PROGRAMACIÓN

UT1: INTRODUCCIÓN A LA PROGRAMACIÓN

1. Conceptos básicos.

¿Qué es la programación?

Es el instrumento que permite la ejecución de tareas automatizadas de un sistema informático.

Trata de imitar el pensamiento humano, ya que está realizada por este, ejecutando unas u otras acciones en base un razonamiento lógico a partir de unas premisas.

¿Qué es un programa?

Serie de órdenes o instrucciones ordenadas con una finalidad concreta que realizan una función determinada.

Algoritmo: Procedimiento paso a paso para resolver un problema en una cantidad finita de pasos.

Instrucciones: regla, norma, o cada paso de un algoritmo. Podemos relacionarlo con cada una de las líneas de nuestro programa.

Datos: Elemento considerado como unidad de tratamiento dentro de un sistema de proceso de datos. Pueden ser de entrada o de salida.

Por tanto, podemos definir un programa como la suma de datos más instrucciones que siguen unos algoritmos para obtener la solución.

2. Tipos de lenguajes de programación

¿Qué es un lenguaje de programación?

Conjunto de notaciones, símbolos y reglas sintácticas para posibilitar la escritura de un algoritmo, que posteriormente será interpretado por el hardware de un ordenador.

Lenguaje de alto nivel: Sus características se encuentran más cercanas al lenguaje natural, haciendo más sencilla la elaboración del código.

Son independientes de la arquitectura de la máquina, por lo que son portables.

Lenguaje de bajo nivel: Sus características se encuentran más cercanas a la arquitectura de la máquina que al lenguaje natural.

Son totalmente dependientes de la arquitectura de la máquina.

Podemos diferenciar el lenguaje ensamblador y el lenguaje máquina.

Los ordenadores hablan un único lenguaje, el lenguaje máquina (o código binario), es decir, solo entiendes señales o impulsos eléctricos (1 cuando existe este impulso o señal y 0 cuando no).

Para poder “hablar” con el ordenador y que entienda nuestras órdenes, será necesario usar un traductor entre el lenguaje que nosotros sabemos (alto nivel) y el lenguaje máquina.

3. Proceso de traducción

El sistema o programa encargado de realizar el proceso de traducción entre un lenguaje y el lenguaje máquina recibe el nombre de Traductor de lenguaje, que pueden ser de tres tipos,

Ensambladores: Encargados de traducir los programas escritos en lenguaje ensamblador en lenguaje máquina.

Compiladores: Programa que lee el código escrito en un lenguaje y lo traduce en un programa equivalente en otro lenguaje.

El compilador hace notar al usuario los fallos de programación que puedan existir para poder “compilar” el programa correctamente.

Intérprete: en lugar de producir un nuevo programa totalmente traducido al lenguaje máquina, los intérpretes van traduciendo y ejecutando de manera simultánea cada instrucción.

¿Cuáles son las diferencias entre intérprete y compilador?

4. Paradigmas de la programación

Existen diferentes formas de resolver un problema, derivándose así diferentes tipos de programación según la forma de resolverlo, o también llamado, paradigma de la programación.

Principalmente podemos dividir la programación en programación **imperativa** y programación **declarativa.**

La programación imperativa o por procedimientos es la más común, estos programas son un conjunto de instrucciones que le indican a la máquina cómo debe realizar una tarea.

4.1 Programación imperativa

Dentro del paradigma imperativo podemos destacar 3 tipos de programación:

- Programación modular

Las secuencias de código con una funcionalidad común son agrupadas en módulos separados. Su fundamento se basa en la descomposición sucesiva del problema inicial.

Esto genera la facilidad de escritura, lectura y comprensión de los programas y el ahorro de espacio debido al código repetido.

- Programación estructurada

Permite la escritura de programas fáciles de leer y modificar. Utiliza un número limitado de estructuras de control que minimizan la complejidad de los problemas y los errores. Incorporan el diseño descendente, recursos abstractos y estructuras básicas.

Deben tener solo un punto de entrada y uno de salida.

Existir de 1 a N caminos desde el inicio hasta el fin que pasen por todos los puntos.

Todas las instrucciones sean ejecutables sin bucles infinitos.

- Programación orientada a objetos

Su principal ventaja es el aumento de la velocidad en el desarrollo, ya que se pueden incorporar o reutilizar “objetos” con las características que necesitemos utilizados anteriormente o de otras personas.

Elimina el problema de la portabilidad y de la vida de los programas, al ser reutilizables.

4.2 Programación declarativa

Por otro lado, en la programación declarativa los programas son desarrollados declarando un conjunto de condiciones, proposiciones, afirmaciones, restricciones o ecuaciones que describen el problema y detallan la solución. Esta solución es obtenida mediante mecanismos internos de control, pero sin especificar exactamente cómo encontrarla. Es decir, describe qué se debe encontrar, pero sin definir cómo se debe hacer.

Dentro del paradigma declarativo podemos encontrar:

- Programación lógica

Considera los programas como una serie de aserciones lógicas, representando el conocimiento como “reglas”.

En estos el conocimiento está especificado, pero no se indica cómo debe ser usado.

Son usados principalmente para la inteligencia artificial.

- Programación funcional

Basado en la utilización de funciones que no manejan datos mutables o de estado. Usados principalmente en matemáticas, estadística y análisis financiero.

5. Fases de elaboración de un programa (Ciclo de vida)

La creación de un programa requiere de tres fases:

5.1 Planificación y análisis

En esta fase se intenta caracterizar el sistema que se va a construir. Hay que determinar la información que va a usar el sistema, las funciones que va a realizar, identificar las condiciones que existen... conseguir la mayor información posible.

Si no se llega a una definición clara de que se busca o necesita no se puede encontrar una solución al problema.

En esta fase se define cuál es la situación de partida, los datos de entrada, los resultados deseados y cuál será la situación final a la que debe conducir el problema una vez implementado.

5.2 Desarrollo

En esta fase se diseñan estructuras de datos y de los programas, se escriben y documentan éstos, y se prueba el software.

Los analistas deberán trabajar con los requerimientos del software desarrollados en la etapa de análisis. Se determinan todas las tareas que cada programa realiza, como así la forma en que se organizan estas tareas cuando se codifique el programa. Los problemas cuando son complejos se pueden resolver más eficientemente cuando se descomponen en subproblemas que sean más fáciles de solucionar que el original.

La descomposición del problema original en subproblemas más simples y a continuación dividir estos subproblemas en otros más simples que pueden ser implementados para su solución se denomina diseño descendente.

Durante esta etapa se utilizan auxiliares de diseño, que son diagramas y tablas que facilitan la delineación de las tareas o pasos que seguirá el programa, por ejemplo: diagramas de flujo, pseudocódigo, etc.

Finalmente se convierte el algoritmo en programa, escrito en un lenguaje de programación como C, Java, etc.

Las pruebas de software son parte esencial del proceso de desarrollo del software. Esta parte del proceso tiene la función de detectar los errores de software lo antes posible.

5.3 Mantenimiento

El despliegue comienza cuando el código ha sido suficientemente probado, ha sido aprobado para su liberación y ha sido distribuido en el entorno de producción.

El mantenimiento o mejora de un software con problemas recientemente desplegado, puede requerir más tiempo que el desarrollo inicial del software. Es posible que haya que incorporar código que no se ajusta al diseño original con el objetivo de solucionar un problema o ampliar la funcionalidad para un cliente. Si los costes de mantenimiento son muy elevados puede que sea oportuno rediseñar el sistema para poder contener los costes de mantenimiento.

6. Documentación

Una de las partes más importantes de programar (y la que menos gusta hacer) es documentar correctamente tanto el código como la aplicación.

Podemos dividirlo en dos categorías.

- Documentación interna.

Aquella que se introduce dentro del código para explicar el funcionamiento de este, hojas de texto donde se disponen el correcto nombre de las variables a usar, … principalmente es documentación para los trabajadores de la aplicación, en caso de tener que consultar cualquier duda, arreglar un fallo o actualizar la aplicación esto sea fácilmente resoluble.

- Documentación externa.

Escritos para el correcto uso de la aplicación. Están dedicados a personas que no hayan trabajado en la elaboración de la aplicación o programa, pero que sí vayan a trabajar con este. Normalmente nos referimos a los manuales de usuario o a los casos de uso.