INTRODUCCIÓN

A lo largo de la historia, el hombre ha necesitado transmitir y tratar información de forma continua. La humanidad no ha parado de crear máquinas y métodos para procesar la información.

La informática congrega a muchas de las técnicas que el hombre ha desarrollado con el fin de potenciar sus capacidades de pensamiento, memoria y comunicación. Se utiliza en múltiples áreas: en gestión de negocios, almacenamiento de información, control de procesos, comunicaciones, transportes, medicina y en muchos otros sectores.

En la informática convergen los fundamentos de las ciencias de la computación, la programación y metodologías para el desarrollo de software, la arquitectura de computadores, las redes de datos (como Internet), la inteligencia artificial y ciertas cuestiones relacionadas con la electrónica. Se puede entender por informática a la unión de todo este conjunto de disciplinas.

Conceptos básicos

La informática.

La Informática es la ciencia que estudia y se ocupa del análisis y resolución de problemas a través de la computadora. El tratamiento de la información consta de tres fases: la entrada de datos, el procesamiento de dichos datos y la salida de los resultados.

Desde el punto de vista de la informática el elemento físico utilizado para el tratamiento de la información es la computadora que se define de la siguiente forma:

Una computadora u ordenador es un aparato electrónico que tiene el fin de recibir y procesar datos para la realización de diversas operaciones siempre que se le den las instrucciones adecuadas.

Están compuestos de dos partes fundamentales:

Hardware

Parte física del computador, comprende todos los dispositivos periféricos físicos o materiales que componen el computador.

Software

Parte lógica del computador, compuesta por el conjunto de instrucciones o código programado que permitiendo realizar tareas específicas y generales.

Una computadora solo es capaz de ejecutar órdenes y operaciones muy básicas, tales como: sumar, restar, multiplicar, etc. Comparar valores numéricos o alfanuméricos y almacenar o recuperar información.

Combinando estas operaciones, y gracias a su gran potencia de cálculo, la computadora puede llevar a cabo procesos muy complejos. Sin embargo, es el programador quien indica a la máquina cómo y qué debe hacer, a través de la lógica y el razonamiento previo, que luego se expresa a través de lo que denominamos "programa".

El tratamiento de la información consta de tres fases: la entrada de datos, el procesamiento de dichos datos y la salida de los resultados, es decir que el ordenador solo es capaz de aceptar datos de entrada, procesarlos y facilitar otros datos o resultados de salida.

La resolución de problemas, utilizando como herramienta una computadora, requiere contar con la capacidad de expresión suficiente como para indicar a la máquina lo que debe llevarse a cabo.

Resolución de problemas con computadora

La resolución de problemas utilizando una computadora no se resume únicamente en la escritura de un programa, sino que, es un proceso que va desde interpretar las necesidades del usuario hasta corroborar que la respuesta obtenida es correcta.

Las etapas son las siguientes:

- Análisis del problema
- Diseño del algoritmo
- Codificación
- Compilación y ejecución
- Verificación
- Depuración
- Documentación

Análisis del problema

En esta primera fase o etapa, requiere de una clara definición, especificando exactamente lo que debe hacer el programa y el resultado o solución deseada. Deben obtenerse los requerimientos del usuario. Se debe ser preciso en las especificaciones de entrada y salida.

Para poder definir correctamente un problema es útil responder a las siguientes preguntas:

- ¿Qué entradas se requieren?
- ¿Cuál es la salida deseada?
- ¿Qué método produce dicha salida?

Diseño del algoritmo

Una vez bien definido el problema que determina *que* debe hacer el programa se procede a la siguiente etapa que determina *cómo* debe hacer el programa la tarea solicitada.

Los métodos más eficaces para el proceso de diseño se basan en el conocido lema de "divide y vencerás". Es decir que la resolución de un problema consiste en la identificación de las partes (subproblemas) que componen todo el problema y la manera en que se relacionan. Cada uno de estos subproblemas debe tener un objetivo específico, es decir, debe resolver una parte del problema original. La integración de los subproblemas es lo que permitirá obtener la solución buscada.

Cada subproblema es resuelto mediante un módulo (subprograma) que tiene un solo punto de entrada y un solo punto de salida. Dichos módulos pueden ser planeados, codificados, comprobados y depurados independientemente y a continuación combinarlos entre sí. El proceso implica la ejecución de los siguientes pasos:

- Programar un módulo.
- Comprobar el módulo.
- Si es necesario, depurarlo.
- Combinar el módulo con los módulos anteriores.

El diseño del algoritmo es independiente del lenguaje de programación en el que se vaya a codificar posteriormente.

Codificación

Es la escritura en un lenguaje de programación de la representación del algoritmo desarrollado en las etapas precedentes.

Un algoritmo es una especificación simbólica que debe convertirse en un programa real sobre un lenguaje de programación concreto. A su vez, un programa escrito en un lenguaje de programación determinado es traducido automáticamente al lenguaje de máquina de la computadora que lo va a ejecutar.

Compilación y ejecución

Una vez que el algoritmo se ha convertido en un programa fuente, es preciso introducirlo en memoria y almacenarlo en un disco. Esta operación se realiza con un programa editor, posteriormente el programa fuente se convierte en un archivo de programa y se graba en disco.

El programa Fuente (código fuente) es un texto o conjunto de líneas de texto (líneas de código) que forman parte esencial de un programa informático, estas líneas luego deben ser traducidas a un lenguaje de maquina tarea de la cual se encarga el compilador.

Si tras la compilación se presentan errores, es preciso volver a editar el programa, corregir errores y compilar nuevamente, proceso que se repite hasta que no se produce ningún error y luego se obtiene el programa objeto.

El programa objeto (código objeto) es justamente el responsable de que el ordenador pueda interpretar las acciones que se ordenan y los comandos para poder ser ejecutados, interpretados y re-transmitidos por los componentes físicos de un ordenador.

Verificación y depuración

La verificación consiste en que una vez que se tiene un programa escrito en un lenguaje de programación se debe verificar que su ejecución produce el resultado deseado, utilizando una amplia variedad de datos representativos llamados datos de test o prueba, que determinarán si el programa tiene o no errores.

Para la verificación se debe desarrollar una amplia gama de datos de test: desde valores normales de entrada , valores extremos y demás valores que comprueben aspectos especiales de un determinado programa.

La facilidad de verificación y la depuración de errores de funcionamiento del programa conducen a una mejor calidad del sistema y es un objetivo central de la Ingeniería de Software.

La depuración es el proceso de encontrar los errores del programa y corregir o eliminar dichos errores.

Se pueden producir tres tipos de errores:

- Errores de compilación

Se producen generalmente por un uso incorrecto de las reglas del lenguaje de programación y suelen ser errores de sintaxis. Frente a un error de sintaxis, la computadora no puede comprender la instrucción y no se obtendrá el programa objeto. Normalmente el compilador mostrará una lista de todos los errores encontrados.

- Errores de ejecución

Se producen por instrucciones que la computadora puede comprender, pero no ejecutar, suelen provenir de la realización de operaciones no permitidas y pueden ocurrir o no dependiendo de los datos de entrada empleados por ello es la necesidad de probar el programa con un conjunto amplio de datos que abarque la mayoría de los casos. Un ejemplo muy común es la división por cero. En estos casos se detiene la ejecución del programa y arroja un mensaje de error.

- Errores lógicos

Se dice que un programa tiene un error de lógica si produce resultados que no son correctos. Este tipo de errores son los más difíciles de detectar, ya que el programa puede

funcionar y no producir errores de compilación ni ejecución, solo puede advertir el error por la obtención de resultados incorrectos. En este caso es necesario volver a la fase de diseño.

Documentación

La documentación consta de las descripciones de los pasos a dar en el proceso de resolución de un problema. es muy importante, pues considera que no siempre vas a estar tu y tu equipo, por eso será indispensable que haya una documentación sobre la cual un nuevo equipo se pueda basar.

Cabe mencionar que programas pobremente documentados son difíciles de leer, más difíciles de depurar y casi imposibles de mantener y modificar.

La documentación de un programa puede ser interna y externa. Se denomina documentación interna al contenido del propio programa fuente, son la líneas de comentarios que deben ser los suficientes para posibilitar la comprensión del mismo. La documentación externa incluye, análisis, diagramas de flujo y/o pseudocódigo, manuales de usuario, descripción d eversiones, etc.

La documentación es vital cuando se desea corregir posibles errores futuros o bien cambiar el programa. Tales cambios son los que denominamos mantenimiento del programa, y luego de cada cambio la documentación debe ser actualizada para facilitar futuros cambios.

Algoritmos

Antes de tratar el tema de la programación propiamente dicha introduciremos un concepto que utilizaremos a lo largo del curso que es término: Algoritmo.

La palabra algoritmo se deriva de la traducción al latín de la palabra árabe Alkhowarismi, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX.

Para dar una definición más específica diremos que un algoritmo es un método con un conjunto de instrucciones utilizadas para resolver un problema específico.

Los algoritmos son independientes de los lenguajes de programación, incluso son más importantes que estos y que las propias computadoras. Un lenguaje de programación es solo un medio para expresar un algoritmo y una computadora es solo un procesador para ejecutarlo.

Características de un algoritmo

- Preciso

Indicar el orden de realización en cada paso. No debe dar lugar a ambigüedades.

- Definido

Si se sigue dos veces, obtiene el mismo resultado.

- Finito

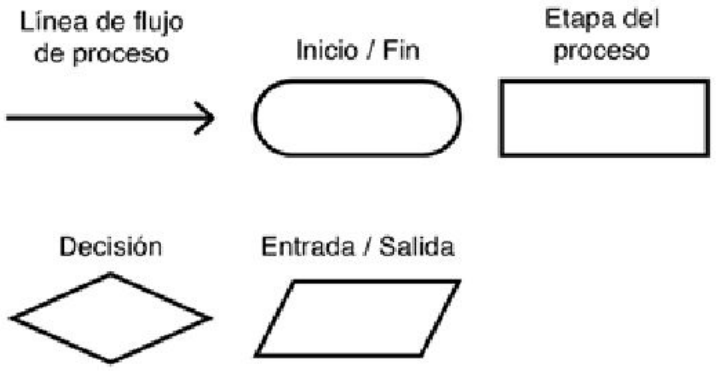
Tiene un número determinado de pasos, es decir tiene fin.

Técnicas de representación de un algoritmo

Las herramientas más utilizadas comúnmente para diseñar algoritmos son: diagramas de flujo y pseudocódigo.

Diagramas de Flujo

Es quizás la forma de representación más antigua, algunos autores suelen llamarlos diagramas de lógica o flujograma. Es una representación gráfica que utiliza símbolos (normalizados por el Instituto Norteamericano de Normalización - ANSI), conectados por líneas y los más frecuentemente utilizados entre otros son:



Pseudocódigo

Es una herramienta de programación en la que las instrucciones se escriben en palabras similares al inglés o español, que facilitan tanto la escritura como la lectura de programas. No existen reglas para la escritura del pseudocódigo en español, pero en general se trata de utilizar una notación estándar que se establece en el momento de trabajar. Algunas palabras comunes son: INICIO, FIN, LEER, ESCRIBIR, entre otras.

Los lenguajes de Programación

Un lenguaje de programación es un notación para escribir programas, a través de los cuales podemos comunicarnos con el hardware y dar así las órdenes adecuadas para la realización de un determinado proceso es decir que la finalidad de un lenguaje de programación es "decirle" al ordenador qué es lo que tiene que hacer paso a paso.

Los distintos niveles de programación nos permiten acceder al hardware, de tal forma que , según utilicemos un nivel u otro, tendremos que usar un determinado lenguaje.

Clasificación de los lenguajes de programación

- Lenguajes de máquina

Son aquellos lenguajes cuyas instrucciones son directamente entendibles por la computadora y no necesitan traducción posterior para que la CPU pueda comprender y ejecutar el programa. Las instrucciones del lenguaje de máquina se expresan utilizando el sistema binario o bit (digito binario, 0, 1).

Fue el primer lenguaje utilizado en la programación de computadoras, eran instrucciones fáciles de leer por la computadora y difíciles para un programador, es por eso que son sustituidos por otros lenguajes más fáciles de aprender y utilizar.

Para evitar la tediosa tarea de escribir programas en lenguaje de máquina se han diseñado otros lenguajes que facilitan la escritura y posterior ejecución de los programas, es así que nacen los lenguajes de bajo y alto nivel.

- Lenguajes de bajo nivel

Son lenguajes generalmente dependientes de la máquina, es decir, se trata de lenguajes de programación que están diseñados para un hardware específico. Un lenguaje típico de bajo nivel es el lenguaje ensamblador y es el primer intento de sustituir el lenguaje de máquina, utilizando para su escritura códigos nemotécnicos en lugar de cadenas de bits.

Abreviaturas típicas de este lenguaje son: ADD, SUB, MPY,DIV, entre otros.

- Lenguajes de alto nivel

También denominados lenguajes evolucionados, son aquellos en los que las sentencias o instrucciones son escritas con palabras similares a los lenguajes humanos, en general inglés, lo que facilita la escritura y la fácil comprensión por el programador.

La idea de evolucionar persigue una serie de objetivos:

1. Lograr independencia de la máquina, pudiendo utilizar un mismo programa en diferentes equipos (transportabilidad), con la única condición de disponer de un programa traductor o compilador.
2. Aproximarse al lenguaje neutral, para que el programa se pueda leer y escribir de manera más sencilla, eliminando muchas de las posibilidades de cometer errores.
3. Incluir rutinas de uso frecuente, como por ejemplo funciones matemáticas, que se guardan en especies de librerías del lenguaje, de manera que sean accesibles siempre que se necesiten.

Son ejemplos de estos: Basic, Pascal, C++, Java, Python etc.

Paradigmas de programación

Un paradigma de programación es un modelo básico de diseño y desarrollo de programas, que permite producir programas con un conjunto de normas específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad, etc.

Los paradigmas pueden ser considerados como patrones de pensamiento para la resolución de problemas. Desde luego siempre teniendo en cuenta los distintos lenguajes de programación.

No es mejor uno que otro, sino que cada uno tiene ventajas y desventajas. También hay situaciones donde un paradigma resulta más apropiado que otro.

Los lenguajes de programación se encuadran en uno o más paradigmas a la vez a partir del tipo de órdenes que permiten implementar.

Clasificación de los paradigmas de programación

- Imperativo

Contienen instrucciones que dicen al ordenador cómo realizar una tarea. Trata con tipos de datos y algoritmos las características de este paradigma son:

1. Describe cómo debe realizarse el cálculo, no el porqué.
2. Un cómputo consiste en una serie de sentencias, ejecutadas según un control de flujo explícito, que modifican el estado del programa.
3. La sentencia principal es la asignación.
4. Las variables son celdas de memoria que contienen datos (o referencias), pueden ser modificadas, y representan el estado del programa.
5. Asociados al paradigma imperativo se encuentran los paradigmas procedural, modular, y la programación estructurada.

El lenguaje representativo sería FORTRAN-77, junto con COBOL, BASIC, PASCAL, C, ADA. También lo implementan Java, C++, C#, Eiffel, Python.

- Declarativo

El enfoque declarativo tiene varias ramas diferenciadas: el paradigma funcional, el paradigma lógico, la programación reactiva y los lenguajes descriptivos, las características de este paradigma son:

1. Describe qué se debe calcular, sin explicitar el cómo.
2. No existe un orden de evaluación prefijado.
3. Las variables son nombres asociados a definiciones, y una vez instanciadas son inmutables.
4. No existe sentencia de asignación.
5. El control de flujo suele estar asociado a la composición funcional, la recursividad y/o técnicas de reescritura y unificación.