**ENTORNOS DE DESARROLLO**

A la hora de programar es conveniente (si no necesario) realizar un esquema del flujo del programa que vamos a hacer. Este esquema se llama algoritmo. Tiene una serie de símbolos con los que estructurar todo el flujo:

INPUT

OUTPUT

4

Cambiador de Hoja

TOMA DE DECISIONES

PROCEDIMIENTO

Similar a función en Python

POSIBILIDAD MULTIPLE

FUNCIÓN

ALMACENAMIENTO EN DISCO

**Ejemplo 1:**

Diagrama de un programa que lea dos números, los sume e imprima el resultado:

INPUT A,B

EL RESULTADO ES Suma

SUMA A+B

Pseudo código del mismo programa:

Inicio Programa Suma Dos Números

Entorno

Variables 🡪 Suma,A,B : Entero

Inicio Procesamiento

Suma 🡨 A+B

Imprime “El resultado es”, Suma

Fin Procesamiento

Fin Programa

**Ejemplo 2:**

Dados el radio y la altura de un cilindro, calcule: Área total y volumen.

Atotal = 2 · π · r · h + 2 · π · r2

Vcilindro = 2 · π · r · h

A

V

INPUT R, H

2 · π · r · h + 2 · π · r2

2 · π · r · h

**Ejemplo 3:**

Dadas las tres dimensiones de un bloque rectangular, calcule y muestre su área total y su volumen.

Área = 2 · (a·h + a·p + h·p)

Volumen = a · p · h

Si queremos preguntar al principio si

A

V

es un rectángulo o un cubo,

pondremos un rombo que pregunte

INPUT A, H, P

si las aristas son iguales. En caso de

serlo, seguimos la ejecución.

Si no lo es, hacemos las fórmulas del

cubo.

A 🡨 2 · (a·h + a·p + h·p)

V 🡨 a · p · h

**Ejemplo 4:**

Hacer un algoritmo que lea los KW consumidos por una familia, y el precio por KW. Si la cantidad de KW consumida es mayor que 700, incrementamos el precio en un 5% para el exceso de KW sobre 700. Mostrar el valor total a pagar.

Pseudo código:

1.- Leer KW

2.- Leer precioKW

3.- Total = KW · precioKW

3.- Si KW > 700 Exceso = KW – 700

Total += Exceso · 0,005

4.- En otro caso devolver Total

Diagrama:

NO SI

700 \* PRECIO + KW -700 \* (PRECIO + PRECIO \* 5/100)

KW \* PRECIO

CANTIDADKW

PRECIOKW

RESULTADO

**LENGUAJES DE ALTO NIVEL**

Son lenguajes cercanos al habla común del ser humano. Al contrario que los lenguajes de bajo nivel, son lenguajes fácilmente entendibles por las personas.

Se crean por la dificultad existente al usar código ensamblador/máquina.

Los primeros que surgieron fueron *FORTRAN* (ámbito científico y académico)y *COBOL* (ámbito de gestión empresarial). En el ámbito educativo y doméstico fue *BASIC*.

Uno de los lenguajes que más ha marcado al resto ha sido C (y sus sucesores C++ y C#). Se usó para escribir el SO Unix (del cual nació posteriormente Linux). Windows está escrito en C++.

**PROCESO DE TRADUCCIÓN (INTÉRPRETES VS COMPILADORES)**

**Intérpretes:** Son programas que se encargan de ir ejecutando las instrucciones del lenguaje de alto nivel sin traducirlas literalmente a código máquina. El propio intérprete está en código máquina.

**Compiladores:** Traduce las instrucciones en lenguaje de alto nivel a código máquina. Hacen optimizaciones para que el código vaya más rápido o para que ocupe menos espacio.

Hay lenguajes ***híbridos***, que permiten ejecutar programas en cualquier IDE sin tener que volver a compilarlo. JAVA es un ejemplo de lenguaje híbrido, porque teniendo una máquina virtual de JAVA en la máquina, se puede ejecutar cualquier aplicación en este lenguaje sin volver a compilarlo.

**TIPOS DE SOFTWARE**

**Software del Sistema:** Libera al usuario del hardware y su gestión. *Ejemplos: Sistema Operativo, drivers, utilidades del SO.*

**Herramientas de programación:** Permiten al usuario crear sus propias aplicaciones. *Ejemplos: Editores de texto, intérpretes, compiladores, IDE.*

**Software de aplicación:** Es el que el usuario usa para realizar ciertas tareas. *Ejemplos: Hojas de cálculo, juegos.*

**Programa:** Serie de órdenes escritas en un lenguaje de programación con una finalidad concreta.

**Aplicación:** Uno o varios programas que permiten a usuario realizar una tarea en concreto.

**Software a medida:** Software creado con un propósito concreto.

**Software estándar:** Software creado sin un cliente específico.

**Software ERP personalizables:** Se usan para mantener una gestión interna adecuada en una empresa. Se ajustan a las necesidades de cada empresa en concreto.

**PARADIGMAS DE PROGRAMACIÓN**

* **Programación imperativa:** Es la que se usa hoy en día, es programar por instrucciones (tipo receta). *C++ en versiones antiguas*.
* **Programación estructurada:** Es la alternativa que se desarrolló durante la crisis del software. Se basa en la construcción secuencial, la estructura iterativa, y la estructura condicional. Fue desarrollada para evitar el *código espagueti*, que se daba por el exceso del uso de la función *GO TO*. *Algol 60, PL/I, Algol 68, Pascal, C, Ada 83, Modula, Modula-2*.
* **Programación orientada a objetos:** Se basa en modelizar los programas con elementos del mundo. Se crea una clase (*coche*) y se van a crear objetos pertenecientes a estas clases, con sus mismas características. *Java, Python, Ruby, C++*.
* **Programación declarativa:** No se expresan los pasos que hay que seguir para resolver un problema, directamente se le dice que lo resuelva. Se consigue mediante un *motor*. *SQL*.
* **Programación funcional:** Define todos los elementos del programa como si fueran funciones (matemáticas). Ante unos datos, siempre va a devolver la misma respuesta. Evita el almacenamiento de estados (*variables*). *Scheme, Haskell, Miranda, ML*.
* **Programación lógica:** Es una variante del paradigma declarativo. Se usan predicados lógicos, *ejemplo: Para saber que uno es abuelo de tres, hay que saber que el hijo de uno es padre de tres. Prolog, GHC, Parlog, Vulcan, Polka, Mercury, Fnil*.

**LENGUAJES DE PROGRAMACIÓN ACTUALES**

En el ranking *TIOBE Index* se pueden ver los lenguajes de programación más utilizados.

**Java:** Es el lenguaje de programación más usado del mundo, es un lenguaje orientado a objetos y se ejecuta en cualquier plataforma ya que se ejecuta en la *JVM* (*Java Virtual Machine*). Permite el *multi-threading*. *IDE: Eclipse*

**Python:** Es un lenguaje interpretado que no se compila, se puede ejecutar en muchas plataformas. Es un lenguaje con posibilidad de ser orientado a objetos, aunque con algo de programación procedural. Está escrito en C, por lo que se puede incrustar en C++. Esto lo usa *Sublime Text*, que aunque está escrito en C++ se pueden instalar extensiones en *Python*. *IDE: Pycharm*

**C#:** Se ejecuta en una máquina virtual como Java, pero esta vez se llama *CLR*. Es un lenguaje de la plataforma *.NET* y su lenguaje deriva de *C++*. Es un lenguaje orientado a objetos. Puede ser compilado pura e híbridamente (*bytecodes*). *IDE: Visual Studio*

**C++:** Se pensó como una extensión orientada a objetos de C, aunque hoy en día son muy diferentes. Es un lenguaje *multiparadigma* (permite programar de forma procedural, orientada a objetos, etc). Hace uso de *punteros/direcciones* *de memoria* para trabajar a bajo nivel como en lenguaje ensamblador. Se usa mucho para escribir SO. No son tan portables. *IDE: C++ Builder*

**JavaScript:** Se ejecuta en el lado del cliente, aunque tiene extensiones como *NodeJS* que se ejecutan en el lado del servidor. Es un lenguaje muy usado, aunque no gusta entre la comunidad de programadores web. *IDE: WebStorm*

**PHP:** Es multiplataforma, aunque la mayoría de servidores hoy en día son Apache, que corre sobre Linux. Es un lenguaje interpretado. *IDE: NetBeans*

**VisualBasic .NET:** Nació después que Basic, y hoy en día son muy distintos. *IDE: Visual Studio*

**Ruby:** Es un lenguaje interpretado. *IDE: NetBeans*

**Ejercicio 12.-** Describir el sistema de ejecución y compilación de *.NET*.

Se escribe el programa en el lenguaje de programación deseado, pero se compila en un lenguaje intermedio.

**Ejercicio 13.-** Describir LLVM. Ajustar sus herramientas a la estructura de un compilador. Buscar relación entre LLVM y WebAssembly.

LLVM significa Low Level Virtual Machine. Es una estructura para desarrollar compiladores. LLVM es el lenguaje utilizado por WebAssembly para compilar los programas multiplataformas.

**Ejercicio 14.-** ¿En qué máquinas virtuales se usan JIT Compilers?

En la JVM.

**INGENIERÍA DEL SOFTWARE**

Es el establecimiento de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia.