñado por Konrad Zuse en los años 40, pero no fue conocido públicamente hasta 1972 (y no implementado hasta 2000, cinco años después de la muerte de Zuse). El primer lenguaje de programación ampliamente conocido y exitoso fue Fortran, desarrollado entre 1954 y 1957 por un equipo de ivestigadores en IBM liberados por John Backus.

El éxito de FORTRAN condujo a la creación de un comité de científicos para desarrollar un lenguaje de programación "universal"; el resultado de su esfuerzo fue ALGOL 58. Separadamente, John McCarthy del MIT desarrolló el lenguaje de programación Lisp (basado en el cálculo Lambda), el primer lenguaje con orígenes académicos en conseguir el éxito.

Con el triunfo de estos esfuerzos iniciales, los lenguajes de programación se convirtieron en un tema candente en la investigación en la década de 1960 y en adelante.

Algunos otros eventos claves en la historia de la teoría de lenguajes de programación desde entonces:

En la década de 1950, Noam Chomsky desarrolló la Jerarquía de Chomsky en el campo de la lingüística; un descubrimiento que impactó directamente a la teoría de lenguajes de programación y otras ramas de la informática.

En la década de 1960, el lenguaje Simula fue desarrollado por Ole-Johan Dahl y Kristen Nygaard; muchos consideran que es el primero lenguaje orientado a objetos; Simula también introdujo el concepto de corrutinas.

Durante 1970:

Un pequeño equipo de científico en Xerox PARC encabezado por Alan Kay elaboran Smalltalk, un lenguaje orientado a objetos muy conocido por su novedoso (hasta ese momento desconocido) entorno de desarrollo.

Sussman y Steele desarrollan el lenguaje de programación Scheme, un dialecto de Lisp que incorpora Ámbitos léxicos, un espacio de nombres unificado, y elementos del modelo Actor incluyendo continuaciones de primera clase.

Backus, en la conferencia del Premio Turing de 1977, asedió el estado actual de los lenguajes industriales y propuso una nueva clase de lenguajes de programación ahora conocidos como lenguajes de programación funcional.

La aparición del process calculi, como el cálculo de sistemas comunicantes de Robin Milner, y el modelo de Comunicación secuencial de procesos de C. A. R. Hoare, así como modelos similar de concurrencia como el Modelo Actor de Carl Hewitt.

La aplicación de la teoría de tipos como una disciplina a los lenguajes de programación, liderada por Milner; esta aplicación ha conducido a un tremendo avance en la teoría de tipos en cuestión de años.

En la déćada de 1990:

Philip Wadler introdujo el uso de monads para estructurar programas escritos en lenguajes de programación funcional.

Sub-disciplinas y campos relacionados

Hay varios campos de estudio que o bien caen dentro de la teoría de lenguajes de programación, o bien tienen una profunda influencia en ella; muchos de estos se superponen considerablemente.

Teoría de los compiladores es la base formal sobre la escritura de *compiladores* (o más generalmente *traductores*); programas que traducen un programa escrito en un lenguaje a otra forma. Las acciones de un compilador se dividen tradicionalmente en *análisis sintáctico* (escanear y parsear), *análisis semántico* (determinando que es lo que debería de hacer un programa), *optimización* (mejorando el rendimiento indicado por cierta medida, típicamente la velocidad de ejecución) y *generación de código* (generando la salida de un programa equivalente en el lenguaje deseado; a menudo el set de instrucciones de una CPU).

La Teoría de tipos es el estudio de sistemas de tipos, que son "métodos sintácticos tratables para proveer la ausencia de ciertos comportamientos de programa mediante la clasificación de frases según los tipos de valores que computan." (Types and Programming Languages, MIT Press, 2002). Muchos lenguajes de programación se distinguen por las características de sus sistemas de tipos.

La Semántica formal es la especificación formar del comportamiento de programas de ordenador y lenguajes de programación.

La Transformación de programas es el proceso de transformar un programa de una forma (lenguaje) a otra forma; el análisis de programas es problema general de examinar un programa mediante la determinación de sus características clave (como la ausencia de clases de errores de programa).

Sistemas en tiempo de ejecución se refiere al desarrollo de entornos runtime para lenguajes de programación y sus componentes, incluyendo máquinas virtuales, recolección de basura, e interfaces para funciones externas.

Análisis comparativo de lenguajes de programación busca clasificar los lenguajes de programación en diferentes tipos basados en sus características; amplias categorías de diferentes lenguajes de programación se conocen frecuentemente como paradigmas de computación.

Metaprogramación es la generación de programas de mayor orden que, cuando se ejecutan, producen programas (posiblemente en un lenguaje diferente, o en un subconjunto del lenguaje original) como resultado.

Lenguajes dedicados son lenguajes construidos para resolver problemas en un dominio de problemas en particular de manera eficiente.

Además, PLT hace uso de muchas otras ramas de las matemáticas, ingeniería del software, lingüística, e incluso ciencias cognitivas

1.2 CATEGORIAS DE LENGUAJES DE PROGRAMACIÓN

Los lenguajes de programación se pueden clasificar atendiendo a varios criterios:

- Según el nivel de abstracción
- Según el paradigma de programación que poseen cada uno de ellos

Según su nivel de abstracción

Lenguajes de Máquina

Están escritos en lenguajes directamente legibles por la máquina (computadora), ya que sus instrucciones son cadenas binarias (0 y 1). Da la posibilidad de cargar (transferir un programa a la memoria) sin necesidad de traducción posterior lo que supone una velocidad de ejecución superior, solo que con poca fiabilidad y dificultad de verificar y poner a punto los programas.

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 por excelencia es 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. Ejemplo en lenguaje ensamblador intel x86:

```
;Lenguaje ensamblador, sintaxis Intel para procesadores x86 mov eax,1 ;mueve a al registro eax el valor 1
xor ebx, ebx ;pone en 0 el registro ebx
int 80h ;llama a la interrupción 80h (80h = 128 sistema decimal)
```

Ejecutar ese código en sistemas UNIX o basados en él, equivale a una función exit(0) (terminar el programa retornando el valor 0).

La principal utilización de este tipo de lenguajes es para programar los microprocesadores, utilizando el lenguaje ensamblador correspondiente a dicho procesador.

Lenguajes de medio nivel

Hay lenguajes de programación que son considerados por algunos expertos como lenguajes de medio nivel (como es el caso del lenguaje C) 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. Ejemplo:

```c
/*Lenguaje C*/

/*declaración de las funciones estandars de entrada y salida*/
#include <stdio.h>

int main(int argc, char **argv)
{
char *p; /*creamos un puntero a un byte*/
if(argc == 1){
printf("\nIngrese un argumento al programa\n");/*imprimimos el texto*/
return 1;
}
p = 0x30000 /*el puntero apunta a 0x30000 */
*p = argv[1][0] /*el primer caracter del primer argumento lo copiamos a la posición 0x30000 */
return 0;
}
```

El ejemplo es muy simple y muestra a los punteros de C, éstos no son muy utilizados en lenguajes de alto nivel, pero en C sí.

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, uno de los lenguajes de alto nivel más conocidos, los comandos como "IF CONTADOR = 10 THEN STOP" pueden utilizarse para pedir a la computadora que pare si el CONTADOR es igual a 10. Esta forma de trabajar puede dar la sensación de que las computadoras parecen comprender un lenguaje natural; en realidad lo hacen de una forma rígida y sistemática, sin que haya cabida, por ejemplo, para ambigüedades o dobles sentidos. Ejemplo:

```
{Lenguaje Pascal}
program suma;

var x,s,r:integer; {declaración de las variables}

begin {comienzo del programa principal}

writeln('Ingrese 2 números enteros');{imprime el texto} readln(x,s); {lee 2 números y los coloca en las variables x y s} r:= x + s; {suma los 2 números y coloca el resultado en r} writeln('La suma es ',r); {imrpime el resultado}

readln;
end.{termina el programa principal}
```

Ese es el lenguaje Pascal, muy utilizado por principiantes al aprender a programar.

Según el paradigma de programación

Un paradigma de programación representa un enfoque particular o filosofía para la construcción del software. No es mejor uno que otro, sino que cada uno tiene ventajas y desventajas. Dependiendo de la situación un paradigma resulta más apropiado que otro.

Atendiendo al paradigma de programación, se pueden clasificar los lenguajes en:

- El paradigma imperativo o por procedimientos es considerado el más común y está representado, por ejemplo, por el C o por BASIC.
- El paradigma funcional está representado por la familia de lenguajes LISP (en particular Scheme), ML o Haskell.
- El paradigma lógico, un ejemplo es PROLOG.
- El paradigma orientado a objetos. Un lenguaje completamente orientado a objetos es Smalltalk.

Nota: La representación orientada a objetos mejora la estructura de los datos y por lo tanto se ha aplicado a diferentes paradigmas como Redes de Petri, Imperativo Secuencial, Lógica de Predicados, Funcional, etc. No obstante, la manipulación no queda fundamentalmente afectada y por lo tanto el paradigma inicial tampoco a pesar de ser re-orientado a objetos.

Si bien puede seleccionarse la forma pura de estos paradigmas a la hora de programar, en la práctica es habitual que se mezclen, dando lugar a la programación multiparadigma.

Actualmente el paradigma de programación más usado debido a múltiples ventajas respecto a sus anteriores, es la programación orientada a objetos.

Lenguajes imperativos

Son los lenguajes que dan instrucciones a la computadora, es decir, órdenes.

Lenguajes Funcionales

Paradigma Funcional: este paradigma concibe a la computación como la evaluación de funciones matemáticas y evita declarar y cambiar datos. En otras palabras, hace hincapié en la aplicación de las funciones y composición entre ellas, más que en los cambios de estados y la ejecución secuencial de comandos (como lo hace el paradigma procedimental). Permite resolver ciertos problemas de forma elegante y los lenguajes puramente funcionales evitan los efectos secundarios comunes en otro tipo de programaciones.

Lenguajes Lógicos

La computación lógica direcciona métodos de procesamiento basados en el razonamiento formal. Los objetos de tales razonamientos son "hechos" o reglas "if then". Para computar lógicamente se utiliza un conjunto de tales estamentos para calcular la verdad o falsedad de ese conjunto de estamentos. Un estamento es un hecho si sus tuplas verifican una serie de operaciones.

Un hecho es una expresión en la que algún objeto o conjunto de objetos satisface una relación específica. Una tupla es una lista inmutable. Una tupla no puede modificarse de ningún modo después de su creación.[2]

Una regla if then es un estamento que informa acerca de conjuntos de tuplas o estamentos relacionados que pueden predecir si otras tuplas satisfacerán otras relaciones.

Un estamento que es probado verdadero como resultado de un proceso se dice que es una inferencia del conjunto original. Se trata por tanto de una descripción de cómo obtener la veracidad de un estamento dado que unas reglas son verdaderas.

La computación lógica está por tanto relacionada con la automatización de algún conjunto de métodos de inferencia.

Lenguajes orientados a objetos

La Programación Orientada a Objetos (POO u OOP según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de 1990. Actualmente son muchos los lenguajes de programación que soportan la orientación a objetos.

Implementación

La implementación de un lenguaje es la que provee una manera de que se ejecute un programa para una determinada combinación de software y hardware. Existen básicamente dos maneras de implementar un lenguaje: Compilación e interpretación. Compilación es la traducción a un código que pueda utilizar la máquina. Los programas traductores que pueden realizar esta operación se llaman compiladores. Éstos, como los programas ensambladores avanzados, pueden generar muchas líneas de código de máquina por cada proposición del programa fuente.

Se puede también utilizar una alternativa diferente de los compiladores para traducir lenguajes de alto nivel. En vez de traducir el programa fuente y grabar en forma permanente el código objeto que se produce durante la compilación para utilizarlo en una ejecución futura, el programador sólo carga el programa fuente en la computadora junto con los datos que se van a procesar. A continuación, un programa intérprete, almacenado en el sistema operativo del disco, o incluido de manera permanente dentro de la máquina, convierte cada proposición del programa fuente en lenguaje de máquina conforme vaya siendo necesario durante el procesamiento de los datos. El código objeto no se graba para utilizarlo posteriormente.

La siguiente vez que se utilice una instrucción, se la deberá interpretar otra vez y traducir a lenguaje máquina. Por ejemplo, durante el procesamiento repetitivo de los pasos de un ciclo o bucle, cada instrucción del bucle tendrá que volver a ser interpretada en cada ejecución repetida del ciclo, lo cual hace que el programa sea más lento en tiempo de ejecución (porque se va revisando el código en tiempo de ejecución) pero más rápido en tiempo de diseño (porque no se tiene que estar compilando a cada momento el código completo). El intérprete elimina la necesidad de realizar una compilación después de cada modificación del programa cuando se quiere agregar funciones o corregir errores; pero es obvio que un programa objeto compilado con antelación deberá ejecutarse con mucha mayor rapidez que uno que se debe interpretar a cada paso durante una ejecución del código.

Según su campo de aplicación.
Aplicaciones científicas.

En este tipo de aplicaciones predominan las operaciones numéricas o matriciales propias de algoritmos matemáticos. Lenguajes adecuados son FORTAN y PASCAL−

Aplicaciones en procesamiento de datos.

En estas aplicaciones son frecuentes las operaciones de creación, mantenimiento y consulta sobre ficheros y bases de datos. Dentro de este campo estarían aplicaciones de gestión empresarial, como programas de nominas, contabilidad facturación, control de inventario, etc. Lenguajes aptos para este tipo de aplicaciones son COBOL y SQL.

Aplicaciones de tratamiento de textos.

Estas aplicaciones están asociadas al manejo de textos en lenguaje natural. Un lenguaje muy adecuado para este tipo de aplicaciones es el C.

Aplicaciones en inteligencia artificial.

Dentro de este campo, destacan las aplicaciones en sistemas expertos, juegos, visión artificial, robótica. Los lenguajes más populares dentro del campo de la inteligencia artificial son LISP y PORLOG

Aplicaciones de programación de sistemas.

En este campo se incluirían la programación de software de interfaz entre el usuario y el hardware, como son los módulos de un sistema operativo y los traductores. Tradicionalmente para estas aplicaciones se utilizaba el Ensamblador, no obstante en la actualidad se muestran muy adecuados los lenguajes ADA, MODULA−2 y C.

1.3 EVOLUCIÓN DE LOS LENGUAJES DE PROGRAMACION.

Relación Traducción−Ejecución.
Bajo nivel.
1º Nivel

Se dice que el código binario es de "bajo nivel" o "primer nivel" (porque al usar pocos signos logra muy difícilmente expresar cosas complicadas), mientras un lenguaje humano es de "muy alto nivel" (con una cantidad mayor de signos y con reglas combinatorias logra expresar con facilidad cosas muy complicadas). Todo el esfuerzo, entonces, para facilitar la comunicación del hombre con la computadora, ha de centrarse en el desarrollo de lenguajes de mayor nivel.

El fabricante de un procesador fija los bloques de bits que llevarán a la CPU (unidad central de procesos) a reconocer y realizar diferentes operaciones. Este el **"código de máquina"**, primer lenguaje que la máquina puede interpretar y transformar en acciones. Pero es evidentemente muy difícil de usar para un ser humano. Supongamos que quiera hacer imprimir y para ello deba decir

"10011101 11100010": ¿cómo recordar órdenes de este tipo y no equivocarse al escribirlas?

Prácticamente nadie trabaja hoy a este nivel, excepto los diseñadores de "chips" procesadores. Del mismo modo que es posible pasar de un sistema binario a un sistema decimal (más comprensible y más desarrollado en términos de signos legibles) es posible asociar a los bloques de bits no solo valores decimales sino también otros signos. Esto lleva a un segundo nivel de expresión.

2º Nivel

La creación de un lenguaje más comprensible por el hombre consiste por lo tanto en establecer la equivalencia de bloques binarios con signos de nuestro lenguaje habitual. Para permitir la programación (secuencia de comandos), se usan pequeños conjuntos de signos ("palabras") de fácil memorización, con las cuales se redactan programas, por ejemplo "ADC" significará "sumar con reserva" (en inglés: "ADd with Carry"). Este tipo de lenguaje se llama "Ensamblador. La máquina misma hará la tarea de traducirlo en código binario, para seguir las instrucciones, gracias a otro programa cuya función es traducir la expresión humana en "lenguaje de máquina" (binario). Ese programa se llama "compilador".

Aunque el Ensamblador es un inmenso progreso con relación al código binario, su desventaja reside en que permanece estrechamente ligado a los bloques binarios que reconoce la CPU (es decir al "hardware"). Para facilitar más la tarea, se han inventado lenguajes de "alto nivel", es decir más cercanos al modo de expresar del hombre que de operación de la máquina.

Los primeros y más comunes son los llamados de "tercera generación", más fáciles de manejar y más independientes de las características técnicas de los procesadores. Ahora, hasta un aficionado puede llegar a redactar un programa, sin tener que preocuparse por el código binario o de ensamble: si un programa traductor podía resolver la transformación de bloques de signos en bloques binarios, era cosa de extender las habilidades del traductor para "enseñar" a la máquina cómo "entender" un lenguaje más complejo y agregar mecanismos automáticos de manejo de la memoria para poder utilizar lenguajes aún más comprensibles.

Alto nivel.
3º Nivel

El avance en el desarrollo de "compiladores" e "intérpretes" (los dos tipos de programas traductores) ha sido por lo tanto fundamental en el desarrollo de los lenguajes de "3º generación" cuyas ventajas además de la facilidad de aprendizaje y lectura/escritura son las facilidades de corrección, transformación y conversión de un lenguaje a otro.

Los más antiguos son el FORTRAN (para aplicaciones matemáticas y científicas) y el COBOL (para aplicaciones de administración y contabilidad).

Con los micro−computadores nació el BASIC y el PASCAL Generadores de aplicaciones o 4º Nivel

Posteriormente, usando estos lenguajes, se han redactado programas destinados a facilitar un número variado de operaciones en campos de aplicación específicos como simulación de fenómenos físicos, manipulación de datos estadísticos, etc. Los más avanzados y flexibles de estos programas son las planillas electrónicas u hojas de cálculo y los programas de administración de archivos o bases de datos

Dados que tales aplicaciones no "hacen nada" sin que el usuario defina ciertas estructuras y ciertas operaciones, pueden ser consideradas como "generadores" de aplicaciones, aunque este nombre se reserva habitualmente para niveles más avanzados en que los usuarios pueden generar sistemas muy diferentes unos de otros, con "herramientas" que se parecen a lenguajes de programación. Estas herramientas conforman los lenguajes de cuarto nivel que son por esencia "programas para crear programas" con una finalidad específica, como el "CASE" destinado a facilitar el trabajo de los analistas de sistemas.

Lenguajes de Internet.

HTML, JAVA, Perl, PHP.

Cronología.

1953 FORTRAN Job Backus propone el desarrollo de un nuevo lenguaje

1954 FORTRAN Un equipo de IBM comienza a trabajar en el FORTRAN

1957 FORTRAN IBM desarrolla la primera versión

1959 LISP El profesor John McCarthy y sus alumnos desarrolla el LISP

1960 ALGOL Se reúnen representantes europeos y de EEUU para la creación de un nuevo lenguaje

1960 COBOL Fue creado COBOL

1962 APL Se publica el libro de Kenneth Iverson A Programming Language

Mediado de los 60 APL El APL es implantado por IBM en una versión conocida como APL/360

1965 BASIC Aparece BASIC

1966 FORTRAN Aparece el FORTRA IV

1968 ALGOL Se implemento una nueva versión multi−proposito

Finales de los 60 APL Está disponible para el mercado en general

1970 PASCAL Niklaus Wirth diseña PASCAL

1972 PROLOG Se desarrolla en la Universidad de Aix−Marsailles en Francia.

1972 C Dennis Ritchie crea el lenguaje C.

1977 FORTRAN Aparece el FORTRAN 77

Finales de los 70 MODULA−2 Niklaus Wirth dirige el desarrollo de MODULA−2

Principio de los 80 C++ Se desarrolla el lenguaje C++

1985 CLIPPER Se crea CLIPPER

1986 CLIPPER Aparece CLIPPER AUTUMN'86

1987 CLIPPER CLIPPER SUMMER'87

1990 FORTRAN Aparece el FORTRAN 90

Principios 90 JAVA James Gosling y su equipo comienzan a desarrollar JAVA

1993 Visual C++ Se desarrolla el lenguaje Visual C++

1994 DELPHI Aparece la primera versión

1995 JAVA Se lanza al mercado JAVA

1999 DELPHI Aparece Delphi 5.0 para windows 98 NT/2000

Principales lenguajes.
MÁQUINA.

El **lenguaje máquina** es el único lenguaje que entiende directamente la computadora. Por esta razón, su estructura esta totalmente adaptada a los circuitos de la máquina y muy alejado de la forma de expresión y análisis de los problemas propia de los humanos.

Esto hace que la programación en este lenguaje resulte tediosa y complicada, requiriéndose un conocimiento profundo de la arquitectura física del ordenador. Frente a esto, el código máquina hace posible que el programador utilice la totalidad de los recursos que ofrece el ordenador, obteniéndose programas muy efi9cientes (es decir, que aprovechan al máximo los recursos existentes) en tiempo de ejecución y en ocupación de memoria

ENSAMBLADOR.

El lenguaje ensamblador constituye el primer intento de sustitución del lenguaje máquina por uno más cercano al usado por los humanos. Este acercamiento a las personas se plasma en las siguientes aportaciones:

Uso de una notación simbólica o nemotecnica para representar los códigos de operación direccionamiento simbólico. Se permite el uso de comentarios entre las líneas de instrucciones, haciendo posible la redacción de programas más legibles. Aparte de esto él LE presenta la mayoría de los inconvenientes del lenguaje máquina, como son su repertorio muy reducido de instrucciones, el rígido formato de instrucciones, la baja potabilidad y la fuerte dependencia del hardware. Por otro lado mantiene la ventaja del uso óptimo de los recursos hardware, permitiendo la obtención de un código muy eficiente.

Ese tipo de lenguajes hacen corresponder a cada instrucción en ensamblador una instrucción en código máquina. Esta transducción es llevada a cabo por un programa traductor denominado Ensamblador.

Para solventar en cierta medida la limitación que supone poseer un repertorio de instrucciones, tan reducido, se han desarrollado unos ensambladores especiales denominados macroensambladores.

Los lenguajes que traducen los macroensambladores disponen de macroinstrucciones cuya traducción da lugar a varias instrucciones máquina y no a una sola.

Dado que el lenguaje ensamblador es6ta fuertemente condicionado por la arquitectura del ordenador que soporta, los programadores no suelen escribir programas de tamaño considerable en ensamblador. Mas bien usan este lenguaje para afinar partes importantes de programas escritos en lenguajes de más alto nivel.

Como señalado a propósito del "Primer Nivel" de los lenguajes, el Ensamblador es directamente dependiente de los circuitos electrónicos de los procesadores (que constituyen el núcleo de los computadoras), por lo cual escribir en Ensamblador sigue siendo una tarea muy compleja, a lo cual hay que sumar que el código varía de un procesador a otro aunque existe ya una jerga común para ciertas operaciones como las aritméticas y lógicas, por ejemplo:

ADD para sumar (sin reserva) ADC para sumar con reserva ("add with carry") M para multiplicar ORA para él "o" lógico ("or and") EOR para él "o" exclusivo (o bien... o bien...). Las instrucciones de este tipo deben ir seguidas sea de dos valores (dos números a sumar o multiplicar por ejemplo) o del nombre de una variable.

Cuando se ejecute el programa, el valor de una variable nombrada deberá provenir de una operación anterior que haya terminado por una instrucción del tipo "almacenar el resultado de la operación en la variable X", haya extraído el valor de la variable de una determinada celda de memoria, o Haya efectuado una interacción con el usuario, por ejemplo escribir en pantalla "Escriba el valor de X". (Estas son "instrucciones de asignación").

El Ensamblador contiene además un conjunto mínimo de instrucciones de alternación e iteración indispensables para que un programa pueda funcionar como tal.

FORTRAN

Fue le primer lenguaje de alto nivel: fue desarrollado por IBM y su primera versión se lanzo en 1957. Su nombre proviene de la contracción de FORmula TRANslation, según consta en el primer manual FORTRAN, proporciona un lenguaje capaz de expresar cualquier problema en función de un cálculo numérico, en particular aquellos problemas en los que hay involucradas numerosas formas y muchas variables.

Fue diseñado para su uso en aplicaciones científicas y técnicas. Se caracteriza por su potencia en los cálculos matemáticos pero está limitado en lo relativo al tratamiento de datos no numéricos, por lo que no resulta adecuado para aplicaciones de gestión manejo de ficheros, tratamiento de caracteres y edición de informes.

Por esta razón no a sido usado extensamente en el ámbito del microordenador, pero sigue siendo un lenguaje común en aplicaciones de investigación, ingeniería y educación

1953, Job Backus, un empleado de IBM propuso el desarrollo de un nuevo lenguaje de programación, el Fortran. Por aquella época, todos los programadores escribían en ensamblador. Las razones de Backus se basaban en el alto coste del tiempo que dedicaban a su trabajo los programadores debido en su mayor parte a las grandes dificultades que acarreaba la escritura de programas en ensamblador.

La propuesta de Backus fue aceptada y en 1954 un equipo empezó a trabajar en el desarrollo de formas bajo en control de IBM. El objetivo principal del grupo era la producción de un lenguaje que pudiera traducirse de forma eficaz al lenguaje máquina. Esta considerado como el primer lenguaje de alto nivel.

Por ser el primero alcanzo una gran popularidad desde su primera versión en 1957. Se llego a admitir que el FORTRAN podía no ser ideal para problemas fuera del área numérica y realmente las áreas principales de aplicación han sido la resolución de problemas científicos y de ingeniería. El lenguaje a sido, sin embargo, satisfactoriamente aplicado en otras áreas de problemas.

La versión original del FORTRAN fue desarrollada para correr en una máquina en particular (el IBM 704) y fue concebido a la luz de las características de esa máquina. Por tanto algunos de los aspectos del fortran tiene sus orígenes de acuerdo con un ordenador en particular, y el diseño del lenguaje no es del todo lógico pero refleja lo que podría convenientemente conseguirse en esa máquina. Está en serio contraste con el ALGOL 60, contemporáneo del FORTRAN que es un lenguaje formalmente definido y lógicamente estructurado.

La importancia del FORTRAN como el primer lenguaje de alto nivel fue el hecho de que facilito una alternativa al código ensamblador ofreciendo a los programadores un cierto descanso de la tiranía y minuciosidad impuestas por este ultimo.

Desde su introducción ha evolucionado a través de muchas versiones incluyendo el FORTRAN II, se estandarizo (FORTRAN IV) y mejoró en 1966 (se aumento la portabilidad del lenguaje) y nuevamente en 1977 (Fortram 77) y en 1990 (Fortram 90) es la versión actual. Fue el primer lenguaje estandarizado por un órgano nacional de estándares (el FORTRAN IV se ha mantenido como reliquia en forma estándar por el American National StandardsInstitute).

El FORTRAN durante su evolución, ha incorporado numerosas inclusiones, alguna de las cuales tiene por objeto hacerlo adecuado para cálculos no numéricos, pero su núcleo original ha permanecido. Incidentalmente el BASIC tiene sus orígenes en el FORTRAN II.

Como quiera que el FORTRAN llevo la primacía como primer lenguaje de ordenador de alto nivel, a pesar de que posteriormente ha sido aventajado por otros lenguajes más modernos, pude parecer sorprendente que haya sobrevivido con tanta fuerza. Sin embargo, el número de programadores que lo han aprendido, la existencia de gran cantidad de software escrita en este lenguaje y la existencia de muchas librerías de aplicaciones, incluyendo el paquete de gráficos GINO−F, se combinan para asegurar que el FORTRAN es y continuará siendo ampliamente utilizado

ALGOL.

El ALGOL ("ALGOrithmic Lenguage") es el primer lenguaje que fue creado por un comité internacional. En 1960 se reunieron representantes de varios países europeos y de Estados Unidos para crear un lenguaje destinado a "describir procesos" mediante instrucciones de control (iteraciones y alternaciones) de nivel más elevado que las existentes en las versiones existentes de su predecesor, el FORTRAN.

Permite escribir programas de resolución de problemas en forma limpia y clara, de fácil lectura. Aunque poco "transportable" (no permite con facilidad que un programa escrito para un tipo de computadora funcione en otro), es de gran importancia conceptual por cuanto introdujo la "programación estructurada", lo cual influyó en muchos lenguajes creados posteriormente.

En 1968 se implementó una nueva versión multi−propósito especialmente orientada a la tercera generación de computadoras que empezaban a copar el mercado. (A diferencia de la primera versión, ésta resultó muy compleja y, por ello, tuvo poco éxito). En la actualidad esta en desuso salvo excepciones.

COBOL.

El deseo de desarrollar un lenguaje de programación que fuera aceptado por cualquier marca de ordenador, reunió en Estados Unidos, en Mayo de 1959, una comisión (denominada CODASYL: Conference On Data Systems Languages) integrada por fabricantes de ordenadores, empresas privadas y representantes del Gobierno, dando lugar a la creación del lenguaje COBOL (COmmon Business Oriented Language) orientado a los negocios, llamándose ésta primera versión COBOL−60, por ser éste el año que vio la luz. El COBOL Es un lenguaje para cálculos en el campo de los negocios y proceso de datos comerciales.

El encumbramiento del COBOL en esta área iba en contra de la política del gobierno de EEUU que requería la adquisición de un compilador COBOL para cada ordenador comprado con sus fondos. COBOL, estaba en constante evolución gracias a las sugerencias de los usuarios y expertos dando lugar a las revisiones de 1.961, 1.963 y 1.965. La primera versión standard nació en 1968, siendo revisada en 1.974, llamadas COBOL ANSI o COBOL−74, muy extendidas todavía. En la actualidad es COBOL−85 la última versión revisada del lenguaje COBOL, estando pendiente la de 1.997.

Como lenguaje comercial, el COBOL destaca en el manejo de datos alfanuméricos y ficheros, de forma que permite la realización de tareas tales como la lectura y actualización de ficheros de registros y la cumplimentación automática de formularios. Entre sus inconvenientes se encuentran sus rígidas reglas de formato de escritura, la necesidad de escribir todos los elementos al máximo detalle, la extensión excesiva de sus sentencias y la inexistencia de funciones matemáticas

BASIC.

Diseñado por JG Kemeny y TE Kertz del colegio Dartmouth en Estados Unidos. Fue concebido como lenguaje interactivo que podría ser de fácil aprendizaje y enseñanza como resultado de su semejanza con el idioma ingles. Estuvo disponible en 1.965. Existen diversas versiones disponibles de BASIC, el dialecto conocido como Microsof BASIC ha sido casi aceptado como un estándar para microordenadores.

El BASIC provee muy pocas estructuras para facilitar al programador la construcción de programas. Esta es, la razón por la cual el BASIC es tan fácil de aprender (otros lenguajes tienden a facilitar repertorios más potentes). Es así mismo un factor determinante característico de los programas en BASIC; tienen que construirse utilizando el mismo número, corto además, de bloques.

El BASIC posee un abanico de funciones; incluye funciones numéricas ampliamente comparables a las que tiene una calculadora científica y funciones para el manejo de caracteres.

El Basic ofrece un reducido repertorio de estructuras de programación a pesar de que al igual que todos los lenguajes de programación, ofrece al usuario la posibilidad de construir otras: Es factible describir cualquier calculo en BASIC, pero para escribir programas de cierta envergadura, tiene definitivamente una serie de restricciones, como consecuencia de la carencia de unas buenas estructuras de programación.

Visual Basic.

Versión de BASIC de Microsoft utilizado para desarrollar aplicaciones de Windows, que se ha vuelto popular. Es similar a QuickBASIC de Microsoft, pero no es 100% compatible con éste. Las interfaces de usuario se desarrollan llevando objetos de la caja de herramientas de Visual Basic hacia el formato de aplicación.

Visual Basic Script.

Es básicamente un lenguaje de *Sript*, que son aquellos lenguajes que se ejecutan sin que sea necesario compilarlos, como apoyo a otros lenguajes o aplicaciones mayores, y siempre dentro de una aplicación cliente. El VBScript es un lenguaje Script ya que cumple las siguientes condiciones.

Se ejecuta como apoyo a otro lenguaje, el HTML. No necesita compilación. Únicamente se ejecuta dentro de un programa mayor, en este caso el navegador Microsoft Internet Explorer

PASCAL.

Fue diseñado por el profesor Hiklaus Wirth del Instituto Federal de Tecnología de Zurich en 1970. Le puso el nombre de un matemático francés del siglo XVII, Blaise Pascal, a quien se debe, entre otros descubrimientos, la primera máquina de calcular.
El lenguaje fue implantado por primera vez por su diseñador y posteriormente estuvo disponible para todos los microordenadores populares. La razón principal que impulsó a Wirth a desarrollar PASCAL fue el ofrecer un lenguaje para enseñanza de la programación como disciplina sistematizada, de forma que los principios de la disciplina estuvieran claramente reflejados por el lenguaje. Se basa en un lenguaje anterior de programación, el ALGOL 60, conservando todas las características deseables de este lenguaje, con las aplicaciones y correcciones necesarias.

El hecho de que el PASCAL fuera adoptado ampliamente no solo para la enseñanza de la programación sino también para el desarrollo de sistemas de microprocesadores y por usuarios de microordenadores es un indicativo de su éxito y también del éxito de su diseñador al conseguir su principal propósito.

También pretendía que una eficiente y fiable implantación del lenguaje, pudiera realizarse con cualquier ordenador. En esta área el Pascal no tuvo tanto éxito. Muchas de sus implantaciones distan mucho de ser compactas necesitando más memoria que sus implantaciones en BASIC. El PASCAL provee un amplio repertorio de estructuras de programación y permite definir tipos de datos según se requieran.

Por consiguiente, al programador se le facilitan todas las características necesarias para dar a sus programas una estructura lógica y se le proporciona la posibilidad de diseñar sus propias estructuras de datos en caso de que las suplidas por el pascal no satisfagan sus necesidades. De esta manera no se hace necesario recurrir a métodos artificiales para diseñar programas y manejar datos.

Las normas impuestas por el Pascal, tales, como el requerir del programador cada variable y decir como se utilizará, deben ser consideradas como un beneficio ya que al permitir el desarrollo de programas en forma sistematizada se evitan automáticamente la mayor parte de los errores más comunes en programación.

Modula−2

A finales de los años 70, Nicklaus Wirth, creador del lenguaje PASCAL, dirige el desarrollo del MODULA−2 (que en principio se denomino simplemente MODULA), con la intención de incluir las necesidades de la programación de sistemas y dar respuesta a las criticas recibidas con respecto a las carencias del lenguaje PASCAL

Además de incluir las características del PASCAL, el nuevo lenguaje supera las principales carencias del mismo, como son la posibilidad de compilación separada, creación de bibliotecas, programación concurrente, mejora del manejo de cadenas de caracteres, procedimiento de entrada salida y de gestión de la memoria, etc. Además posee grandes facilidades para la programación de sistemas.

Este lenguaje posee cualidades didácticas, por lo cual ha sido ampliamente adoptado en la comunidad universitaria como herramienta idónea para la enseñanza de la programación

COMAL.

(COMmon Algorithmic Languaje) Existe debido a la disconformidad del educador y pedagogo danés Borge Christensen con el BASIC de Microsoft y todos los BASIC semejantes disponibles para microordenadores.

Christensen llegó a estar convencido de que el BASIC no era especialmente apropiado para enseñar una buena práctica de programación en ambiente escolar.

Al utilizar los microordenadores en la escuela, él quería conservar la simplicidad del BASIC que le caracteriza como fácil de aprender pero consideraba firmemente que el BASIC no era un vehículo satisfactorio para escribir programas bien estructurados. Como resultado desarrollo un lenguaje que pudiera satisfacer sus propios requerimientos.

Lo realizo observando las necesidades y dificultades de sus propios alumnos e introduciendo las posibilidades que él pensó podrían completar sus necesidades y ayudar a solucionar sus problemas. Encontró respuesta en sus alumnos a la innovación, observando sus puntos de vista y aceptándolos como resultado.

Las nuevas peculiaridades de COMAL se encuentran próximas a las estructuras de control del Pascal, de forma que el COMAL puede ser considerado como un híbrido del BASIC y del Pascal el cual posee muchas de las mejores propiedades de ambos lenguajes. Este híbrido puede y debería beneficiar no solo a los estudiantes que intentan aprender un lenguaje por primera vez, sino a cualquier programador que busca un lenguaje que sea razonablemente simple al tiempo que permita la producción de programas bien estructurados.

Cristhensen ha puntualizado que los programas que utilizan comal ventajosamente, pueden ser mas desarrollados mas rápidamente que con BASIC y pueden mantenerse con mas facilidad. En un ambiente educativo, las ventajas especificas del comal como híbrido del BASIC y Pascal son: es fácil de aprender de forma que el estudiante pueda programar desde la nada muy rápidamente y supone un puente intermedio hacia el Pascal que es el lenguaje de ordenador utilizado en la mayoría de los cursos de computadora de la Universidades y escuelas politécnicas.

COMAL está ampliamente extendido para microordenadores en muchos países incluido Dinamarca y Alemania. Los Ministerios de Educación de Dinamarca e Irlanda han adoptado el Comal como lenguaje universal para utilización en sus escuelas secundarias.

APL.

El APL fue diseñado por Kenneth Iverson que lo describió en su libro Un Lenguaje de Programación (A Programming Language) publicado por Wiley en 1962. El titulo del libro dio nombre al lenguaje. La motivación de original de Iverson para inventar este lenguaje fue, no tanto el facilitar un lenguaje de programación, como el inventar una representación en la cual pudieran expresarse con precisión los algoritmos y también que se pudiera describir exactamente el comportamiento del hardware.

APL se ha venido utilizando con éxito para describir el hardware del ordenador de manera formal, y para describir las semánticas o significados de un lenguaje de programación al facilitar una forma de expresión en la cual los efectos de sus instrucciones pueden darse exactamente. Debido a su utilización en aplicaciones como las expuestas, se ha argumentado que el APL es un sistema de expresión más que un lenguaje de programación: se ha implantado en muchas máquinas y se ha encontrado eco en los programadores para muchas y variadas aplicaciones.

Fue implantado en principio por IBM en una versión conocida como APL/360 en la mitad de los años 60 y estuvo disponible en general al final de la década. Le sucedieron otras implantaciones, incluidas algunas para microordenadores.
El APL como lenguaje de programación está concebido para describir procedimientos relativos al proceso de la información. El manejo de arrays tales como vectores hileras y matrices, es completo, ya que todos se pueden tratar como elementos singulares. Esta posibilidad condujo a la elección del lenguaje para planificación de negocios, ayudas a dirección empresarial y diseño de ingeniería, por ejemplo.

Adicional mente el APL se diseño para ser interactivo, en el sentido entendido por un programador que desarrolla funciones de comprobación (test) y modificación de programas ante un teclado. El usuario es animado a intentar realizar sus propias ideas y los errores se tratan de manera cordial y provechosa. De este modo, APL anima a la programación de los supuestos de investigación tipo ¿Qué pasa sí?(WHAT IF?) Que pueden ayudar a un directivo planificador o diseñador a tomar una decisión.

EL hecho de que APL puede manejar arrays como elementos singulares contribuye a dar otra característica al lenguaje, que es que los programas escritos en APL tienden a ser breves. Incluso para cálculos complejos los programas pueden ser cortos. Esta brevedad se puede considerar como una ventaja y como un inconveniente. Generalmente el relativamente sencillo el determinar la estructura de un programa corto y así mismo lleva menos tiempo de desarrollo. El principal inconveniente es que los progrMAS concisos son difíciles de comprender y por tanto de modificar.

La potencia de APL se demuestra mas fácilmente en la potencia para los programas concisos pero inclusos en estos programas se tiene la impresión de que APL es un lenguaje difícil

LOGO.

Creado por Seymour Papert, padre de la "computación educativa", el LOGO está destinado a la enseñanza de la programación a los niños, desde temprana edad. Por ello es sobretodo conocido por su capacidad gráfica y su "tortuga", que es el puntero con el cual se realizan los dibujos. Es altamente modular y deja gran libertad al usuario para definir procedimientos desde muy simples hasta muy complejos, en forma jerárquica, permitiendo incluso el control de periféricos mecánicos (operación de pequeños robots). Aunque bastante poderoso (se han escrito procesadores de palabras en LOGO), prácticamente no es utilizado fuera de la escuela básica.

HYPERTALK

"HyperTalk" es el lenguaje desarrollado por Dan Winkler para Bill Atkinson, el creador del "HyperCard" para Apple−Macintosh. Está orientado a la creación de aplicaciones conforme al sistema de "hiperarchivos" (sistemas de fichas interrelacionadas donde se facilita el "navegar" de un archivo a otro).

HyperTalk es un buen ejemplo de lenguaje orientado a objetos. Este tipo de lenguaje combina la lógica declarativa con los algoritmos. Un programa ya no es una secuencia de instrucciones sino un conjunto de objetos agrupados en conjuntos, definidos mediante atributos y a los cuales pueden asociarse instrucciones. Así, en HyperCard, existen archivos ("stacks" o "pilas") que agrupan fichas ("cards"), y cada una de éstas contiene campos de datos y botones. Todos son "objetos" que −si bien mantienen entre sí una relación jerárquica− tienen asociados paquetes de instrucciones ("scripts") independientes unos de otros. Cada objeto pertenece a un conjunto (como fichas o botones) que tiene "atributos" propios comunes a todos sus miembros, y cada atributo tendrá un valor común o específico para cada caso. Para dar o buscar dicho valor intervienen "facetas" que son instrucciones (procedimientos) asociadas. En la actualidad esta en desuso salvo excepciones.

ADA

Es un lenguaje estructurado parecido al PASCAL, destinado a controlar mecanismos en "tiempo real" (o sea una velocidad compatible con las necesidades reales), pero de gran complejidad. Admite una programación "orientada a objetos" y un sistema de alta modularidad de tipo hipertexto.

Fue elaborado a pedido del Departamento de Defensa de los Estados Unidos y establecido como norma para todos los fabricantes que participaban en el programa de la Iniciativa de Defensa Estratégica (IDE, también llamado "Guerra de las Galaxias").

C.

El lenguaje fue creado en 1972 por Dennis Ritchie, que junto con Ken Thompson había diseñado anteriormente el sistema operativo UNIX, y su intención al desarrollar el lenguaje C fue conseguir un lenguaje idóneo para la programación de sistemas que fuese independiente de la máquina para utilizarlo en la implementación del sistema operativo UNIX. Desde entonces, tanto el UNIX como el C han tenido un enorme desarrollo y proliferación, hasta convertirse en un estándar industrial para el desarrollo de software

El C es un lenguaje moderno de propósito general que combina las características de un lenguaje de alto nivel (programación estructurada, tipos y estructura de datos, recursividad, etc.) con una serie de características más propias de lenguajes de más bajo nivel. Esta cualidad del C hace posible que el programador use la programación estructurada para resolver tareas de bajo nivel, obteniendo un código ejecutable veloz y eficiente. Debido a sus características de más bajo nivel, mucha gente considera al C como un lenguaje de nivel medio.

Debido a esta libertad de programación que proporciona este lenguaje, se ha vuelto muy popular y es el lenguaje más usado entre los desarrolladores profesionales de software de aplicaciones comerciales (procesamiento de textos, bases de datos, aplicaciones cientifico−técnicas, etc.). Además C, es un lenguaje pequeño (posee pocas instrucciones) y conciso (no presenta instrucciones redundantes). El coste de un lenguaje tan potente y útil es que no es particularmente fácil de aprender. De hecho, la programación segura y fiable en este lenguaje requiere un conocimiento bastante profundo del mismo

C++.

El C++, el sucesor del lenguaje C, fue desarrollado por Bjarn Stroustup en los laboratorios Bell a principio de la década de los ochenta. En el lenguaje C, C++ es una orden que equivale a C: =C+1, por lo que se entiende que con C++ el lenguaje C se eleva hacia su siguiente nivel.

C++ introduce la programación orientada a objetos en C. Los objetos proporcionan una forma completamente nueva de ver los programas, una nueva filosofía de programación.

Al igual que C, C++ es un lenguaje muy poderoso y eficiente. Sin embargo C++ es aún más difícil de aprender que C. Dado que C es un subconjunto de C++, aprender C++ significa aprender todo acerca de C y después aprender la filosofía de la programación orientada a objetos y el uso que hace C++ de la misma.

Visual C++

Sistema de desarrollo C y C++ para aplicaciones DOS y Windows, de Microsoft. Introducido en 1993, el Standard Edition de Visual C++ reemplaza a QuickC para Windows, y el Professional Edition incluye el Windows SDK y reemplaza Microsoft C/C++ 7.0.

LISP.

El objeto del Lisp es el Proceso de listas (LISt Processing). El proceso de listas quizás no parezca una actividad tan común como para justificar un lenguaje especial, pero el hecho es que una lista es una estructura de datos generales muy particular y con su ayuda pueden ser emprendidos problemas de muchos tipos de manera asequible.

Como List trata listas de cualquier tipo de elementos, permite descubrir y ejecutar cálculos no numéricos y proporcionar en particular una herramienta para el manejo de símbolos. Lisp fue desarrollado por el profesor John McCarthy y sus alumnos en el Instituto de Tecnología de Massachusetts en 1.959.

Su propósito original fue desarrollar un sistema de programación llamado el Registrador de Avisos que seria capaz de manejar hechos y comandos, utilizando los hechos según el sentido común para ayudar a interpretar y llevar a cabo los comandos. Los trabajadores en otras áreas, particularmente los relacionados en trabajos de Inteligencia Artificial se dieron pronto cuenta de que el lenguaje de McCarthy aportaban los medios de manipular los símbolos que estaban buscando. El manejo de símbolos es un requisito común a muchas áreas de investigación que son parte de inteligencia artificial (AI), incluyendo la resolución de problemas de tipo general, reconocimiento de patrones, prueba de teoremas y manejo de cálculos lingüísticos y algebraicos. Como consecuencia el Lisp ha llegado ha ser el lenguaje mas utilizado de AI.

PROLOG.

Se origino en un departamento universitario de AI y su uso mas allá de sus primeros años, se extendió en los confines de departamentos semejantes. Fue originalmente desarrollado en la Universidad de Aix−Marseilles en Francia. Desde 1972 ha habido implantaciones del lenguaje allí y en otros lugares, incluido el departamento de AI de la universidad de Edimburgo y el departamento de Cáculo y Control del Imperial College de Londres.

PROLOG (PROgramacion con LOGica) es un lenguaje de ordenador, sencillo pero potente, desarrollado inicialmente para la ayuda en la comprobación automática de teoremas. La utilización de una lógica formal para procesos de razonamiento del modelo humano es algo nuevo, pero si los ordenadores se utilizan en su investigación, entonces un lenguaje apropiado ayuda considerablemente. PROLOG puede utilizarse con buenos resultados en muchas áreas además de la prueba automática de teoremas. Puede utilizarse como lenguaje de consulta de base de datos o para la automatización de razonamientos deductivos o como lenguaje para representar información para el proceso de lenguaje natural

Actualmente, PROLOG esta disponible en un ámbito más amplio. Por ejemplo, ha sido implantado para una rama de ordenadores DEC y también hay versiones para microordenadores. Esta amplia disponibilidad junto a la programación del ámbito de usuarios ha permitido aplicar el lenguaje en otras muchas aplicaciones, y no solo para las que fue concebido. Muchos proyectos educativos en los que se incluye la utilización de PRTOLOG como herramienta para enseñanza de lógica para los niños, están entre las nuevas aplicaciones.

FORTH.

Fue diseñado por el astrónomo americano Charles MOORE como lenguaje para escribir programas para controlar radiotelescopios y otros equipos de astronomía. A pesar de que fue originariamente desarrollado para aplicaciones de control, ha sido adoptado por un numero cada vez mayor de entusiastas del hobby ya que es rápido y por que es un lenguaje extensible al cual se le pueden añadir fácilmente características que no posea ya, de forma que constituyan parte efectiva de él.

Al disponer de este tipo de flexibilidad FORTH puede ser construido fácilmente para cualquier aplicación el FORTH también es llamado Lenguaje enhebrado, que significa que las características proporcionadas mantiene como una lista encadenada de elementos.

En esta lista el nombre de cada elemento se almacena como una rutina en código máquina para proveer esa facilidad y como resultado, los programas FORTH pueden ejecutarse casi tan rápidamente como los programas escritos en código máquina. Se puede añadir cualquier característica nueva muy simplemente, a la lista encadenada llegando por tanto a constituir una parte del lenguaje indistinguible de la parte original. Cuando se define una característica basada en otras existentes solamente se necesitan almacenar su nombre con punteros para la rutina relevante en código máquina a fin de proveer el código máquina para la nueva característica

Perl

Es un lenguaje especializado en el procesamiento de textos, particularmente extraer y validar las respuestas a cuestionarios incluidos en páginas Web.

Clipper.

CLIPPER es un dialecto creado como otros tantos con la intención de mejorar las prestaciones de DBASE. Su primera versión se creó en 1985 en los laboratorios de Natuncket. CLIPPER está escrito en lenguaje C y Ensamblador y se presentó como un lenguaje atrevido que ha dado muchos quebraderos de cabeza en Ashthon−Tate.

En el primer contacto que se tiene con él es difícil encontrar muchas diferencias con respecto a DBASE, ya que CLIPPER es un lenguaje formado por un conjunto de comandos y funciones similares a las usadas con DBASE, incluso la mayoría con igual formato sintáctico.

La principal diferencia, está en que todos los programas escritos en Clipper pueden compilarse y enlazarse. El resultado obtenido es un fichero ejecutable que puede utilizarse de forma independiente al gestor de base de datos y sin necesidad de incluir módulo runtime. Esto repercute en la velocidad de ejecución de los programas.

CLIPPER es ahora sin duda el compilador más utilizado en aplicaciones de gestión de datos para microordenadores. La última versión aparecida en el mercado es la CLIPPER 5.01 versión reparada de la CLIPPER 5.0. Hasta el momento, la versión más utilizada quizás por su largo tiempo de vigencia es la CLIPPER SUMMER '87. Anteriores a ésta eran la CLIPPER AUTUMN '86 y la versión de 1985.

De todas las versiones detalladas la SUMMER '87 ha sido la más difundida. Muchas aplicaciones se han desarrollado con esta versión, por ello, aún, muchos programadores se resisten al cambio a versiones más actuales.

Otras prestaciones de CLIPPER SUMMER '87 a destacar son las siguientes:

Provee un conjunto de funciones para el tratamiento de ficheros en redes de área local. Permite manejar ficheros de bajo nivel. Posibilita la creación de funciones de usuarios y agruparlas en librerías. Permite el uso de arrays unidimensionales. Proporciona un depurador avanzado.

Delphi.

Permite crear aplicaciones Windows con un esfuerzo mínimo, sin apenas conocimiento del funcionamiento interno de Windows.
Permite crear aplicaciones simplemente añadiendo iconos que representan objetos, modificando propiedades, que son las características de esos objetos, y escribiendo algo de código.

El resultado es que una aplicación cuyo desarrollo en un lenguaje como C puede tener una complejidad importante, utilizando una de estas herramientas de desarrollo visual resulta muy simple.

La primera versión de Dephi apareció en el mercado en el año 1.994. Basado en un compilador de indudable calidad, el de Borland Pascal, Dephi es capaz de generar aplicaciones de menor tamaño y mucho más rápidas que las que sean desarrollar con otros productos similares.

La aparición de Delphi 2.0 incorporó muchas novedades al entorno, como la posibilidad de generar código de 32 bits, para Windows 95 y NT, nuevos componentes y herramientas para trabajo con bases de datos y unas posibilidades de conectividad importantes. A todo esto Delphi 3.0 añadió nuevas posibilidades, como la creación de controles ActiveX, servidores de Internet, etc. Después apareció Delphi 4.0, con novedades en el lenguaje y nuevos componentes que simplificaban la creación de interfaces de usuario, así como el desarrollo de aplicaciones distribuidas.

En el 99 aparece el Dephi 5.0 para windows 98 y NT/2000. Ofrece un entorno en el que la escritura de código es más fácil que nunca, contando con todas las características para crear aplicaciones con avanzadas interfaces de usuario, servicios locales y distribuidos y acceso de todo tipo de orígenes de datos.

HTML.

Está formado por un conjunto de identificadores, designados con el término ingles *tag*, que definen el formato de una página de texto, permitiendo insertar en ella elementos multimedia, tales como imágenes, sonido y vídeo. Por lo tanto, la función del navegador de Internet es la de traducir este código un contenido gráfico

El HTML 4.0 es una aplicación SGML (Lenguaje de Etiqueta Generalizado Estándar) conforme al estándar internacional ISO 8879 y está ampliamente considerado como el lenguaje de publicación estándar del World Wide Web.

HTML, tal como fue concebido, era un lenguaje para el intercambio de documentos científicos y técnicos adaptado para ser usado por no especialistas en el tratamiento de documentos.
HTML resolvió el problema de la complejidad del SGML sirviéndose de un reducido conjunto de etiquetas estructurales y semánticas apropiadas para la realización de documentos relativamente simples. Además de simplificar la estructura de documentos, HTML soportaba el hipertexto. Las posibilidades de usar elementos multimedia fueron añadidas con posterioridad.

En un corto período de tiempo, HTML se hizo muy popular y rápidamente superó los propósitos para los que había sido creado. Desde los albores del HTML, ha habido una constante invención de nuevos elementos para ser usados dentro de HTML (como estándar) y para adaptar HTML a mercados verticales, altamente especializados. Esta cantidad de nuevos elementos ha llevado a problemas de compatibilidad de los documentos en las distintas plataformas.

XHTML.

La especificación XHTML 1.0 (recomendación del 26 de enero del 2000) es una reformulación del HTML como aplicación XML, exactamente es la reformulación de las tres definiciones de tipo de documento HTML 4.0 como aplicaciones XML. Su finalidad es ser usado como lenguaje de contenidos que es a su vez conforme a XML y, si se siguen algunas sencillas directrices, funciona en agentes de usuario conformes con HTML 4.0.

PHP

Lenguaje que se acopla al HTML (páginas Web) para definir procedimientos que ha de realizar el servidor de web, por ejemplo procesar un formulario, enviar o extraer datos de una base de datos (acoplándose también con un lenguaje de tipo SQL), enviar una u otra página Web según determinadas condiciones prefijadas por el programador, etc.

SQL

Lenguaje desarrollado especialmente para facilitar la consulta de bases de datos (BD), acotando progresivamente la búsqueda (de ahí el nombre de "Sequential Query Language"). Existen hoy numerosas aplicaciones de administración de bases de datos que recurren al SQL (Las más conocidas, potentes − y caras − son Oracle e Informix).

Hoy se pueden acoplar las bases de datos a hipertextos (páginas Web), para lo cual las buenas aplicaciones ya traen módulos que hacen la conexión. El lenguaje PHP del cual hablamos más arriba también sirve para definir procedimientos de inserción y de consulta de datos en BD que funcionan con SQL.

PL/1.

EL "PL/1" es un lenguaje multi−propósito creado por IBM y SHARE, especialmente a raíz del paso de la segunda a la tercera generación de computadoras, cuando se preveía la creciente difusión de estas máquinas y su posible uso en una gama creciente de actividades. Pretendía ampliar las posibilidades del FORTRAN fusionando conceptos provenientes del COBOL y el ALGOL.

La gran cantidad de instrucciones, tipos de datos y casos especiales que contempla lo hacen difícil de aprender y dominar, razón de su poca difusión. En la actualidad esta en desuso salvo excepciones

Java.

Java nació para intentar encontrar la solución a un problema. Este problema radicaba en las dificultades y costes que suponía la actualización muy frecuente del software de microprocesadores de reducidas prestaciones que se montan en dispositivos electrónicos de bajo precio, como electrodomésticos, relojes y calculadoras.

Esto suponía la obligatoriedad de modificar el código para cada microprocesador, aun cuando fuera escrito en un lenguaje de alto nivel con C++, debido a las particularidades de cada microprocesador en cuestión.

Los primeros en plantearse este problema fueron los desarrolladores de la empresa Sun Microsystem, encabezados por James Gosling, los cuales principios de los años 90 junto con su equipo, se marcan el objetivo de desarrollar un nuevo lenguaje de programación capaz de adecuarse a cualquier entorno de ejecución (portable) y que se basara en la simplicidad.

Para ello, decidieron eliminar todas aquellas instrucciones y funciones (que no eran imprescindibles en un lenguaje moderno, como el C++) culpables de numerosos errores habituales, pero manteniendo las características de un lenguaje de alto nivel.

Y es así como nació Java. Su lanzamiento y presentación mundial se llevo a cabo en el verano de 1.995.

Con el auge de Internet, el grupo de Goslling, se plantea la posibilidad de demostrar la afirmación de que su lenguaje podía adaptarse a cualquier entorno de ejecución, incluso que los programas escritos en Java podían ejecutarse desde cualquier punto de la red, como si se tratase de un elemento mas de la Web

Para demostrar esto, se tuvo que diseñar un navegador que integrara Java y que permitiese la ejecución de Java tal y como se había afirmado.

Así nació la primera versión de HotJava. Este hecho fue determinante en la carrera de éxitos que ha cosechado Java, y sobretodo en la decisión de Sun Microsystem de ofrecer de forma gratuita y abierta sus herramientas de desarrollo para Java

Java Script.

Es un lenguaje de Script de funcionalidad idéntica a la del VBScript y se puede decir que es su máximo y principal competidor. Su sintaxis es parecida a la del Java y C++ aunque esta bastante mas limitado que estos lenguajes

1.4 DEFINICIÓN DE LENGUAJE DE PROGRAMACIÓN

Con la aparición de las computadoras desaparecen las secuencias de posiciones de llaves mecánicas que debían desconectarse para obtener una acción determinada, una llave conectada era un 1 y una llave desconectada era un 0. Una sucesión de llaves en cualquiera de sus dos posiciones definía una secuencia de ceros y unos (por ejemplo: 0100011010011101...) que venía a representar una instrucción o un conjunto de instrucciones (programa) para el ordenador (o computador) en el que se estaba trabajando. A esta primera forma de especificar programas para una computadora se la denomina lenguaje máquina o código máquina.

La necesidad de recordar secuencias de programación para las acciones usuales llevó a denominarlas con nombres fáciles de memorizar y asociar: ADD (sumar), SUB (restar), MUL (multiplicar), CALL (ejecutar subrutina), etc. A esta secuencia de posiciones se le denominó "instrucciones", y a este conjunto de instrucciones se le llamó lenguaje ensamblador.

Posteriormente aparecieron diferentes lenguajes de programación, los cuales reciben su denominación porque tienen una estructura sintáctica similar a los lenguajes escritos por los humanos.

Concepto

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.

Aunque muchas veces se usan los términos 'lenguaje de programación' y 'lenguaje informático' como si fuesen sinónimos, no tiene por qué ser así, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como, por ejemplo, el HTML (lenguaje para el marcado de páginas web que no es propiamente un lenguaje de programación).

Un lenguaje de programación permite a uno o más programadores especificar de *manera precisa* sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar *relativamente* próximo al lenguaje humano o natural, tal como sucede con el lenguaje Léxico.

Una característica relevante de los lenguajes de programación es precisamente que más de un programador puedan tener un conjunto común de instrucciones que puedan ser comprendidas entre ellos para realizar la construcción del programa de forma colaborativa.

Los procesadores usados en las computadoras son capaces de entender y actuar según lo indican programas escritos en un lenguaje fijo llamado lenguaje de máquina. Todo programa escrito en otro lenguaje puede ser ejecutado de dos maneras:

- Mediante un programa que va adaptando las instrucciones conforme son encontradas. A este proceso se le llama *interpretar* y a los programas que lo hacen se los conoce como intérpretes.

- Traduciendo este programa, al programa equivalente escrito en lenguaje de máquina. A ese proceso se le llama *compilar* y al programa traductor se le denomina compilador.