Taller 1:

1.

1. Definir los conceptos fundamentales involucrados en el proceso de resolución de un problema a través de la computadora.

* Análisis del problema:

El análisis consiste en una clara definición del problema, donde se contemple exactamente lo que debe hacer el programa y el resultado o solución deseada.

Dado que se busca una solución por computadora, se precisan especificaciones detalladas de entradas y salidas.

Para resolver un problema con un ordenador hay que disponer de los datos de entrada, estudiar el tratamiento que se ha de realizar a dichos datos, la información que se desea obtener como resultado y de que manera debe presentarse.

* Diseño del algoritmo:

Teniendo en cuenta que un algoritmo es un método para resolver problemas, una vez analizando el mismo se precisa diseñar un algoritmo que indique claramente los pasos a seguir para resolverlo.

Para realizar un determinado proceso, se le debe suministrar al ordenador una formula para la resolución de un problema (algoritmo), cuyo diseño debe ser independiente de la computadora que resuelve el problema.

* Programación:

Una vez que el diagrama de flujo o el algoritmo de resolución del problema esta definido se pasa a la fase de codificación del programa en cualquier lenguaje (c, basic, cobol, pascal, etc.) cuyo resultado será el programa fuente, el cual sigue las reglas de sintaxis que el lenguaje escogido exija.

Después de codificado el programa, se introduce en el ordenador mediante unos programas especiales llamados editores.

* Ejecución y pruebas:

El hecho de haber diseñado un buen algoritmo y luego haberlo codificado en algún lenguaje de programación no significa que el programa resuelva correctamente el problema en cuestión.

Por eso, antes de dar por finalizada cualquier labor de programación, es fundamental preparar un conjunto de datos más representativo posible del problema, que permitan probar el programa cuando se ejecute y así verificar los resultados.

1. Definir lenguaje, programa, compilador e intérprete.

* Lenguaje:

Un lenguaje de programación es básicamente un sistema estructurado de comunicación, similar al humano, el cual nos permite comunicarnos por medio de signos, ya sean palabras, sonidos o gestos. Refiriéndonos a los aparatos, este sistema está organizado para que se entiendan entre sí y a su vez interprete las instrucciones que debe ejecutar.

* Programa:

Cuando hablamos específicamente de programa en informática, estamos haciendo referencia a un [software](http://definicion.de/software). Se trata de aplicaciones y recursos que permiten desarrollar diferentes tareas en una [computadora](http://definicion.de/computadora) (ordenador), un teléfono u otros equipos tecnológicos.

* Compilador:

Un compilador es un programa informático, que se encarga de traducir el código fuente de una aplicación que este en desarrollo, es decir convierte un programa hecho en lenguaje de programación de alto nivel a un lenguaje de máquina , el cual es conocido como de bajo nivel, de tal forma que sea más entendible y mucho más fácil de procesar en el equipo en el que se esta ejecutando.

* Interprete:

Un intérprete es un software que recibe un programa en lenguaje de alto nivel, lo analiza y lo ejecuta. Para analizar el programa completo, va traduciendo sentencias de código y ejecutándolas si están bien, así hasta completar el programa origen.

Los intérpretes informáticos, al contrario que los compiladores, no generan un fichero ejecutable u otro programa equivalente en otro lenguaje, por lo que cada vez que se ejecuta el programa original debe pasar por la fase de análisis. Esto hace de los intérpretes más lentos que los compiladores, donde las fases de análisis y ejecución son independientes, por lo que solo se compila una vez y se ejecuta cuantas veces se quiera.

1. Identificar los antecedentes de la programación orientada a objetos.

La Programación Orientación a Objetos (P.O.O.) surge en Noruega en 1967 con un lenguaje llamado Simula 67, desarrollado por Krinsten Nygaard y Ole-Johan Dahl, en el centro de cálculo noruego.

Simula 67 introdujo por primera vez los conceptos de clases, corrutinas y subclases (conceptos muy similares a los lenguajes Orientados a Objetos de hoy en día).

El nacimiento de la Programación Orientación a Objetos en Europa pasó inadvertido para gran parte de los programadores. Hoy tenemos la Orientación a Objetos como un niño de 33 años al que todos quieren bautizar.

Uno de los problemas de inicio de los años setentas era que pocos sistemas lograban terminarse, pocos se terminaban con los requisitos iniciales y no todos los que se terminaban cumpliendo con los requerimientos se usaban según lo planificado. El problema consistía en cómo adaptar el software a nuevos requerimientos imposibles de haber sido planificados inicialmente.

1. Definir los conceptos involucrados en la programación orientada a objetos.

* Objeto.
* Herencia.
* Abstracción.
* Polimorfismo.
* Encapsulamiento.
* Método.
* Programación estructurada.

1. Identificar las etapas para la resolución de un problema bajo un enfoque orientado a objetos.

* Definición del problema.
* Análisis orientado a objetos.
* Programación.
* Documentación.

1. Identificar los tipos de datos que se emplean en el desarrollo de un programa.

* Enteros.
* Flotantes.
* Cadena de caracteres.
* Booleanos.
* Arreglos.
* Estructuras.

1. Definir los conceptos de variables, expresiones y constantes.

* Variables:

Es un espacio de memoria reservado para almacenar un valor que corresponde a un tipo de dato soportado por el lenguaje de programación. Una variable es representada y usada a través de una etiqueta (un nombre) que le asigna un programador o que ya viene predefinida.

* Expresiones:

Es una combinación de operadores y operandos de cuya evaluación se obtiene un valor. Los operandos pueden ser nombres que denoten objetos variables o constantes, funciones, literales de cualquier tipo adecuado de acuerdo con los operadores u otras expresiones más simples. La evaluación de una expresión da lugar a un valor de algún tipo, una expresión se dice que es del tipo de su resultado.

* Constantes:

Es un valor que no puede ser alterado o modificado durante la ejecución de un programa, únicamente puede ser leído. Una constante corresponde a una longitud fija de un área reservada en la memoria principal del ordenador, donde el programa almacena valores fijos.

1. Describir el procedimiento para declarar los formatos de las sentencias de asignación y de entrada y salida.

* Sentencia de asignación:

Se debe identificar la variable y asignarle con el símbolo “=” el contenido de la variable que contiene la información.

Ejemplo:

int variableA = variableB;

* Sentencia de entrada:

Se debe hacer uso de la librería Scanner para capturar la información proporcionada por el usuario.

Ejemplo:

Scanner entrada = new Scanner(System.in);

* Sentencia de salida:

Se hace uso del método “System.out.println” que permite imprimir por consola el parámetro entregado.

Ejemplo:

System.out.println(“Hola mundo”);