

Liceo Compu-Market

Catedrático: Erick Gonzales

Catedra: programación

Tema:

Temario

Nombre: caren Amanda Cifuentes Arias

Grado: 5to BACO

Sección: “A”

Fecha: 19/04/2017

INDICE

TEMA PÁGINA

Caratula 1

Índice 2

Introducción 3

Historia 4-5

¿Qué es programación? 6

Lenguajes de Programación 7

Tipos de lenguajes de programación 8

Programas y algoritmos 9-21

Compilación 22

Objetivos de la programación 23

Entorno de desarrollo integrado 24-26

Conclusión 25

INTRODUCCION

El presente trabajo trata sobre todo lo que engloba la programación en el podremos ver que es programación sus tipos de leguaje etc.

Historia

Para que la computadora entienda nuestras instrucciones debe usarse un lenguaje específico conocido como código máquina, el cual la máquina comprende fácilmente, pero que lo hace excesivamente complicado para las personas. De hecho sólo consiste en cadenas extensas de números

Para facilitar el trabajo, los primeros operadores de computadoras decidieron hacer un traductor para reemplazar los 0 y 1 por palabras o abstracción de palabras y letras provenientes del inglés; éste se conoce como lenguaje ensamblador. Por ejemplo, para sumar se usa la letra A de la palabra inglesa add (sumar). El lenguaje ensamblador sigue la misma estructura del lenguaje máquina, pero las letras y palabras son más fáciles de recordar y entender que los números.

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 semejante a la de los lenguajes escritos por los humanos, denominados también lenguajes de alto nivel.

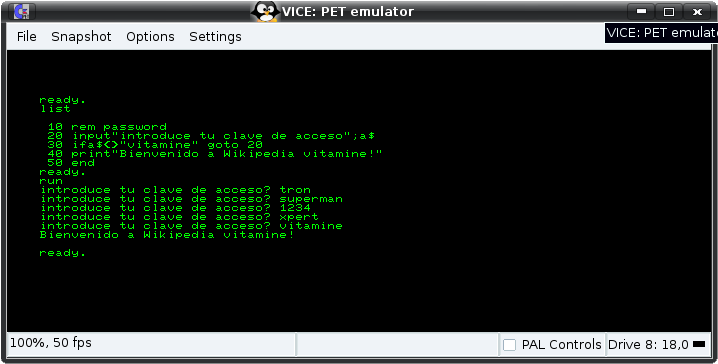
El primer programador de computadora que se haya conocido fue una mujer: Ada Lovelace, hija de Anabella Milbanke Byron y Lord Byron. Anabella inició en las matemáticas a Ada quien, después de conocer a Charles Babbage, tradujo y amplió una descripción de su máquina analítica. Incluso aunque Babbage nunca completó la construcción de cualquiera de sus máquinas, el trabajo que Ada realizó con éstas le hizo ganarse el título de primera programadora de computadoras del mundo. El nombre del lenguaje de programación Ada fue escogido como homenaje a esta programadora.

A finales de 1953, John Backus sometió una propuesta a sus superiores en IBM para desarrollar una alternativa más práctica al lenguaje ensamblador para programar la computadora central IBM 704. El histórico equipo Fortran de Backus consistió en los programadores Richard Goldberg, Sheldon F. Best, Harlan Herrick, Peter Sheridan, Roy Nutt, Robert Nelson, Irving Ziller, Lois Haibt y David Sayre.2

El primer manual para el lenguaje Fortran apareció en octubre de 1956, con el primer compilador Fortran entregado en abril de 1957. Esto era un compilador optimizado, porque los clientes eran reacios a usar un lenguaje de alto nivel a menos que su compilador pudiera generar código cuyo desempeño fuera comparable al de un código hecho a mano en lenguaje ensamblador.

En 1960, se creó COBOL, uno de los lenguajes usados aún en la actualidad, en informática de gestión.

A medida que la complejidad de las tareas que realizaban las computadoras aumentaba, se hizo necesario disponer de un método más eficiente para programarlas. Entonces, se crearon los lenguajes de alto nivel, como lo fue BASIC en las versiones introducidas en los microordenadores de la década de 1980. Mientras que una tarea tan sencilla como sumar dos números puede necesitar varias instrucciones en lenguaje ensamblador, en un lenguaje de alto nivel bastará una s



¿Qué es Programación?

Se llama Programación a la implementación de un algoritmo en un determinado lenguaje de programación, para realizar un programa.

Algoritmo es una secuencia no ambigua, finita y ordenada de instrucciones que han de seguirse para resolver un problema.

Programa (Software en inglés) es una secuencia de instrucciones que una computadora puede interpretar y ejecutar.

El proceso de creación de software es materia de la ingeniería del software, una de las ramas propias de la Ingeniería Informática.

Según Niklaus Wirth un programa está formado por algoritmos y estructura de datos.

Se han propuesto diversas técnicas de programación, cuyo objetivo es mejorar tanto el proceso de creación de software como su mantenimiento. Entre ellas se pueden mencionar las programaciones lineales, estructurada, modular y orientada a objetos

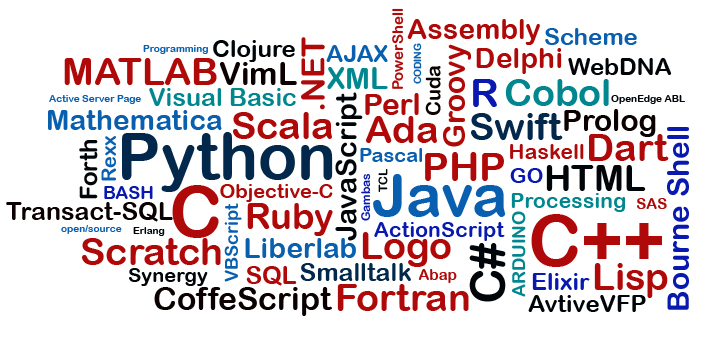
Lenguajes de Programación

Lenguaje de programación es el idioma utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Hay muchos lenguajes de programación, pero para programar no es necesario conocer todos los lenguajes, es igual que cuando hablamos, podemos comunicarnos en español aunque no sepamos alemán. Aunque la palabra debería ser idioma ya que lenguaje realmente abarca todos los idiomas pero en computación equivocadamente se usa el término lenguaje cuando el término correcto es idiomas de programación.

En la actualidad los lenguajes de programación están escritos para ser comprensibles por el ser humano, a este código se le llama código fuente, pero no es comprendido por la máquina ya que esta solo maneja el lenguaje binario.

La compilación es el proceso de traducir un programa en código fuente a programa en código objeto (que usa el lenguaje binario), el programa encargado de compilar se llama compilador. La mayoría de software de programación trae su propio compilador.



Tipos de lenguajes de programación

Los lenguajes de programación se clasifican en varios tipos, los hay por nivel, y por la forma como se ejecutan.

Por nivel: Hay lenguajes de programación de alto nivel y lenguajes de bajo nivel. Los lenguajes de alto nivel permiten que con pocas palabras se logre hacer lo mismo que se logra con un lenguaje de bajo nivel.

Ejemplos de lenguajes de alto nivel: C++, Python, Java

Ejemplos de lenguaje de bajo nivel: assembler

Por la forma como se ejecutan Hay lenguajes compilados e interpretados.

Los lenguajes compilados necesitan de un programa especial que lea el código fuente y cree un archivo binario ejecutable para una plataforma específica. Ejm: C++, Pascal.

Los lenguajes interpretados necesitan de un programa que traduzca en directo el código fuente escrito a instrucciones de la plataforma en la que se ejecutan. Ejm: Python, Visual Basic, Script.

Los primeros son más rápidos, mientras que los segundos son más lentos, esto debido a que al compilar un programa las ordenes son más entendibles para la computadora, mientras que al interpretarlo la máquina primero debe leer el código y convertir al paso las instrucciones a instrucciones de máquina entendibles para ella.



Programas y algoritmos

Un algoritmo es una secuencia no ambigua, finita y ordenada de instrucciones que han de seguirse para resolver un problema. Un programa normalmente implementa (traduce a un lenguaje de programación concreto) uno o más algoritmos. Un algoritmo puede expresarse de distintas maneras: en forma gráfica, como un diagrama de flujo, en forma de código como en pseudocódigo o un lenguaje de programación, en forma explicativa.

Los programas suelen subdividirse en partes menores, llamadas módulos, de modo que la complejidad algorítmica de cada una de las partes sea menor que la del programa completo, lo cual ayuda al desarrollo del programa. Esta es una práctica muy utilizada y se conoce como "refino progresivo".

Según Niklaus Wirth, un programa está formado por los algoritmos y la estructura de datos.

La programación puede seguir muchos enfoques, o paradigmas, es decir, diversas maneras de formular la resolución de un problema dado. Algunos de los principales paradigmas de la programación son:

Programación declarativa

Programación estructurada

Programación modular

Programación orientada a objetos

Programación declarativa

La programación declarativa, en contraposición a la programación imperativa, es un paradigma de programación que está basado en el desarrollo de programas especificando o "declarando" un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución. La solución es obtenida mediante mecanismos internos de control, sin especificar exactamente cómo encontrarla (tan sólo se le indica a la computadora qué es lo que se desea obtener o qué es lo que se está buscando). No existen asignaciones destructivas, y las variables son utiliza

**Tipos**

Existen varios tipos de lenguajes declarativos:

Los lenguajes lógicos, como Prolog.

Los lenguajes algebraicos, como Maude y SQL.

Los lenguajes funcionales, como Haskell y Erlang.

Ventajas[editar]

**Prolog**

El Prolog (o PROLOG), proveniente del francés PROgrammation en LOGique,1 es un lenguaje para programar artefactos electrónicos mediante el paradigma lógico con técnicas de producción final interpretada. Es bastante conocido en el área de la Ingeniería Informática para investigación en Inteligencia Artificial.

**Maude y SQL**

MAUDE es un lenguaje de programación para especificaciones formales mediante el uso de términos algebraicos. Se trata de un lenguaje interpretado que permite la verificación de propiedades y transformaciones sobre modelos y que permite ejecutar la especificación como si fuera un prototipo.

SQL: (por sus siglas en inglés Structured Query Language; en español lenguaje de consulta estructurada) es un lenguaje específico del dominio que da acceso a un sistema de gestión de bases de datos relacionales que permite especificar diversos tipos de operaciones en ellos. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como hacer cambios en ellas. SQL (Lenguaje estructurado de consulta) es un lenguaje específico de dominio que se usa en la programación y está diseñado para manejar los datos que hay en un sistema de base de datos, o para el procesamiento de flujos de datos en un sistema de gestión de flujos.

**Haskell y Erlang.**

Haskell (pronunciado /hæskəl/)1 es un lenguaje de programación estandarizado multi-propósito puramente funcional con semánticas no estrictas y fuerte tipificación estática. Su nombre se debe al lógico estadounidense Haskell Curry. En Haskell, "una función es un ciudadano de primera clase" del lenguaje de programación. Como lenguaje de programación funcional, el constructor de controles primario es la función. El lenguaje tiene sus orígenes en las observaciones de Haskell Curry y sus descendientes intelectuales.

Erlang es un lenguaje de programación concurrente (u orientado a la concurrencia) y un sistema de ejecución que incluye una máquina virtual (BEAM) y bibliotecas (OTP).4

El subconjunto de programación secuencial de Erlang es un lenguaje funcional, con evaluación estricta, asignación única, y tipado dinámico. Fue diseñado en la compañía Ericsson para realizar aplicaciones distribuidas, tolerantes a fallos, soft-real-time y de funcionamiento ininterrumpido. Proporciona el cambio en caliente de código de forma que éste se puede cambiar sin parar el sistema. Originalmente, Erlang era un lenguaje propietario de Ericsson, pero fue cedido como software de código abierto en 1998. La implementación de Ericsson es principalmente interpretada, pero también incluye un compilador HiPE (sólo soportado en algunas plataformas).

Programación estructurada

La programación estructurada es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora, utilizando únicamente subrutinas y tres estructuras: secuencia, selección (if y switch) e iteración (bucles for y while), considerando innecesario y contraproducente el uso de la instrucción de transferencia incondicional (GOTO), que podría conducir a "código espagueti", que es mucho más difícil de seguir y de mantener, y era la causa de muchos errores de programación.

**Orígenes de la programación estructurada**

A finales de los años 1970 surgió una nueva forma de programar que no solamente daba lugar a programas fiables y eficientes, sino que además estaban escritos de manera que facilitaba su mejor comprensión, no sólo proveyendo ventajas durante la fase de desarrollo, sino también posibilitando una más sencilla modificación posterior.

El teorema del programa estructurado, propuesto por Böhm-Jacopini, demuestra que todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:

Secuencia.

Instrucción condicional.

Iteración (bucle de instrucciones) con condición al principio.

Solamente con estas tres estructuras se pueden escribir todos los programas y aplicaciones posibles. Si bien los lenguajes de programación tienen un mayor repertorio de estructuras de control, estas pueden ser construidas mediante las tres básicas citadas.

**Ventajas de la programación estructurada**

Ventajas de la programación estructurada comparada con el modelo anterior (hoy llamado despectivamente código espagueti).

\*Los programas son más fáciles de entender, pueden ser leídos de forma secuencial y no hay necesidad de hacer engorrosos seguimientos en saltos de líneas (GOTO) dentro de los bloques de código para intentar entender la lógica.

\*La estructura de los programas es clara, puesto que las instrucciones están más ligadas o relacionadas entre sí.

\*Reducción del esfuerzo en las pruebas y depuración. El seguimiento de los fallos o errores del programa (debugging) se facilita debido a su estructura más sencilla y comprensible, por lo que los errores se pueden detectar y corregir más fácilmente.

\*Reducción de los costos de mantenimiento. Análogamente a la depuración, durante la fase de mantenimiento, modificar o extender los programas resulta más fácil.

\*Los programas son más sencillos y más rápidos de confeccionar.

\*Se incrementa el rendimiento de los programadores.

**Lenguajes de programación estructurada**

Es posible hacer la programación estructurada en cualquier lenguaje de programación, aunque es preferible usar algo como un lenguaje de programación procedimental. Algunos de los lenguajes utilizados inicialmente para programación estructurada incluyen: ALGOL, Pascal.

\* ALGOL (o Algol) a un lenguaje de programación. La voz es un acrónimo de las palabras inglesas Algorithmic Language (lenguaje algorítmico).

Fue muy popular en las universidades durante los años 60, pero no llegó a cuajar como lenguaje de utilización comercial.

**Ejemplo de programa en ALGOL 60**

procedure Absmax(a) Dimensiones:(n, m) Resultado:(y) Subíndices:(i, k);

value n, m; array a; integer n, m, i, k; real y;

comment De la matriz a se toma el elemento con el valor absoluto mayor y se coloca en y.

Los subíndices del elemento se colocan en i y k;

begin integer p, q;

y:= 0; i:= k:= 1;

for p:= 1 step 1 until n do

for q:= 1 step 1 until m do

if abs(a[p, q]) > y then

begin y:= abs(a[p, q]);

i:= p; k:= q

end

end Absmax

**Pascal** es un lenguaje creado por el profesor suizo Niklaus Wirth entre los años 1968 y 1969 y publicado en 1970. Su objetivo era crear un lenguaje que facilitara el aprendizaje de programación a sus alumnos, utilizando la programación estructurada y estructuración de datos. Sin embargo con el tiempo su utilización excedió el ámbito académico para convertirse en una herramienta para la creación de aplicaciones de todo tipo.

Pascal se caracteriza por ser un lenguaje de programación estructurado fuertemente tipado. Esto implica que:

El código está dividido en porciones fácilmente legibles llamadas funciones o procedimientos. De esta forma Pascal facilita la utilización de la programación estructurada en oposición al antiguo estilo de programación monolítica.

El tipo de dato de todas las variables debe ser declarado previamente para que su uso quede habilitado.

**Ejemplo**

Hola Mundo

Un sencillo ejemplo del programa Hola Mundo:

PROGRAM HolaMundo (OUTPUT);

BEGIN

WriteLn('¡Hola Mundo!')

{ como la siguiente instrucción no es ejecutable "end."

no se requiere ; aunque puede ponerse según las

preferencias del programador }

END.

La programación modular

es un paradigma de programación que consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable.

Se presenta históricamente como una evolución de la programación estructurada para solucionar problemas de programación más grandes y complejos de lo que esta puede resolver.

Al aplicar la programación modular, un problema complejo debe ser dividido en varios subproblemas más simples, y estos a su vez en otros subproblemas más simples. Esto debe hacerse hasta obtener subproblemas lo suficientemente simples como para poder ser resueltos fácilmente con algún lenguaje de programación. Esta técnica se llama refinamiento sucesivo, divide y vencerás ó análisis descendente (Top-Down).

Un 'módulo' es cada una de las partes de un programa que resuelve uno de los subproblemas en que se divide el problema complejo original. Cada uno de estos módulos tiene una tarea bien definida y algunos necesitan de otros para poder operar. En caso de que un módulo necesite de otro, puede comunicarse con éste mediante una interfaz de comunicación que también debe estar bien definida.

Si bien un módulo puede entenderse como una parte de un programa en cualquiera de sus formas y variados contextos, en la práctica se los suele tomar como sinónimos de procedimientos y funciones. Pero no necesaria ni estrictamente un módulo es una función o un procedimiento, ya que el mismo puede contener muchos de ellos. No debe confundirse el término "módulo" (en el sentido de programación modular) con términos como "función" o "procedimiento", propios del lenguaje que lo soporte.

La programación orientada a objetos

(POO, u OOP según sus siglas en inglés) es un paradigma de programación que viene a innovar la forma de obtener resultados. Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial.

Muchos de los objetos pre-diseñados de los lenguajes de programación actuales permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus propias bibliotecas.

Está basada en varias técnicas, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.

Su uso se popularizó a principios de la década de 1990. En la actualidad, existe una gran variedad de lenguajes de programación que soportan la orientación a objetos.

La POO es una forma de programar que trata de encontrar una solución a estos problemas. Introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

[**Clase**](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica))

Definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ella.

[**Herencia**](https://es.wikipedia.org/wiki/Herencia_(inform%C3%A1tica))

Por ejemplo, herencia de la clase C a la clase D, es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables públicas declaradas en C. Los componentes registrados como "privados" (*private*) también se heredan, pero como no pertenecen a la clase, se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. En el caso de los componentes registrados como "protegidos" (*protected*) también se heredan, pero solo para esa clase, no para futuras clases heredadas. Esto es así para mantener hegemónico el ideal de POO.

[**Objeto**](https://es.wikipedia.org/wiki/Objetos_(programaci%C3%B3n_orientada_a_objetos))

Instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa).

[**Método**](https://es.wikipedia.org/wiki/M%C3%A9todo_(programaci%C3%B3n_orientada_a_objetos))

Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

**Evento**

Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento la reacción que puede desencadenar un objeto; es decir, la acción que genera.

**Atributos**

Características que tiene la clase.

**Mensaje**

Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

**Propiedad o atributo**

Contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

**Estado interno**

Es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos). No es visible al programador que maneja una instancia de la clase.

**Componentes de un objeto**

Atributos, identidad, relaciones y métodos.

**Identificación de un objeto**

Un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

En comparación con un lenguaje imperativo, una "variable" no es más que un contenedor interno del atributo del objeto o de un estado interno, así como la "función" es un procedimiento interno del método del objeto.

Algunos lenguajes orientados a objetos

Simula (1967) es aceptado como el primer lenguaje que posee las características principales de un lenguaje orientado a objetos. Fue creado para hacer programas de simulación, en donde los "objetos" son la representación de la información más importante.

Smalltalk (1972 a 1980) es posiblemente el ejemplo canónico, y con el que gran parte de la teoría de la programación orientada a objetos se ha desarrollado.

Entre los lenguajes orientados a objetos se destacan los siguientes:

***Java*** es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.1 2

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

***JavaScript*** (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,3 basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas4 aunque existe una forma de JavaScript del lado del servidor(Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

Desde el 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1, una versión de javascript. Los navegadores más antiguos soportan por lo menos ECMAScript 3. La sexta edición se liberó en julio del 2015.5

JavaScript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo, Java y JavaScript tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

***PHP*** es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Fue creado originalmente por Rasmus Lerdorf en el año 1995. Actualmente el lenguaje sigue siendo desarrollado con nuevas funciones por el grupo PHP.2 Este lenguaje forma parte del software libre publicado bajo la licencia PHP, que es incompatible con la Licencia Pública General de GNU debido a las restricciones del uso del término PHP.3

***Visual FoxPro*** es un lenguaje de programación procedural, orientado a objetos que posee un Sistema Gestor de Bases de datos o Database Management System (DBMS) y Sistema administrador de bases de datos relacionales, producido por Microsoft.

Visual FoxPro provee a los desarrolladores un conjunto de herramientas para crear aplicaciones de bases de datos para el escritorio, entornos cliente/servidor, tablet PC o para la Web.

Entre sus características se pueden enumerar:

Capacidades para el manejo de datos nativos y remotos.

Flexibilidad para crear soluciones de bases de datos.

Lenguaje de Programación Orientado a objetos.

Utilización de Sentencias SQL en forma nativa.

Manejo de vistas, cursores y control completo de estructuras relacionales.

Su propio gestor de base de datos incorporado. Sin embargo, también puede conectarse con servidores de base de datos, tales como Oracle, Microsoft SQL Server o MySQL.

***Visual Basic*** (VB) es un lenguaje de programación dirigido por eventos, desarrollado por Alan Cooper para Microsoft. Este lenguaje de programación es un dialecto de BASIC, con importantes agregados. Su primera versión fue presentada en 1991, con la intención de simplificar la programación utilizando un ambiente de desarrollo que facilitó en cierta medida la programación misma.

La última versión fue la 6, liberada en 1998, para la que Microsoft extendió el soporte hasta marzo de 2008.

En 2001 Microsoft propuso abandonar el desarrollo basado en la API Win32 y pasar a un framework o marco común de librerías, independiente de la versión del sistema operativo, .NET Framework, a través de Visual Basic .NET (y otros lenguajes como C Sharp (C#) de fácil transición de código entre ellos); fue el sucesor de Visual Basic 6.

Aunque Visual Basic es de propósito general, también provee facilidades para el desarrollo de aplicaciones de bases de datos usando Data Access Objects, Remote Data Objects o ActiveX Data Objects.

Visual Basic contiene un entorno de desarrollo integrado o IDE que integra editor de textos para edición del código fuente, un depurador, un compilador (y enlazador) y un editor de interfaces gráficas o GUI.

Visual Basic dio un paso más en innovación y ahora es posible programar aplicaciones Nativas para Android o IPhone utilizando un software de la compañía Anywhere Software que transforma código Visual Basic (creado por dicha compañía) en APPs 100 % nativas en Java para ambos sistemas operativos de dispositivos móviles.1

**Entorno de desarrollo de visual basic**

Existe un único entorno de desarrollo para Visual Basic, desarrollado por Microsoft: Microsoft Visual Basic x.0, correspondientes a versiones desde la 2.0 hasta la 20.0, (con respectivas diferencias entre versiones del lenguaje).

El entorno de desarrollo es muy similar al de otros lenguajes. Realizando una instalación típica del producto, las características básicas se presentan de la siguiente forma:

En la parte superior aparecen tres elementos, en este orden: la barra de título donde figura el nombre del proyecto en curso y su estado (diseño o ejecución); la barra de menú con 13 opciones desplegables y una barra de herramientas; esta última se puede personalizar, posibilitando la inclusión de prácticamente la totalidad de los comandos del IDE.

En la parte central, cubriendo la franja de mayor área, se encuentra el espacio de trabajo. Este incluye y muestra las ventanas del proyecto, las vistas del código fuente de los módulos, los objetos y los controles que contienen las ventanas de la aplicación y el panel de controles.

El panel de controles, que aunque es móvil normalmente está ubicado a la derecha, por defecto cuenta con los siguientes controles:

PictureBox: Caja de imágenes

Label: Etiqueta

TextBox: Caja de texto

Frame: Marco

CommandButton: Botón de comando

CheckBox: Casilla de verificación

OptionButton: Botón de opción

ComboBox: Lista desplegable

ListBox: Lista

HScrollBar: Barra de desplazamiento horizontal

VScrollBar: Barra de desplazamiento vertical

Timer: Temporizador

DriveListBox: Lista de unidades de disco

DirListBox: Lista de directorios

FileListBox: Lista de archivos

Shape: Figura

Line: Línea

Image: Imagen

Data: Conexión a origen de datos

OLE: Contenedor de documentos embebidos compatibles con Object Linking and Embedding

Programación concurrente   
  
Este tipo de programación se utiliza cuando tenemos que realizar varias acciones a la vez.   
  
Se suele utilizar para controlar los accesos de usuarios y programas a un recurso de forma simultanea.   
  
Se trata de una programación más lenta y laboriosa, obteniendo unos resultados lentos en las acciones.   
  
Programación funcional   
  
Se caracteriza principalmente por permitir declarar y llamar a funciones dentro de otras funciones.   
  
Programación lógica   
  
Se suele utilizar en la inteligencia artificial y pequeños programas infantiles. Se trata de una programación basada en el cálculo de predicados (una teoría matemática que permite lograr que un ordenador basándose en hecho y reglas lógicas, pueda dar soluciones inteligentes).

Compilación

El programa escrito en un lenguaje de programación de alto nivel (fácilmente comprensible por el programador) es llamado programa fuente y no se puede ejecutar directamente en una computadora. La opción más común es compilar el programa obteniendo un módulo objeto, aunque también puede ejecutarse en forma más directa a través de un intérprete informático.

El código fuente del programa se debe someter a un proceso de traducción para convertirlo a lenguaje máquina o bien a un código intermedio, generando así un módulo denominado "objeto". A este proceso se le llama compilación.

Habitualmente la creación de un programa ejecutable (un típico.exe para Microsoft Windows o DOS) conlleva dos pasos. El primer paso se llama compilación (propiamente dicho) y traduce el código fuente escrito en un lenguaje de programación almacenado en un archivo de texto a código en bajo nivel (normalmente en código objeto, no directamente a lenguaje máquina). El segundo paso se llama enlazado en el cual se enlaza el código de bajo nivel generado de todos los ficheros y subprogramas que se han mandado compilar y se añade el código de las funciones que hay en las bibliotecas del compilador para que el ejecutable pueda comunicarse directamente con el sistema operativo, traduciendo así finalmente el código objeto a código máquina, y generando un módulo ejecutable.

Estos dos pasos se pueden hacer por separado, almacenando el resultado de la fase de compilación en archivos objetos (un típico .o para Unix, .obj para MS-Windows, DOS); para enlazarlos en fases posteriores, o crear directamente el ejecutable; con lo que la fase de compilación puede almacenarse solo de forma temporal. Un programa podría tener partes escritas en varios lenguajes, por ejemplo, Java, C, C++ y ensamblador, que se podrían compilar de forma independiente y luego enlazar juntas para formar un único módulo ejecutable.

Objetivos de la programación

La programación debe perseguir la obtención de programas de calidad. Para ello se establece una serie de factores que determinan la calidad de un programa. Algunos de los factores de calidad más importantes son los siguientes:

Correctitud. Un programa es correcto si hace lo que debe hacer tal y como se estableció en las fases previas a su desarrollo. Para determinar si un programa hace lo que debe, es muy importante especificar claramente qué debe hacer el programa antes de su desarrollo y, una vez acabado, compararlo con lo que realmente hace.

Claridad. Es muy importante que el programa sea lo más claro y legible posible, para facilitar tanto su desarrollo como su posterior mantenimiento. Al elaborar un programa se debe intentar que su estructura sea sencilla y coherente, así como cuidar el estilo de programación. De esta forma se ve facilitado el trabajo del programador, tanto en la fase de creación como en las fases posteriores de corrección de errores, ampliaciones, modificaciones, etc. Fases que pueden ser realizadas incluso por otro programador, con lo cual la claridad es aún más necesaria para que otros puedan continuar el trabajo fácilmente. Algunos programadores llegan incluso a utilizar Arte ASCII para delimitar secciones de código; una práctica común es realizar aclaraciones en el código fuente utilizando líneas de comentarios. Contrariamente, algunos por diversión o para impedirle un análisis cómodo a otros programadores, recurren al uso de código ofuscado.

Eficiencia. Se trata de que el programa, además de realizar aquello para lo que fue creado (es decir, que sea correcto), lo haga gestionando de la mejor forma posible los recursos que utiliza. Normalmente, al hablar de eficiencia de un programa, se suele hacer referencia al tiempo que tarda en realizar la tarea para la que ha sido creado y a la cantidad de memoria que necesita, pero hay otros recursos que también pueden ser de consideración para mejorar la eficiencia de un programa, dependiendo de su naturaleza (espacio en disco que utiliza, tráfico en la red que genera, etc.).

Portabilidad. Un programa es portable cuando tiene la capacidad de poder ejecutarse en una plataforma, ya sea hardware o software, diferente a aquella en la que se desarrolló. La portabilidad es una característica muy deseable para un programa, ya que permite, por ejemplo, a un programa que se ha elaborado para el sistema GNU/Linux ejecutarse también en la familia de sistemas operativos Windows. Esto permite que el programa pueda llegar a más usuarios más fácilmente.

Un entorno de desarrollo integrado

1 2 o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense). Algunos IDE contienen un compilador, un intérprete, o ambos, tales como NetBeans y Eclipse; otros no, tales como SharpDevelop y Lazarus.

El límite entre un IDE y otras partes del entorno de desarrollo de software más amplio no está bien definido. Muchas veces, a los efectos de simplificar la construcción de la interfaz gráfica de usuario (GUI, por sus siglas en inglés) se integran un sistema controlador de versión y varias herramientas. Muchos IDE modernos también cuentan con un navegador de clases, un buscador de objetos y un diagrama de jerarquía de clases, para su uso con el desarrollo de software orientado a objetos.

***Programación visual***

La programación visual es un marco de usuario en la que generalmente se requiere una IDE. Los IDE visuales le permiten a los usuarios crear nuevas aplicaciones de programación en movimiento, bloques de construcción, o nodos de código para crear diagramas de flujo o diagramas de estructura que luego son compilado o interpretado. Estos diagramas de flujo muchas veces se basan en el lenguaje de modelado unificado.

Esta interfaz ha sido popularizada con los Lego Mindstorms, y se ha mantenido activa por un número de compañías deseando capitalizar el poder de los buscadores personalizados como los fundados en Mozilla. KTechlab apoya el flowcode y es un IDE de código abierto (opensource) y un simulador para desarrollar software para micro-controladores.

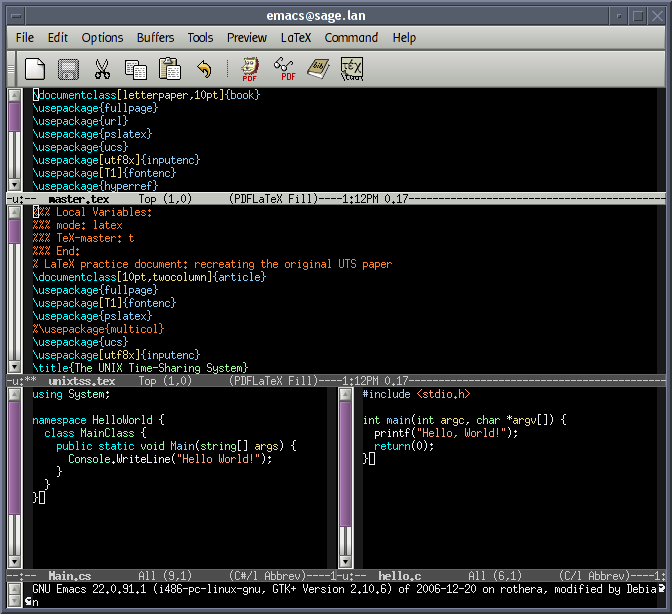
La programación visual también es responsable del poder de la distribución de software (LabVIEW y software EICASLAB). Un primitivo sistema visual de programación, Max, fue modelado a partir de un sintetizador de diseño análogo siendo desarrollado para utilizar el desempeño de la música en tiempo real desde los años 1980. Otro ejemplo primitivo fue Prograph, un programa a base de flujo de datos, originalmente desarrollado para la Macintosh. El ambiente de programación gráfica "Grape" es usado para programar qfix robot kits.

Este acercamiento es también utilizado por software especializados, tales como Openlab, donde el usuario final quiere la flexibilidad completa de un lenguaje de programación, sin la tradicional curva de aprendizaje.

***Soportes del lenguaje***

Algunos IDE soportan múltiples lenguajes, tales como GNU Emacs basados en C y Emacs Lisp, y Eclipse, IntelliJ IDEA, MyEclipse o NetBeans, todos basados en Java, o MonoDevelop, basados en C#.

Normalmente, el soporte para lenguajes alternativos regularmente es proveído por un plug-in, permitiéndoles ser instalados en el mismo IDE, al mismo tiempo.4 Eclipse, y Netbeans tienen plugins para C/C++, Ada, (por ejemplo AdaGIDE), Perl, Python, Ruby, y PHP, los cuales son seleccionados entre extensión de archivos, ambientes o ajustes de proyectos.



Conclusión

La programación engloba muchos Tipos de lenguajes Programas, algoritmos y Objetivos