INTRODUCCIÓN.[1]

La informática es la ciencia que estudia el procesamiento automático de la información. La piedra maestra sobre la cual se ha podido desarrollar la informática la representa el computador, que es una herramienta de gran eficacia en muy diversos trabajos, y en particular en aquellos que manejan un gran volumen de datos o de operaciones. Esta versatilidad tiene dos aspectos: por un lado, es posible usarlo como herramienta para aplicaciones concretas como los procesadores de textos, las hojas de calculo, los gestores de bases de datos, etc y por otro se pueden diseñar soluciones a la medida de problemas nuevos, mediante la programación.

El desarrollo de un programa nuevo para resolver un determinado problema requiere, por una parte, conocer algún procedimiento sistemático (algoritmo) que lleve a su solución, y por otra, la necesidad de expresarlo en un lenguaje de programación que el computador pueda comprender y ejecutar.

1. CONCEPTO DE LENGUAJE DE PROGRAMACIÓN.[1,2]

Un lenguaje de programación es un lenguaje artificial, diseñado para representar expresiones e instrucciones de forma inteligible para las computadoras.

Los lenguajes de programación son, en gran medida, comparables a los lenguajes naturales: sus símbolos básicos constituyen su alfabeto, y con ellos se construye el vocabulario del lenguaje, cuyos elementos se llaman tokens. Estos tokens se combinan de acuerdo con las reglas sintácticas del lenguaje, formando expresiones y sentencias cuyo significado viene dado por la semántica del lenguaje.

Los lenguajes de programación son sin embargo considerablemente más simples que los naturales en su sintaxis y, especialmente, en su semántica.

En definitiva un lenguaje de programación es un conjunto predefinido de palabras y símbolos que se utilizan siguiendo unas reglas prefijadas (sintaxis) para expresar algoritmos

2. EVOLUCIÓN DE LOS LENGUAJES DE PROGRAMACIÓN.[3,4]

Los primeros lenguajes de programación surgieron de la idea de Charles Babagge a mediados del siglo XIX, fue profesor matemático de la Universidad de Cambridge. Babagge predijo muchas de las teorias en las cuales se basan los ordenadores actuales, junto a el colaboró Ada Lovedby, la cual es considerada como la primera programadora de la historia. Los programas de Ada nunca llegaron a ejecutarse, pero si suponian un punto de partida de la programación.

Fue en el año de 1945 cuando el matemático Jhon Von Neumann elaboró un estudio en el que demostró que una computadora podía tener una estructura física muy simple y fija, y aun así ser capaz de ejecutar cualquier tipo de programa mediante un control correcto, sin la necesidad de modificar la computadora para esto. A partir de esta innovación de Von Neunmann, que en conjunto se conoce como la técnica de programa almacenado es cuando se inicia la era de los lenguajes de programación.

En 1949 aparece el primer lenguaje que se usó en computadoras electrónicas (Shortcode) requería que el programador compilara su programa a 0's y 1's de manera manual. No fue hasta el año 1951, cuando Grace Hopper trabajando para Remington Rand, comienza a desarrollar el primer compilador, lo que trajo consigo una programación más rápida.

Es en el año 1957 cuando aparece el primero de los grandes lenguajes de programación de aún uso actual, Fortran (Formula Translating System), fue desarrollado por IBM para cálculos científicos, el líder del proyecto fue John Backus, que después contribuiría en Algol.
El lenguaje original era bueno manejando números, pero malo manejando entrada y salida, lo cual propicio la aparición de otros lenguajes orientados a negocios.

En el año de 1958, el profesor John McCarthy del M.I.T. comenzó a desarrollar la teoría de un lenguaje de procesamiento de listas. En 1959 aparece publicamente la primera implementación llamada Lisp 1.5.

Es importante resaltar que McCarthy no solo marco un hito en la historia de los lenguajes de programación sino que creo un modelo de programación que ha demostrado ser superior, tanto que podemos decir que en la actualidad existen dos grandes modelos de programación el de C y el de Lisp.

Es en 1960 cuando aparece el lenguaje Algol 6.0 el primer lenguaje estructurado en bloques. Este lenguaje fue muy popular en el segundo lustro de los 60's, su principal contribución es ser la raiz del arbol que ha producido lenguajes tales como Pascal, C, C++, y Java.

En 1959, Conference on Data Systems and Languages (CODASYL) crea Cobol, un lenguaje para negocios que fuera facil de aprender para gente que no tuviera formación en ciencias de la informática. Las sentencias de Cobol se parecen mucho a las usadas por el idioma inglés, haciendo que fuera fácil de aprender.

Basados en los primeros lenguajes de programación han surgido muchos otros lenguajes que siempre tienen la intención de tomar lo mejor, desechar lo malo, y agregar alguna novedad, respecto a los existentes.

- *BASIC. (**B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode)*

Fue creado en 1964 por John George Kemeny y Thomas Eugene Kurtz en el Colegio Dartmounth.
Es un lenguaje muy limitado que fue diseñado para personas que no fueran del área de ciencias de la computación.
Los ocho principios de diseño de BASIC fueron:

1. Ser fácil de usar para los principiantes.
2. Ser un lenguaje de propósito general.
3. Permitir que los expertos añadieran características avanzadas, mientras que el lenguaje permanecía simple para los principiantes.
4. Ser interactivo.
5. Proveer mensajes de error claros y amigables.
6. Responder rápido a los programas pequeños.
7. No requerir un conocimiento del hardware de la computadora.
8. Proteger al usuario del sistema operativo.

El lenguaje se baso en Fortran y Algol 6.0

- *PASCAL.*

Fue diseñado por Niklaus Wirth como una herramienta de enseñanza de la programación. Sus desarrolladores se concentraron en desarrollar buenas herramientas que contribuyeran a la enseñanza, tal como un buen debugger, y un buen editor. Además tuvieron como meta el tener soporte para la mayoría de los microprocesadores populares en esa época en las instituciones de enseñanza.

Fue diseñado de una manera muy ordenada, reflejando la experiencia de su diseñador, tomo las mejores características de los lenguajes de su tiempo, COBOL, ALGOL, y FORTRAN, al mismo tiempo que busco evitar sus deficiencias, y hacerlo lo mas claro posible. La combinación de sus características de entrada/salida, y sus solidas características matemáticas pronto lo convirtieron en un lenguaje muy exitoso. También implemento el tipo apuntador y agrego el CASE, e hizo uso de variables dinámicas.

Sin embargo no implemento arreglos dinámicos ni agrupamiento de variables lo cual contribuyo a su perdida de popularidad frente a nuevos lenguajes.
Delphi es una versión moderna y orientada a objetos de Pascal.

- *C.*

Fue diseñado en 1971, por Dennis Ritchie y Ken Thompson mientras trabajaban para los Laboratorios Bell, y se basó en los lenguajes de programación B y BCPL.

Se basa en el paradigma imperativo y desde su creación estuvo pensado para programación de sistemas operativos, se creo para usarse en UNIX, y creció de la mano del desarrollo de UNIX, lo que propició la creación de características avanzadas tales como variables dinámicas, multitarea, manejo de interrupciones, forking y un poderoso manejo entrada/salida de bajo nivel. Debido a esto C es comunmente usado para programación de nivel de sistema en UNIX, Linux y Mac.

Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Destaca su gran riqueza de operadores y expresiones.

- *C++.*

A finales de los 70's y principio de los 80's un nuevo modelo de de programación fue desarrollado, la programación orientada a objetos, la idea básica es que los objetos son piezas de código autocontenidas y reusables. Bjarne Stroustroup también de los Laboratorios Bell, desarrolló un nuevo lenguaje basado en C que aplica los conceptos de la programación orientada a objetos, inicialmente se llamo C con clases, para posteriormente tomar su nombre definitivo C++ cuando fue publicado en 1983.

Las principales características del C++ son abstracción, el soporte para programación orientada a objetos y el soporte de plantillas o programación genérica. Por lo cual , se puede decir que C++ es un lenguaje multi-paradigma que abarca tres paradigmas de la programación: La programación estructurada, la programación genérica y la programación orientada a objetos.
Actualmente cuenta con un estándar ISO y es muy popular en la programación de aplicaciones.

- *JAVA.*

Fue desarrollado por James Gosling y sus equipo en Sun Microsystems,entre 1990 y 1994, pensado originalmente como un reemplazo de C++, orientado a dispositivos embebidos, y a la televisión interactiva, posteriormente fue rescatarlo del fracaso y reorientado hacia su aplicación en la Web.
Las cinco metas del lenguaje Java son:

- Orientado a objetos
- Multiplataforma
- Soporte integrado para redes de computadoras
- Diseñado para ejecutar codigo de fuentes remotas de modo seguro
- Facil de usar

- *JAVA SCRIPT.*

Es un lenguaje de script basado en objetos, que se apoya en el modelo de prototipos. Es muy popular por su uso en sitios Web, fue desarrollado en 1995,

- *PHP.*

Creado por Rasmus Lerdorf en 1994,. Inicialmente significaba “Personal Home Page Tools”, y fue publicado por primera vez el 8 de junio de 1995, actualmente su nombre oficial es: “PHP Hypertext Preprocessor”.
Las caracteristicas mas destacadas de PHP son su facilidad de aprendizaje y que es software libre.

- *C#.*

Es un lenguaje orientado a objetos desarrollado por Microsoft, tomando ideas de C++ y Java, como parte de su estrategia comercial .Net. En el año 2003 se convirtio en un estandar ISO “(ISO/IEC 23270)”.

Gracias a la maquinaría comercial de Microsoft y la amplia base de escritorios Windows, el uso de C# es muy extendido en entornos corporativos y en el mundo Windows en general.

Los proyectos libres de C#, no son muy populares entre la comunidad del software libre, por la desconfianza hacia Microsoft y sus patentes.

3. CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN.[2]

El procesador de un computador únicamente puede interpretar y ejecutar programas escritos en lenguaje código máquina. Este lenguaje depende del procesador de que se trate y esta muy alejado de los distintos lenguajes utilizados en los distintos campos de aplicación de la informática. Se han desarrollado lenguajes de alto nivel que son independientes del computador y que permiten expresar o representar algoritmos de una forma de que por un lado es facilmente compresible para las personas y por otro lado es traducible automaticamente a código máquina.

Para ejecutar un algoritmo en un computador es necesario representarlo en forma de programa, para escribir un programa hay que utilizar un lenguaje de programación, que es un conjunto predefinido de palabras y simbolos que se utilizan siguiendo unas reglas prefijadas (sintaxis) para expresar algoritmos. Programar consiste por lo tanto en establecer órdenes para un computador utilizando un lenguaje de programación.

Un computador solo es capaz de entender y ejecutar directamente programas escritos en su lenguaje máquina, una caracteristica de este lenguaje es que es muy engorroso de utilizar, necesitando conocer la arquitectura fisica del computador con cierto detalle, está completamente adaptado a los circuitos del procesador y muy alejado del lenguaje habitual frente a esto un programa escrito en lenguaje máquina es directamente interpretable por el procesador central, no es necesario transformaciones previas para ser ejecutado, ademas se ejecuta muy eficientemente (con rapidez), ya que el usuario lo redacta especificamente para los circuitos que lo han de interpretar, y desde este código se utilizan en su totalidad los recursos de la máquina.

Las principales características del lenguaje máquina son las siguientes:

- Las instrucciones son cadenas de ceros y unos. Cada instrucción contiene un campo o apartado donde se especifica el código de operación de la instrucción y otros campos que indican el lugar donde se encuentran los operandos.

- Los datos se utilizan por medio de las direcciones de registros o de memoria donde se encuentran. En las instrucciones no aparecen nombres de variables (x,y,z) sino que el programador debe hacer una asignación de registros y de direcciones de memoria para todas las variables y constantes del problema.

- Las instrucciones realizan operaciones relativamente simples.

- Existe muy poca versatilidad para la redacción de las instrucciones. Éstas tienen un formato rígido en cuanto a posición de los distintos campos que configuran la instrucción.

- El lenguaje máquina depende y está ligado intimamente al procesador del computador. Si dos computadores tienen procesadores distintos, tienen distintos lenguajes máquina.

- En un programa en código máquina no pueden incluirse comentarios que faciliten la legibilidad del mismo, además, debido a su representación totalmente númerica, es muy dificil de reconocer o interpretar por el usuario.

Dadas las limitaciones señaladas de los lenguajes máquina se han desarrollado lenguajes simbólicos, estos facilitan notablemente el trabajo de programación y hacen los programas más legibles. Se caracterizan porque en vez de ceros y unos se pueden utilizar nombres simbolicos para identificar las instrucciones y para denominar las variables y direcciones de memoria. Existen dos tipos de lenguajes simbólicos: lenguajes ensambladores y lenguajes de alto nivel.

3.1. LENGUAJES ENSAMBLADORES.

Para facilitar la programación ya en la decada de los años cicuenta se desarrollaron programas ensambladores, en comparación con un lenguaje máquina, este tipo de lenguajes permiten al programador:

- Escribir las instrucciones utilizando una notacion simbólica o nemotécnica, en vez de códigos binarios. Normalmento los códigos nemotécnicos estan constituidos por tres o cuatro letras que, en forma abreviada, indican la operación a realizar. Debido al origen anglosajón de los fabricantes de computadores, los nemotécnicos son abreviaturas en ingles.

- Utilizar direcciones simbólicas de memoria, en lugar de direcciones binarias. Existen sentencias declarativas para indicar al traductor la correspondencia entre direcciones simbólicas y direcciones de memoria, con estas sentencias el traductor crea una tabla con cuya ayuda, al generar las instrucciones máquina, se sustituyen las direcciones simbólicas por las direcciones binarias correspondientes.

- Insertar lineas de comentarios entre las lineas de instrucciones. El traductor las elimina automaticamente, no incluyendolas en el código máquina que genera.

El traductor de lenguaje ensamblador a lenguaje máquina se denomina traductor de ensamblador o sencillamente ensamblador, y mejora o resuelve los problemas citados anteriormente referidos al lenguaje máquina, persistiendo algunas limitaciones ya que este tipo de lenguajes hace corresponder a cada instrucción en ensamblador una instrucción en codigo máquina. Un programa en ensamblador no puede ejecutarse directamente por el computador, siendo necesario ser traducido previamente.

La mayoria de los ensambladores actuales son macroensambladores. Con ellos se solventa en cierta medida la limitación de tener un repertorio de instrucciones muy reducido. Un lenguaje macroensamblador dispone de macroinstrucciones, como por ejemplo transferir un bloque de datos de memoria principal a disco, multiplicar, dividir, etc. La macro instrucción es una llamada a un módulo o rutina, llamada macro, de una biblioteca.

3.2. LENGUAJES DE ALTO NIVEL.

Aunque los lenguajes ensambladores facilitan notablemente la tarea de programar, siguen presentando inconvenientes notables, como son la dependencia de la arquitectura del procesador, la poca versatilidad del programador para crear sus propias instrucciones y su semantica, que sigue muy alejada de la de los dominios de las aplicaciones.
Para facilitar aun mas el trabajo de programación se desarrollaron los lenguajes de alto nivel. El primer lenguaje de programación ampliamente distribuido fue FORTRAN (1954) diseñado por John Backus en IBM.

Los lenguajes de alto nivel se caracterizan por:

- Ser independientes de la arquitectura del computador. El programador no tiene porque conocer los detalles del procesador que utiliza, y por tanto los programas son transportables, ya que se pueden utilizar en computadores con distintos lenguajes máquina.

- Disponer de instrucciones potentes, coteniendo operadores y funciones de gran diversidad, aritmeticas (seno,coseno,módulo), especiales, logicas y de tratamiento de caracteres.

- Usar una sintexis parecida al lenguaje natural o al lenguaje del dominio de aplicación del programa planteado, esto quiere decir que se puede utilizar texto y asignar nombres simbólicos a determinados componentes del programa para facilitar su compresión.

4. PARADIGMAS DE PROGRAMACIÓN.[5,2]

Un paradigma de programación es una colección de modelos conceptuales que juntos modelan el proceso de diseño y determinan, al final, la estructura de un programa. Los paradigmas se agrupan en tres categorías de acuerdo con la solución que aportan para resolver el problema (sólo se comentarán los paradigmas relacionados con la programación de alto nivel).

4.1. PARADIGMA IMPERATIVO.

Describe paso a paso el modo de construir la solución. Es decir, señala la forma de obtener la solución. La principal caracteristica de estos paradigmas es que la secuencia computacional realiza etapa a etapa para resolver el problema. Su mayor complejidad reside en determinar si el valor computado es una solución correcta del problema, por lo que se han desarrollado multitud de técnicas de depuración y verificación para saber si es cierta la corrección de los problemas desarrollados basándose en estos paradigmas.

Pueden ser de dos tipos dependiendo de su flujo de control: estructurados y no estructurados. Los lenguajes imperativos no estructurados utilizan la instruccion GOTO como instruccion básica para establecer el flujo de control, mientras que los estructurados proporcionan sentencias estructuradas que hacen mas facil que el flujo de ejecuccion coincida con la secuencia de instrucciones del texto del programa

Dentro de los paradigmas de programacion se puede hacer otra clasificacion atendiendo a los mecanismos que utilizan para organizar los programas: procedurales, modulares y orientados a objetos. En los procedurales la unidad de descomposicion es el subprograma. En los modulares se proporciona el concepto de modulo, como un mecanismo de agrupacion de elementos de software. Y finalmente en los orientados a objetos, las clases juegan el papel de los módulos, a la vez que actuan como distintos tipos de datos.

4.2. PARADIGMA ORIENTADO A OBJETOS.

Es el paradigma mas apropiado para crear software de gran calidad. Este paradigma se caracteriza por los conceptos de clase, objeto, herencia y polimorfismo. Los conceptos de clase y objeto permiten reducir el salto semántico entre el dominio del problema y el programa, ya que los datos que forman un programa orientado a objetos aparecen como datos abstractos, cada uno de los cuales define un dominio de valores más un conjunto de posibles operaciones sobre ese dominio. Por otra parte, la herencia define jerarquias de clases que estan relacionadas por una relacion de generalizacion.

Cuando una clase hereda de otra, esta puede adquirir todas las propiedades de la primera. La herencia combinada con los conceptos de mensaje, polimorfismo y ligadura dinámica, es el principio para escribir un codigo reutilizable. Los lenguajes mas utilizados en este paradigma son C#, Java y Pascal.

4.3. *PARADIGMA DECLARATIVO.*

En este paradigma un programa se construye señalando hechos, restricciones, ecuaciones, transformaciones y otras propiedades derivadas del conjunto de valores que configuran la solución. A partir de la informacion que se da, el sistema debe proporcionar un esquema que incluya el orden de evaluación que compute una solución. Estos paradigmas permiten el uso de variables para almacenar valores intermedios, pero no para actualizar estados de información.

Dado que estos paradigmas especifican la solución sin indicar cómo construirla, en principio suprimen la necesidad de probar que el valor calculado es el valor solución. En la práctica, las soluciones son todavía producidas como construcciones más bien que cómo especificaciones. Por lo que los paradigmas resultantes y los lenguajes que los soportan no son verdaderamente declarativos, sino pseudodeclarativos. En este grupo se encuentran: el funcional, el lógico y el de transformación.

En definitiva, los paradigmas declarativos no son soluciones inherentes de tipos serie o paralelo, ya que no dirigen la secuencia de control y no pueden alterar el natural del algoritmo. No obstante, los paradigmas pseudodeclarativos necesitan al menos un limitado grado de secuencia, y por lo tanto admiten versiones en serie y paralelo

A modo de resumen se pueden clasificar los lenguajes de programacion en niveles, estilos y campos de aplicación.

NIVELES.

- Lenguajes de bajo nivel:

Lenguaje máquina. Es el lenguaje que esta especializado en entender directamente la computadora. Para ello utiliza el alfabeto binario, que consta de dos unicos numeros: 0 y 1. Sus instrucciones son cadenas binarias que especifican una operacion, ademas de las operaciones de memoria. La ventaja que tiene este lenguaje es que tiene una mayor velocidad de ejecucion que cualquier otro lenguaje de programacion existente. Por contra, es poco fiable, ya que fue el primer lenguaje computacional y ademas tiene una gran dificultad para verificar y poner a punto los programas.

- Simbólicos.
  - Ensambladores. Intenta deflexibilizar la representacion de los distintos campos, no utilizando codigo binario y si un lenguaje mas próximo al de escritura.
  - Alto Nivel.

ESTILOS O PARADIGMAS DE PROGRAMACIÓN.

- Programación imperativa:

Consiste en una secuencia de instrucciones que al ejecutarse procesan los datos obteniendo el resultado buscado.

- Programación orientada a objetos:

Consiste en considerar los conjuntos de datos como objetos activos utilizando para ello el paso de mensajes que se emiten y reciben en los objetos mediante interfaces bien definidas.

- Programación funcional:

Consiste en la utilización y definición de funciones. Estas funciones son como cajas negras que reciben entradas y crea resultados con ellas.

- Programación declarativa:

Utiliza un principio de razonamiento lógico, basado en deducciones.

- Programación guiada por eventos:

Consiste en un programa principal, cuyo lazo indefinido reacciona en respuesta a la recepción de incidencias generadas en orden e instantes de tiempo no predecibles.

- Programación concurrente:

Este estilo esta asociado a la computación paralela, los paradigmas seran explicados con mayor importancia en la siguiente pregunta.

DOMINIOS DE APLICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN.

- Aplicaciones científico-técnicas:

Se busca la realización de calculos complejos con gran precisión y la obtención de resultados en tiempos razonables.

- Aplicaciones de gestión de informática:

Se trata de definir bases de datos, recuperar información de archivos y generar informes.

- Aplicaciones de inteligencia artificial:

Consiste en manipular símbolos, representación funcional, etc.

- Aplicaciones de programación de sistemas:

Consiste en el desarrollo de programas que esté lo mejor posible adaptado a la arquitectura del computador.

- Aplicaciones web:

Consite en la implementación de modelos interactivos para facilitar la relación del usuario con el sistema.

5. TRADUCTORES DEL LENGUAJE.[6,2]

Un traductor es un programa que traduce un programa de uno a otro lenguaje. Cuando programamos en un lenguaje distinto al lenguaje máquina, los programas diseñados deben ser traducidos a código binario, para que así las intrucciones en ellos especificadas puedan ser entendidas y ejecutadas por el procesador de un computador. El sistema o programa de software encargado de traducir estos programas (denominados programas fuentes) puede ser un ensamblador, intérprete o compilador. También reciben el nombre de traductores del lenguaje.

Como ya se ha comentado en el punto anterior el procesador de un computador sólo puede interpretar y ejecutar código máquina, por tanto para ejecutar un programa redactado en un lenguaje de alto nivel es necesario traducirlo a lenguaje máquina. Los lenguajes de alto nivel posibilitan la utilización de una simbología y una terminología próximas a las usadas tradicionalmente en la descripcion de problemas.

Los ensambladores, intérpretes y compiladores se hacen necesarios dada la gran dificultad de traducir los programas al lenguaje máquina entendible por el ordenador.

5.1. ENSAMBLADORES.

Son los encargados de transformar o traducir directamente (lo que aporta al computador mayor velocidad de operación) los programas escritos en ensamblador a su equivalente en código máquina o código binario para que puedan ser ejecutados por el procesador.

Lenguaje ensamblador es la primera abstracción del lenguaje máquina , consistente en asociar a los opcodes palabras clave que faciliten su uso por parte del programador.

Una característica que hay que resaltar, es que al depender estos lenguajes del hardware, hay un distinto lenguaje de máquina (y, por consiguiente, un distinto Lenguaje Ensamblador) para cada CPU.

5.2. INTERPRETES.

Un intérprete es un programa de software encargado de procesar y traducir cada instrucción o sentencia de un programa escrito en un lenguaje de alto nivel a código máquina y después ejecutarla es decir, que el microprocesador ejecuta la orden o instrucción una vez traducida y después de comprobar que no existe error alguno de sintaxis.

La traducción o interpretación y la ejecución no se realizan como procesos independientes, sino en una misma operación e instrucción por instrucción.

Un intérprete suele proporcionar un editor a través del cual se puede escribir el correspondiente programa fuente, facilitando así la edición y su posterior interpretación y ejecución, permitiendo la comprobación y corrección de los posibles errores producidos durante el desarrollo del mismo.

Una de las principales desventajas de un intérprete es su lentitud en sucesivas ejecuciones del programa, ya que debe traducir y ejecutar cada una de las instrucciones de código del programa desarrollando en alto nivel, repitiendo así todo el proceso de traducción y ejecución.

5.3. COMPILADORES

Un compilador es un programa de software escrito en algún lenguaje de programación cuyo objetivo es traducir el correspondiente programa fuente (fichero constituido por un conjunto de instrucciones desarrolladas en un lenguaje de alto nivel) a su equivalente en código máquina, también denominado programa objeto.

La diferencia más destacables entre un compilador y un intérprete radica en que mientras que un intérprete acepta un programa fuente que traduce y ejecuta simultáneamente analizando cada sentencia por separado, un compilador efectúa dicha operación en dos fases independientes.

1-Traduce completamente el programa fuente a código máquina.

2- Ejecuta el programa.

Para comprobar el buen funcionamiento del programa una vez editado, éste debe ser compilado, es decir, traducido a código máquina en su totalidad, En ejecuciones sucesivas, un programa compilado (a diferencia de uno interpretado) no necesita tiempo de ejecución.

Se puede decir , por tanto, que los compiladores presentan todo lo bueno de los ensambladores y los intérpretes.

5.3.1. FASES DEL PROCESO DE COMPILACIÓN.

5.3.1.1 Edición.

Consiste en la escritura del programa (empleando un lenguaje de programación previamente seleccionado y siguiendo ciertas reglas sintácticas y semánticas) y su posterior grabación sobre un soporte de almacenamiento permanente (por ejemplo, en un disco). La edición del programa fuente debe realizarse mediante la utilización de un editor, que puede formar parte o no del compilador. En esta fase se obtiene el denominado programa fuente.

5.3.1.2 Compilación.

En esta fase se traduce el programa fuente a su equivalente en código máquina, obteniendo en caso de no producirse ningún error, el denominado programa objeto. En caso de producirse errores, el compilador los mostrará utilizando los mensajes correspondientes, que nos permitirán corregir el programa fuente y proceder de nuevo a su compilación.

5.3.1.3 Linkado.

Esta fase también recibe el nombre de montaje y consiste en unir o enlazar el programa objeto obtenido en la fase de compilación con determinadas rutinas internas del lenguaje y, si el método de programación es modular, se enlazan los distintos módulos para obtener así el programa ejecutable.

5.3.1.4 Ejecución

Esta fase consiste en la llamada del programa ejecutable por el sistema operativo. Inicialmente se realizan las pruebas necesarias para detectar posibles errores y comprobar el buen funcionamiento del programa mediante el uso de unos juegos de pruebas que especifican los resultados que se desean obtener en función de unos determinados datos de entrada.

Se pueden dar diversos tipos de errores en la ejecución, entre los que se encuentran:

- Datos de entrada incorrectos que producen una parada del sistema (por ejemplo, introducir un dividendo con valor cero en una operación de división).
- Bucles mal definidos que producen un funcionamiento continuo del programa (por ejemplo, un bucle sin fin o bucle infinito).
- Datos de salida incorrectos, producidos por un mal desarrollo del programa o ambigüedad en las especificaciones del usuario.

6. BIBLIOGRAFÍA.

1. PAREJA, Cristobal; ADEYRO, Ángel; OJEDA, Manuel. *Introducción a la Informática.* 1ª Edición. Madrid: Editorial Complutense, Febrero 1994. ISBN: 84-7491-4892.

2. PRIETO ESPINOSA, Alberto; PRIETO CAMPOS, Beatriz. *Conceptos de Informática.* 1ª Edición en Español. Editorial McGraw-Hill, 2005. ISBN: 84-481-9857-3.

3. Monografías. *Las Tendencias en los Lenguajes de Programación*[en linea]: Trabajos.
<http://www.monografias.com/trabajos/tendprog/tendprog.shtml> [Consulta: 25 de Septiembre de 2006].

4. Cibercalli. *Historia de los Lenguajes de Programación*[en línea]:
<http://www.cibercalli.com/erick/hackingnews/historia-de-los-lenguajes-de-programacion> [Consulta: 26 de Septiembre de 2006].

5. GARCIA MOLINA, Jesús. J., et al. *Una Introducción a la programación (Un enfoque algoritmico).* 1ª Edición. Editorial Thomson. ISBN: 84-9732-185-5.

6. QUERO CATALINAS, Enrique; LÓPEZ HERRANZ, José. *Programación en Lenguajes Estructurados.* Editorial Paraninfo, 1997. ISBN: 84-283-2412-3.