0 - Introducción

Hoy en día, casi todo lo que nos rodea está relacionado con la programación, los programas y el tratamiento de información.

> *Una receta de cocina es como un programa, una secuencia de pasos con la finalidad de preparar un plato.*

Haremos un recorrido por los conceptos fundamentales y la programación: las técnicas que podemos emplear, las diferentes formas de programación existentes, las fases del desarrollo de un programa…

Los programas pretenden dar solución a un determinado problema. Y tal como intentamos resolver los problemas que nos surgen en el día a día nos podemos plantear la resolución mediante técnicas de programación.

Vida real	**Programación...**
Observación del problema.	**Análisis del problema:** definir el problema y comprenderlo.
Buscar posibles soluciones.	**Diseño o desarrollo de algoritmos:** procedimiento paso a paso para solucionar el problema.
Aplicar solución más adecuada.	**Resolución del algoritmo elegido:** consiste en convertir el algoritmo en programa, ejecutarlo y verificar que soluciona el problema.

Algoritmos y programas

Tras analizar el problema, tendremos que diseñar y desarrollar el algoritmo adecuado. Un **algoritmo** es una secuencia ordenada de pasos, descrita sin ambigüedades, que conducen a la solución de un problema.

El algoritmo es independiente del lenguaje de programación, y podrá ser expresado en cualquiera de ellos. Así obtendremos un **programa,** que podrá ser ejecutado en cualquier ordenador.

Si los problemas son complejos, es necesario descomponer éstos en problemas más simples y estos, en otros más pequeños. **divide y vencerás:**

- **diseño descendente**. Consistente en la descomposición del problema en problemas más sencillos de resolver.
- **diseño modular** (**top-down design**). Que consiste en dividir la solución a un problema en módulos más pequeños o subprogramas. Las soluciones de los módulos se unirán para obtener la solución general del problema.

Podemos representar gráficamente los algoritmos que vamos a diseñar, para ello disponemos de diferentes herramientas:

- **Diagramas de flujo**: Esta técnica utiliza símbolos gráficos para la representación del algoritmo. Suele utilizarse en las fases de análisis.
- **Pseudocódigo**: Esta técnica se basa en el uso de palabras clave mezcladas con lenguaje natural
- **Tablas de decisión**: Sirven para evaluar las posibles decisiones a tomar para resolver el problema. Cobre todo con condiciones complejas.

0 - Introducción

Paradigmas de programación

El paradigma representa un enfoque o filosofía para la construcción de software.

Programación declarativa

Desarrollo basado en la “declaración” de un conjunto de condiciones, proposiciones, afirmaciones, restricciones y transformaciones que describen el problema y detallan su solución.

SQL es un lenguaje baso en este paradigma

Programación funcional

La programación funcional es un paradigma declarativo. Nos enfocaremos en "qué" estamos haciendo y no en "cómo" se está haciendo que sería el enfoque imperativo. Esto quiere decir que nosotros expresaremos nuestra lógica sin describir controles de flujo; no usaremos ciclos o condicionales.

El lenguaje LISP es un ejemplo

Programación lógica

Se especifica que hacer y no como hacerlo. Es utilizado en inteligencia artificial

Un ejemplo es el lenguaje PROLOG

Programación imperativa

Consiste en una serie de comandos que un ordenador ejecutará. Utiliza variables, tipos de datos, expresiones y estructuras de control de flujo

Programación convencional

Conocida como “no estructurada”. Las líneas de programa se ejecutaban sucesivamente, usando saltos incondicionales basadas en (GOTO) para modificar el flujo del programa

Programación estructurada

Permite utilizar estructuras que facilitan la modificación, reutilización y lectura del código.

Permite el uso de funciones, que pueden ser utilizadas tantas veces como se necesiten.

Estructuras condicionales, bucles, etc.

Programación orientada a objetos

El mundo se ve desde el punto de vista de los objetos que hay en él. Los objetos tienen características (propiedades) y con ellos se pueden realizar acciones (métodos).

Incluye herencia, abstracción, polimorfismo y encapsulamiento.

Ejemplo son Java, C++, PHp

Visual, orientada a aspectos, a eventos, …

Son un conjunto de paradigmas que pueden ser relevantes, aunque gran parte de ellos incluyen los conceptos del resto de los paradigmas

0 - Introducción

Fases de la programación

1- Resolución del problema

- **Análisis**.

  En esta fase debemos aprender a analizar la documentación disponible, investigar, observar todo lo que rodea el problema y recopilar cualquier información útil.

- **Diseño.**

  Realizar un análisis más detallado, indicando la lógica del proceso, o lo que es lo mismo, el algoritmo que resolverá el problema.

  Este algoritmo debe ser probado antes de su implementación, para ello haremos un seguimiento paso a paso del proceso con valores reales, verificando que el resultado obtenido es correcto.

2- Implementación

- **Codificación**

  Esta etapa consiste en transformar o traducir los resultados obtenidos a un determinado lenguaje de programación.

  Cuando realizamos la traducción del algoritmo al lenguaje de programación debemos tener en cuenta las reglas gramaticales y la sintaxis de dicho lenguaje. Obtendremos entonces el código fuente, lo que normalmente conocemos por ***programa***.

- **Traducción a lenguaje máquina**

  Este proceso de traducción puede hacerse de dos formas, compilando o interpretando el código del programa.

  Compilación: Es el proceso por el cual se traducen las instrucciones escritas en un determinado lenguaje de programación a lenguaje que la máquina es capaz de interpretar.

  Intérprete: sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.

3- Pruebas de ejecución, validación y documentación

Implantar la aplicación en el sistema donde va a funcionar y comprobar si su funcionamiento es correcto. Utilizando diferentes datos de prueba se verá si el programa responde a los requerimientos especificados, si se detectan nuevos errores, si éstos son bien gestionados y si la interfaz es amigable.

Documentación interna: Encabezados, descripciones, declaraciones del problema y comentarios que se incluyen dentro del código fuente.

Documentación externa: Son los manuales que se crean para una mejor ejecución y utilización del programa.

4- Explotación.

Cuando el programa ya está instalado en el sistema y está siendo de utilidad para los usuarios, decimos que se encuentra en fase de explotación.

Mantenimiento del software: es el proceso de mejora y optimización del software después de su entrega al usuario final. Corregir defectos y adición de nuevas funcionalidades para mejorar la usabilidad y aplicabilidad del software.

Lenguajes de programación

El lenguaje de programación es un conjunto de reglas sintácticas y semánticas, símbolos y palabras especiales.

- **Gramática del lenguaje**: Reglas aplicables al conjunto de símbolos y palabras especiales.
- **Léxico**: Es el conjunto finito de símbolos y palabras especiales, el vocabulario del lenguaje.
- **Sintaxis**: Son las posibles combinaciones de los símbolos y palabras especiales.
- **Semántica**: Es el significado de cada construcción del lenguaje, la acción que se llevará a cabo.

Los lenguajes de programación pueden ser clasificados en función de lo cerca que estén del lenguaje humano (algo nivel) o del lenguaje de los computadores (bajo nivel).

1- Lenguaje máquina.

Este es el lenguaje utilizado directamente por el procesador, consta de un conjunto de instrucciones codificadas en binario.

Programar en este tipo de lenguaje presentaba los siguientes inconvenientes:

- Cada programa era válido sólo para un tipo de procesador u ordenador.
- La lectura o interpretación de los programas era extremadamente difícil y, por tanto, hacer modificaciones era muy costoso.
- Obligaba memorizar las instrucciones disponibles para cada tipo de procesador.
- Introducir los códigos binarios en el computador, conllevaba tiempo de preparación y posibles errores.

2- Lenguaje Ensamblador.

La evolución del lenguaje máquina fue el lenguaje ensamblador. Las instrucciones ya no son secuencias binarias, se sustituyen por códigos de operación que describen una operación elemental del procesador. Es un lenguaje de bajo nivel, al igual que el lenguaje máquina, ya que dependen directamente del hardware donde son ejecutados.

3- Lenguajes compilados.

Se acercan a un lenguaje más cercano al humano que al del computador. Algunos ejemplos de este tipo de lenguajes son: Pascal, Fortran, C, etc.

Entre sus ventajas están:

- Son mucho más fáciles de aprender y de utilizar que sus predecesores.
- Se reduce el tiempo para desarrollar programas, así como los costes.
- Son independientes del hardware, los programas pueden ejecutarse en diferentes tipos de máquina.
- La lectura, interpretación y modificación de los programas es mucho más sencilla.

El proceso de compilación realizará la traducción y además informará de los posibles errores. Se generará el programa traducido a código máquina, conocido como código objeto.

4- Lenguajes interpretados.

Cada instrucción escrita en un lenguaje interpretado se analiza, traduce y ejecuta tras haber sido verificada. El proceso de traducción y de ejecución se llevan a cabo simultáneamente, no se genera programa objeto, ni programa ejecutable.

Ejemplos de lenguajes interpretados son: Perl, PHP, Python, JavaScript, etc.

5- Lenguajes precompilados.

Es el caso de Java, que es compilado a un lenguaje intermedio llamado **bytecode**, que después debe ser interpretado. Con lo que se consigue un programa que se puede ejecutar en cualquier sistema operativo y procesador sin necesidad de crear varios ejecutables.

La Máquina Virtual Java se encargará de interpretar el **bytecode** y lo traducirá a código máquina del procesador en particular sobre el que se esté trabajando.