2º de Bachillerato

Tecnologías de la Información y Comunicación

Contenidos

Conceptos básicos de programación: Hablando con las máquinas: lenguajes de programación

Todos somos conscientes de la importancia de los avances tecnológicos. Cada día, a nuestro alrededor, **interactuamos con máquinas** que intentan hacernos la vida más fácil.

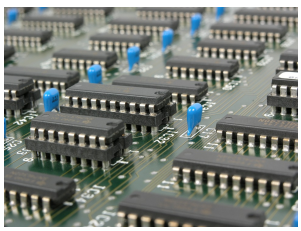

Imagen en pixabay de Peggy-Marco bajo licencia CC

Desde el cajero de un banco, pasando por la centralita electrónica de nuestro coche, o el ordenador que tenemos en el trabajo o en casa.

Este último puede ser una potente herramienta con la que desarrollar la tarea de comunicación con esta y otras máquinas. *Hablando con las máquinas* es en nuestros días una realidad. Para conseguirlo **es indispensable dominar la teoría y práctica de los denominados lenguajes de programación**.

A lo largo del tema, se muestra qué son, sus partes e iniciarnos a cómo usarlos para resolver problemas construyendo programas informáticos.

1. ¿Qué vas a aprender en este tema?

Al finalizar el tema:

- Conocerás los principales tipos de lenguajes de programación.
- Comprenderás el proceso de traducción de un programa informático.
- Aprenderás los elementos básicos de un lenguaje de programación.
- Conocerás las fases o etapas para elaborar un programa.
- Comprenderás la estructura y tipos de instrucciones que contiene un programa.

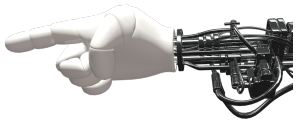

Imagen en pixabay de DirtyOpi bajo licencia CC

El ejemplo comentado en el tema anterior consistía en comparar dos números enteros y devolver el mayor.

A continuación, se detallan las **etapas ordenadas que se deben seguir para realizar un programa** que resuelva el problema planteado. Una aplicación informática (o conjunto de programas) se desarrolla, como hemos comentado, con el objetivo de solucionar un determinado problema.

Para ello se deberá tener en cuenta el llamado ciclo de vida, compuesto por las etapas que se deben seguir secuencial y ordenadamente. Dichas etapas son:

Ciclo de vida

TAREAS	ETAPAS
¿Qué necesidades debemos cubrir? • Comparar dos números enteros y devolver el mayor	**[1]** **Análisis**

¿Qué recursos necesitaremos?
• Hardware: CPU, monitor, ratón, teclado
• Software: Editor de lenguaje de programación
¿Cómo resolvemos el problema analíticamente?
1. Leemos los dos números entrados por teclado
2. Comparamos los dos números
  1. Si el primer número es el mayor lo devuelvo
  2. Si el segundo número es el mayor lo devuelvo
  3. Si ambos números son iguales devuelvo el primero
	[2]
	Diseño

¿Qué lenguaje de programación elegimos? • ¿interpretado? por ejemplo PHP • ¿compilado? por ejemplo C	**[3]** **Codificación**
¿Cómo lo usamos? • Obtener el programa ejecutable en el Sistema Operativo correspondiente • Instalarlo • Ejecutarlo y comprobar los resultados	**[4]** **Explotación**

¿Podemos mejorarlo? • Corrigiendo errores • Aportando mejoras que lo hagan más eficiente	**[5]** **Mantenimiento**

2. Lenguajes de programación

Como se vio en el tema anterior, **un lenguaje de programación es un conjunto de símbolos y caracteres combinados entre sí, de acuerdo con una sintaxis ya definida y respetando unas reglas establecidas, para posibilitar la comunicación con la CPU del ordenador**.

Este proceso implica tres niveles de abstracción:

Niveles de abstracción

Programador	Lenguaje natural	Lenguaje de programación	Lenguaje máquina	CPU del ordenador
	Nivel 1	Nivel 2	Nivel 3	

Un poco de historia

Desde los orígenes del primer lenguaje de programación (antes de 1940) hasta el día de hoy, los lenguajes de programación han ido evolucionando, influenciados en gran medida por los avances tecnológicos.

En la siguiente tabla se **resumen** algunos períodos y **lenguajes** importantes en dicha evolución:

Fecha	Lenguajes importantes
Antes de 1940	Códigos entendibles sólo por la máquina
Años 40	Plankalkül, Sistema de codificación ENIAC
Años 50	Ensamblador, FORTRAN, LISP, COBOL, BASIC
Años 60	Logo, B
Años 70	Pascal, C, Smalltalk, Prolog, Scheme, SQL
Años 80	C++, Ada, MATLAB, Perl, Mathematica
Años 90	Pyton, Visual Basic, HTML, Ruby, Java, Delphy, JavasSript, PHP, R
A partir del año 2000	ActionScript, C#

Curiosidad

Imagen en pixabay de Maialisa bajo licencia CC

En informática, un programa **Hola mundo** es el que imprime el texto «¡Hola, mundo!» en un dispositivo de visualización, en la mayoría de los casos una pantalla de monitor. Este programa suele ser usado como introducción al estudio de un lenguaje de programación, siendo un primer ejercicio típico, y se lo considera fundamental desde el punto de vista didáctico.

El programa *Hola Mundo* también puede ser útil como prueba de configuración para asegurar que el compilador, el entorno de desarrollo y el entorno de ejecución estén instalados correctamente y funcionando. En algunos lenguajes, configurar un conjunto de herramientas básicas completo desde cero hasta el punto en que los programas triviales puedan ser compilados y ejecutados involucra una cantidad de trabajo sustancial. Por esta razón, generalmente es usado un programa muy simple para probar un nuevo conjunto de herramientas.

puertos del sistema, dando a cada uno de dichos niveles un valor de retardo. (Texto de Wikipedia).

A modo de **ejemplo** se muestra el programa Hola mundo **en tres lenguajes de programación** distintos: C, JavaScript y PHP.

Lenguaje **C**	Lenguaje **JavaScript**	Lenguaje **PHP**
`#include<stdio.h>` `/* Programa "Hola Mundo" */` `int main() {` `  printf("Hola mundo\n");` `  return 0;` `}`	`// Programa "Hola Mundo"` `function holamundo() {` `  document.write("Hola mundo",' ');` `}`	`<?php` `  // Programa "Hola Mundo"` `  echo 'Hola mundo',PHP_EOL;` `?>`

Si quieres ver **más implementaciones del programa Hola mundo** en otros lenguajes, en este enlace puedes encontrar más de cien distintos.

2.1. Tipos de lenguajes de programación

La **clasificación** de los lenguajes de programación no es única y en este sentido existen varias:

- Lenguajes de alto y bajo nivel.
- Lenguajes interpretados y compilados.
- Lenguajes orientados o no a objetos.
- Lenguajes de propósito científico u otros.
- Etc.

```
# Funciones:

cambiaNaNporcero<- function(valores){
  resu<- valores
  for(i in 1:nrow(valores)){
    for(j in 2:ncol(valores)){
      resu[i,j]<- replace(resu[i,j],is.na(resu[i,j]),0)
    }
  }
  return(resu)
}

calcula<- function(valores,grupos,funcion,redondeo,nombre){ #Para
  resu <- by(valores,grupos,funcion)
  resu <- do.call(rbind,as.list(resu))
  resu<- data.frame(resu)
  resu<- round(resu,redondeo)
  resu<- cbind.data.frame(row.names(resu),resu)
  colnames(resu)<- c("Aula",nombre)
  resu<- na.omit(resu)
  row.names(resu)<- seq(1:nrow(resu))
  return(resu[2])
}
```

Imagen de creación propia bajo licencia CC

Las combinaciones entre ellas dan lugar también a clasificaciones según su función, estructura, conversión, etc. En este curso se tomará como referencia la primera de las especificadas, es decir, diferenciando entre lenguajes de alto nivel (más cercanos al lenguaje natural humano y por tanto a la forma de pensar del programador), y lenguajes de bajo nivel (entendibles por la máquina según su diseño físico).

Ejemplos de **lenguajes de alto nivel** son:

- Java: de alto nivel, de propósito general, orientado a objetos, compilado e independiente de la plataforma o hardware de la máquina.
- R: de alto nivel, de propósito estadístico, orientado a objetos, interpretado e independiente del sistema operativo.
- PHP: de alto nivel, de uso general, para plataformas web, orientado a objetos, interpretado del lado del servidor.

Ejemplos de **lenguajes de bajo nivel** son:

- Ensamblador: de bajo nivel que implementa una representación simbólica de los códigos de máquina binarios.
- Lenguaje máquina: de bajo nivel, interpretado directamente por el microprocesador.

En la tabla siguiente, se muestra una lista de algunos lenguajes **según el campo o área de aplicación** en que se emplean:

Lenguaje	Principal área de aplicación	Compilado/interpretado
ADA	Tiempo real	Lenguaje compilado
BASIC	Programación para fines educativos	Lenguaje interpretado
C	Programación de sistema	Lenguaje compilado
C++	Programación de sistema orientado a objeto	Lenguaje compilado
Cobol	Administración	Lenguaje compilado
Fortran	Cálculo	Lenguaje compilado
Java	Programación orientada a Internet	Lenguaje intermediario
MATLAB	Cálculos matemáticos	Lenguaje interpretado
R	Cálculos matemáticos y estadísticos	Lenguaje interpretado
LISP	Inteligencia artificial	Lenguaje intermediario
Pascal	Educación	Lenguaje compilado
PHP	Desarrollo de sitios web dinámicos	Lenguaje interpretado
Scheme	Inteligencia artificial	Lenguaje interpretado
Perl	Procesamiento de cadenas de caracteres	Lenguaje interpretado

Para saber más

Imagen en pixabay de Clker-Free-Vector-Images bajo licencia CC

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.

El lenguaje Java se creó con cinco **objetivos** principales:

1. Debería usar el paradigma de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

El diseño de Java, su robustez, el respaldo de la industria y su fácil portabilidad han hecho de Java uno de los lenguajes con un mayor crecimiento y amplitud de uso en distintos ámbitos de la industria de la informática. Los **entornos de funcionamiento** de este lenguaje son:

- En dispositivos móviles y sistemas embebidos
- En el navegador web
- En sistemas de servidor
- En aplicaciones de escritorio

(Texto de Wikipedia)

2.2. Proceso de traducción

Imagen en pixabay de geralt bajo licencia CC

Anteriormente, ya hemos comentado que la CPU sólo entiende el lenguaje máquina. Este, para cada máquina concreta, está compuesto por instrucciones en formato binario (0 ó 1).

El programador, para hacer esta tarea más sencilla y productiva, emplea lenguajes de programación específicos para construir un programa informático. Dicho programa se denomina código o programa fuente, no entendible por la máquina, así es necesario un proceso de traducción para convertir dicho programa **a otro lenguaje que sí pueda ser entendido por la CPU** en la que vamos a trabajar.

Esta traducción o conversión se puede hacer **de dos formas distintas** que se denominan interpretación o compilación. Por tanto, existen dos programas o aplicaciones encargadas de realizar cada tipo de traducción, estos son, los intérpretes y los compiladores.

1. Intérpretes:

Un intérprete es a su vez un programa que convierte o traduce un programa, escrito en un lenguaje de programación de alto nivel, instrucción a instrucción o sentencia a sentencia de una forma secuencial, es decir, cada instrucción es traducida una vez traducida la instrucción anterior, así traducción y ejecución se realizan conjuntamente. La principal ventaja de esta forma de trabajar es que la ejecución es inmediata, así pueden corregirse posibles errores que vayan surgiendo durante el proceso y continuar a partir de ese punto.

- Ejemplos de lenguajes interpretados son: JavaScript, Perl, PHP, Pyton, Lisp, Scheme, R, HTML.

2. Compiladores:

Un compilador es un programa, que a partir del todo código fuente genera lo que se llama el código objeto en lenguaje máquina. Así, este proceso lo realiza en dos fases independientes, la primera traduciendo completamente el programa fuente a código máquina y la segunda ejecutando dicho código máquina u objeto.

- Ejemplos de lenguajes compilados son: C, C++, C#, COBOL, Delphi, Fortran, Pascal.

El proceso de compilación requiere más tiempo que en un intérprete, sin embargo, una vez traducido, la ejecución es más rápida al trabajar directamente con el código máquina.

Para saber más

Imagen en pixabay de SuperMerrily bajo licencia CC

Es importante considerar **en qué Sistema Operativo ejecutaremos nuestro programa**. Hay lenguajes sólo para plataformas Windows (C#, VisualBasic, ASP, etc), para iOS (Objetive-C, Swift, etc), para Linux (Shell Script, etc) o multiplataforma (Java, PHP, R, C, Python, etc).

Atendiendo a esta característica el software para realizar la compilación será diferente. Sin embargo, el proceso básico de **compilación** consta de una serie de **etapas** ordenadas:

1. Edición

En esta etapa se obtiene el código fuente, escribiendo en un

2. Compilación

Esta es la etapa pura de traducción del código fuente a código máquina o programa objeto, si no se produce ningún error en dicho proceso. En caso de errores, el compilador emite una serie de mensajes que ayudan al programador a corregirlos, para proceder nuevamente a la compilación.

3. Linkado

El programa obtenido en la etapa anterior (programa objeto) debe ser montado o linkado, uniendo el programa objeto con las funciones de librería que en él se han referenciado o enlazando módulos para conseguir el programa ejecutable.

4. Ejecución

Esta última etapa consiste en la llamada, por parte del sistema operativo, del programa ejecutable. De esta forma el usuario o sistema, a partir de unos datos de entrada, obtendrá los resultados correspondientes.

Proceso de compilación

Edición	Compilación	*¿errores?*	Linkado	Ejecución
Programa fuente	Programa objeto	Sí: volver a compilar No: pasar a linkado	Programa ejecutable	Mostrar resultados

3. Elementos básicos del lenguaje

Aunque cada lenguaje posee sus propias **características**, hay algunas que son **comunes** o de las que la mayoría disponen. En esta sección resumimos las principales, mostrando los elementos básicos necesarios para construir un programa informático. En los temas sucesivos se expondrán con más detalle.

Imagen en pixabay de geralt bajo licencia CC

1. Tipos de datos

Según su uso y función los datos con los que puede trabajar un lenguaje de programación son principalmente:

- Entero: para representar números enteros.
- Real: para representar números con punto decimal.
- Cadena: para datos de tipo texto o carácter.

2. Palabras reservadas

Se trata de un conjunto de palabras que el lenguaje de programación considera propias de su sintaxis, y son empleadas para construir instrucciones. Este tipo de palabras no pueden ser usadas, por ejemplo, para crear o nombrar variables.

3. Operadores

Los operadores son usados para crear instrucciones realizando cálculos matemáticos, comparaciones u operaciones lógicas. Así encontramos tres tipos:

Operadores aritméticos

- Exponenciación: ^
- Producto y división: * y /
- Suma y resta: + y -

Operadores relacionales

- Mayor: >
- Mayor o igual: >=
- Menor: <
- Menor o igual: <=
- Igual: ==
- Distinto: <>

Operadores lógicos

- Y: AND
- O: OR
- NO: NOT
- O exclusivo: XOR

4. Constantes y variables

Ambos deben ser definidos por un identificador para hacer referencia a ellos durante el programa.

- Las constantes son datos que no cambian durante la ejecución del programa.
- Las variables por el contrario son datos que pueden cambiar durante el proceso de ejecución.

5. Estructuras de control de flujo

Según su función encontramos las siguientes:

- FOR...NEXT: Repite un bloque de instrucciones un número de veces fijado.
- IF...THEN...ELSE: A raíz de realizar una comparación lógica, ejecutan una o varias instrucciones.

6. Funciones

Son una especie de variables, definidas por el usuario o pertenecientes al lenguaje, que al llamarlas ejecutan un bloque de código (subprograma) con el objetivo de simplificar el código del programa en caso de invocarlas en varias ocasiones.

7. Comentarios

Trozos de código que no se ejecutan, así su función es la de comentar o aclarar partes del programa para facilitar su seguimiento.

4. Fases de elaboración de un programa informático

Una aplicación informática (o conjunto de programas) se desarrolla, como hemos comentado, con el objetivo de solucionar un determinado problema. Para ello se deberá tener en cuenta el llamado **ciclo de vida**, compuesto por las etapas que se deben seguir secuencial y ordenadamente. Dichas etapas son:

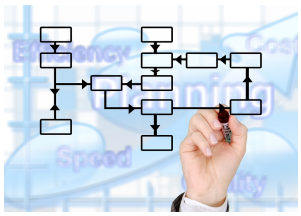

Imagen en pixabay de geralt bajo licencia CC

1. Análisis

Fase de especificación de cuáles son las necesidades que debe satisfacer nuestra aplicación y el planteamiento para cubrirlas.

2. Diseño

Una vez identificadas las necesidades, en la fase de diseño se deben detallar todos y cada uno de los elementos que usaremos: recursos físicos (característica del ordenador, periféricos, etc) y lógicos (sistema operativo, compilador, herramientas de utilidad, etc).

3. Codificación

Como se ha visto en una sección anterior, la tarea aquí es la de traducir a un lenguaje de programación la solución obtenida en la fase de diseño.

4. Explotación

Para explotar nuestra aplicación o programa se deberá implantar en el sistema informático que estemos usando, esto es, instalándola para posteriormente ejecutarla y comprobar los resultados que nos ofrece.

5. Mantenimiento

Esta última etapa está fundamentalmente relacionada con la idea de mejorar el programa desarrollado, corrigiendo errores y proporcionando ideas que lo hagan más eficiente. Así, en ocasiones puede dar lugar al reinicio del ciclo de vida.

Para saber más

A continuación te detallamos el **ejemplo** de programa informático ya citado para **comparar dos números enteros y devolver el mayor**. Hemos usado el software PSeInt que es bastante intuitivo.

Como aclaración, te presentamos el concepto de pseudocódigo como una metodología y sintaxis específica para el desarrollo de programas. No te preocupes si no entiendes algo, en tema siguiente trabajarás con dicho software a fondo. El resultado es el siguiente:

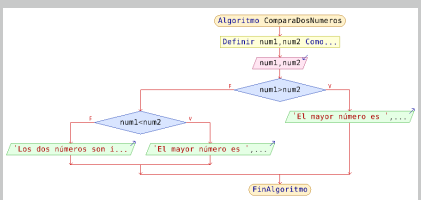

Imagen de creación propia bajo licencia CCO

Paso 1: Diagrama de flujo

Con dicho gráfico representamos el orden secuencial y posibilidades de ejecución del programa.

5. Estructura de un programa informático

Un programa informático (programa) es una **secuencia de acciones** (instrucciones) **que manipulan un conjunto de objetos** (datos).

Existen **dos** partes o **bloques** que componen un programa:

1. Bloque de **declaraciones**: en este se detallan todos los objetos que utiliza el programa (constantes, variables, archivos, etc).
2. Bloque de **instrucciones**: conjunto de acciones u operaciones que se han de llevar a cabo para conseguir los resultados esperados.

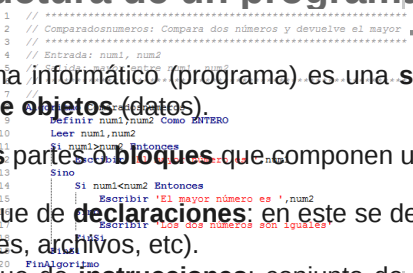

Imagen de creación propia bajo licencia CCO

Paso 2: Pseudocódigo

Diseño del programa en pseudocódigo que posteriormente tendremos que traducir a un lenguaje de programación concreto.

El bloque de instrucciones está compuesto a su vez por tres partes, aunque en ocasiones no están perfectamente delimitadas, y aparecerán entremezcladas en la secuencia del programa, podemos localizarlas según su función. Estas son:

1. **Entrada** de datos: instrucciones que almacenan en la memoria interna datos procedentes de un dispositivo externo.
2. **Proceso** o algoritmo: instrucciones que modifican los objetos de entrada y, en ocasiones, creando otros nuevos.
3. **Salida** de resultados:conjunto de instrucciones que toman los datos finales de la memoria interna y los envían a los dispositivos externos.

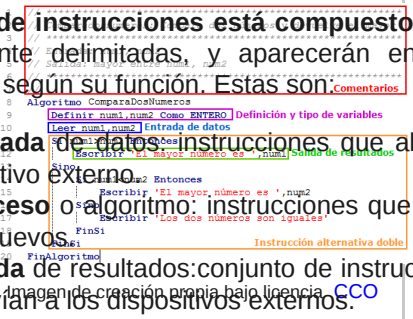

Imagen de creación propia bajo licencia CCO

Paso 2: Pseudocódigo (aclaración)

Explicación de cada bloque de pseudocódigo según su función.

Partes del bloque de instrucciones

Entrada	--> Algoritmo -->	Salida
Inicio de programa: datos	Proceso de programa: cálculos	Fin de programa: resultados

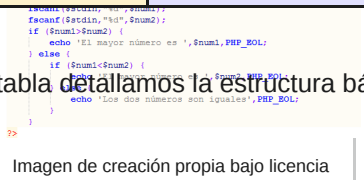

Imagen de creación propia bajo licencia

El siguiente paso es convertir el pseudocódigo general a un lenguaje de programación concreto, en este caso a lenguaje PHP.

En la siguiente tabla detallamos la estructura básica de un programa informático:

CCO**Estructura de un programa informático**

Cabecera	A modo de comentarios se suele especificar: • Nombre del programa • Datos de entrada • Datos de salida
Funciones	Definición de funciones propias creadas por el programador para usarlas en varias ocasiones
Declaraciones	Definiciones y tipos de: • variables • constantes • nuevos tipos de datos
Asignaciones	Valores iniciales de los identificadores declarados previamente
Entradas	Instrucciones para almacenar en memoria los valores de algunos identificadores
Control	Instrucciones de control de flujo del programa. Pueden ser: • Alternativas • Repetitivas

5 Salidas

Instrucciones para devolver los resultados obtenidos

Un programa informático está **compuesto por una serie de instrucciones**, cada una de ellas distinta según su función o cometido. Así, existen:

Imagen en pixabay de geralt bajo licencia CC

1. Instrucciones de definición de datos

Usando los llamados identificadores, nombramos cada variable o constante y el tipo de dato que contendrá. De esta forma al definirlos el programa podrá referenciarlas posteriormente.

2. Instrucciones primitivas

Son instrucciones primitivas las de asignación y las de entrada/salida.

3. Instrucciones compuestas

Constituidas por un conjunto de instrucciones o bloque, normalmente diseñado para una función específica.

4. Instrucciones de control

Controlan la ejecución de otras instrucciones en función de una comparación lógica.

- **Instrucciones alternativas:** En función de una condición (simple, doble o múltiple) controlan la ejecución de otras instrucciones.
- **Instrucciones repetitivas:**Sirven para ejecutar un bloque, o conjunto de acciones, un número determinado de veces. A este tipo de instrucciones se le conoce generalmente como bucles.

5.2. Elementos auxiliares

Son **variables que realizan funciones concretas** dentro de un programa, y por su gran utilidad, frecuencia de uso y peculiaridades, conviene hacer un estudio separado de las mismas. Las más importantes son:

Imagen en pixabay de Fotomek bajo licencia CC

1. Contadores: Un contador es un campo de memoria (variable) cuyo valor se incrementa o decrementa en una cantidad fija, generalmente asociado a una instrucción repetitiva (bucle). Le asignamos un valor inicial antes de comenzar su función, y cada vez que se realiza el suceso, incrementa su valor. Se utiliza:

- para contabilizar el número de veces que es necesario repetir una acción (variable de control de un bucle).
- para contar un suceso particular solicitado por el enunciado del problema (asociado a un bucle o independientemente).

2. Acumuladores: Un acumulador es una variable cuyo valor se incrementa sucesivas veces en cantidades variables. Se usa:

- en aquellos casos en que se desea obtener el total acumulado de un conjunto de cantidades, siendo necesario inicializarlo con el valor cero.
- también en las situaciones en que hay que obtener un total como producto de distintas cantidades, debiéndose inicializar con el valor 1.

3. Interruptores (switches): Un interruptor es un campo de memoria o variable que puede tomar dos valores exclusivos (0 y 1, -1 y 1, FALSO y CIERTO, etc.). Se utiliza para:

- recordar en un determinado lugar de un programa la ocurrencia o no de un suceso anterior, para salir de un bucle o para decidir en una instrucción alternativa qué acción realizar.
- para hacer que dos acciones distintas se ejecuten alternativamente dentro de un bucle.

6. Repasamos

Si has seguido todos los apartados de este tema, estás preparado para ponerte a prueba. Te proponemos dos actividades para que:

- compruebes tu nivel de comprensión de los conceptos expuestos
- practiques programando. ¿Te atreves?

Imagen en pixabay de OpenClipart-Vectors bajo licencia CC

Estos últimos apartados del tema, debes afrontarlos con entusiasmo y sobre todo aprendiendo de los errores. En los temas siguientes se explicarán con más detalle las instrucciones que podemos construir con los lenguajes de programación.

Te invitamos a repasar las secciones correspondientes y volver a contestar si fuera necesario.

Comprueba lo aprendido

Un **lenguaje de programación** es

- un conjunto de palabras y reglas para compilar correctamente un diagrama de flujo.
- un conjunto de símbolos y reglas para comunicarnos con el ordenador
- un conjunto de programas para traducir a código máquina el código fuente

Incorrecto

Opción correcta

Incorrecto

Solution

1. Incorrecto
2. Opción correcta
3. Incorrecto

Los lenguajes de programación pueden **clasificarse** en

- compilados e interpretados
- de alto, medio y bajo nivel
- orientados a objetos y orientados a lenguaje máquina

Opción correcta

Incorrecto

Solution

1. Opción correcta
2. Incorrecto
3. Incorrecto

El proceso de **traducción** consiste en

- convertir el código objeto a código fuente
- convertir el código fuente a código ensamblador
- convertir el pseudocódigo a código fuente

Incorrecto

Incorrecto

Opción correcta

Solution

1. Incorrecto
2. Incorrecto
3. Opción correcta

Algunos **tipos de datos** básicos son

- entero, carácter y fuente
- carácter y real
- entero y simultáneo

Incorrecto

Opción correcta

Incorrecto

Solution

1. Incorrecto
2. Opción correcta
3. Incorrecto

Para realizar **cálculos** matemáticos usamos

- los operadores lógicos
- los operadores relacionales

Incorrecto

Incorrecto

Opción correcta

Solution

1. Incorrecto
2. Incorrecto
3. Opción correcta

El operador **AND**

- es de tipo lógico
- es de tipo relacional
- es de tipo lógico aunque a veces se usa como operador aritmético

Opción correcta

Incorrecto

Incorrecto

Solution

1. Opción correcta
2. Incorrecto
3. Incorrecto

El **ciclo de vida** comprende una serie de etapas ordenadas de la siguiente forma

- análisis, diseño, codificación, explotación y mantenimiento
- diseño, análisis, codificación, explotación y mantenimiento
- análisis, explotación, codificación, diseño y mantenimiento

Opción correcta

Incorrecto

Incorrecto

2. Incorrecto
3. Incorrecto

El **bloque de instrucciones** de un programa está compuesto ordenadamente por

- entradas, salida y resultados
- entrada, proceso y salida
- entrada, salida y ejecución

Incorrecto

Opción correcta

Incorrecto

Solution

1. Incorrecto
2. Opción correcta
3. Incorrecto

Un lenguaje **multiplataforma**

- es aquel que se puede codificar a su vez en varios lenguajes
- es aquel que puede ser compilado e interpretado
- es aquel que se puede ejecutar en varios Sistemas Operativos

Incorrecto

Incorrecto

Opción correcta

Solution

1. Incorrecto
2. Incorrecto
3. Opción correcta

6.1 ¿Te atreves?

Queremos que practiques programando lo que has aprendido **¿Te atreves?**

En esta sección es importante que lo intentes, no mires la solución, antes vuelve a los contenidos y en todo caso, siempre intenta comprender el por qué de cada respuesta.

Ejercicio resuelto

¿Recuerdas las tres claves fundamentales para aprender a programar? Exacto: practicar, practicar y practicar. Es el momento de que empieces a llevarlas a cabo. ¡Ahora te toca a tí!

Imagen en pixabay de StartupStockPhotos bajo licencia CC

A continuación, te proponemos que resuelvas tres ejercicios que sin duda te ayudarán a poner en práctica los conceptos de este tema, serán la prueba de fuego y te introducirán en la tarea de la programación. No te preocupes si no consigues resolverlos completamente, el principal objetivo es comprender la metodología propuesta.

Es importante que tomes "cuaderno y bolígrafo" y hagas tus propuestas sobre el papel, te ayudará no sólo en cada paso individual, sino en la solución final en conjunto.

Ejercicio 1: Realiza el diseño de un programa que devuelva por pantalla una palabra entrada por teclado.

Mostrar retroalimentación

Los pasos a realizar son los siguientes:

- Paso 1: **Preguntar** al usuario la palabra deseada
- Paso 2: **Guardar** la palabra entrada en una variable, por ejemplo, palabra
- Paso 3: **Mostrar** la palabra (variable) por pantalla

Ejercicio 2: Realiza el diseño de un programa que permita sumar dos números y muestre el resultado.

Mostrar retroalimentación

Los pasos a realizar son los siguientes:

- Paso 1: **Mostrar** un mensaje al usuario indicando que entre el primer número
- Paso 2: **Guardar** la respuesta (primer número entrado) en una variable, por ejemplo num1

- Paso 5: **Sumar** los dos números y guardarlos en una variable, esto es, resultado <- num1 + num2
- Paso 6: **Mostrar** el resultado, es decir, el valor de la variable resultado

Ejercicio 3: Realiza el diseño de un programa que adivine un número entrado.

Mostrar retroalimentación

Los pasos a realizar son los siguientes:

- Paso 1: **Fijar** el número secreto que se debe adivinar asignando su valor a una variable, esto es, num_secreto <- 5
- Paso 2: **Mostrar** un mensaje al usuario indicando que entre un número
- Paso 3: **Guardar** el número entrado por el usuario en una variable, por ejemplo, num_entrado
- Paso 4: **Comparar** el valor de las variable num_secreto con el valor de la variable num_entrado y en función del resultado hacer:
  - Paso 4.1: Si num_secreto == num_entrado entonces mostrar el mensaje "Correcto"
  - Paso 4.2: Si num_secreto <> num_entrado entonces mostrar el mensaje "Incorrecto"

Resumen

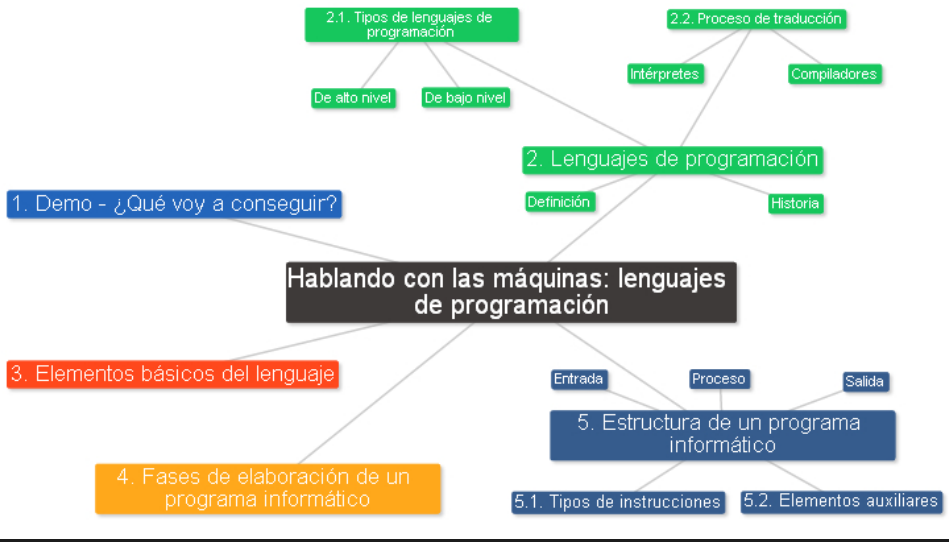

Imagen del autor bajo licencia Creative Common CCO

Descargar PDF

Aviso Legal

El presente texto (en adelante, el "**Aviso Legal**") regula el acceso y el uso de los contenidos desde los que se enlaza. La utilización de estos contenidos atribuye la condición de usuario del mismo (en adelante, el "**Usuario**") e implica la aceptación plena y sin reservas de todas y cada una de las disposiciones incluidas en este Aviso Legal publicado en el momento de acceso al sitio web. Tal y como se explica más adelante, la autoría de estos materiales corresponde a un trabajo de la **Comunidad Autónoma Andaluza, Consejería de Educación y Deporte (en adelante Consejería de Educación y Deporte**).

Con el fin de mejorar las prestaciones de los contenidos ofrecidos, la Consejería de Educación y Deporte se reserva el derecho, en cualquier momento, de forma unilateral y sin previa notificación al usuario, a modificar, ampliar o suspender temporalmente la presentación, configuración especificaciones técnicas y servicios del sitio web que da soporte a los contenidos educativos objeto del presente Aviso Legal. En consecuencia, se recomienda al Usuario que lea atentamente el presente Aviso Legal en el momento que acceda al referido sitio web, ya que dicho Aviso puede ser modificado en cualquier momento, de conformidad con lo expuesto anteriormente.

Régimen de Propiedad Intelectual e Industrial sobre los contenidos del sitio web.

Imagen corporativa. Todas las marcas, logotipos o signos distintivos de cualquier clase relacionados con la imagen corporativa de la Consejería de Educación y Deporte que ofrece e