Unidad III

Herramientas de Programación

3.1 Simbología.

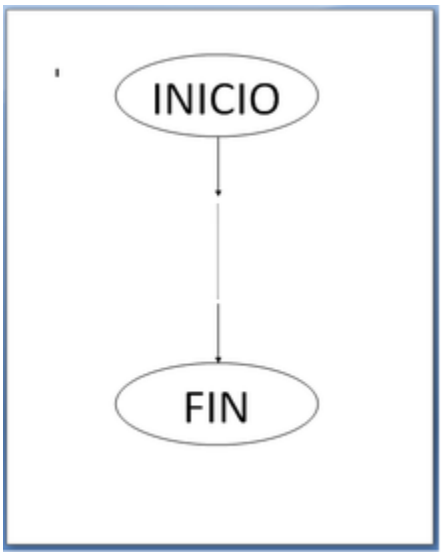
La simbología es el estudio de los símbolos o el conjunto de éstos. Un símbolo, por otra parte, es la representación sensorial de una idea que guarda un vínculo convencional y arbitrario con su objeto.

La noción de simbología se utiliza para nombrar al sistema de los símbolos que identifican a los diferentes elementos de algún ámbito. En este sentido puede hablarse, por ejemplo, de la simbología electrónica (con los íconos o representaciones gráficas que permiten reconocer cada elemento interviniente).
La electricidad, la química y la mecánica, entre otros ámbitos del conocimiento, tienen su propia simbología. Quien conoce la simbología de una especialidad, puede expresarse mediante los símbolos e interpretar diagramas o esquemas que apelen a los símbolos en lugar de las palabras.
Es posible clasificar a la simbología según su objeto de estudio o área de incumbencia. La simbología francmasónica analiza los símbolos de los masones y devela los mensajes encerrados en cada uno de ellos. La simbología religiosa, por otra parte, estudia los símbolos que intervienen en una creencia o práctica de una religión.
La simbología de un país refiere a todos los símbolos que permiten reflejar la identidad nacional. El himno nacional, la bandera y el escudo son ejemplos de símbolos que forman parte de la simbología de un país.

3.2 Reglas para la construcción de diagramas.

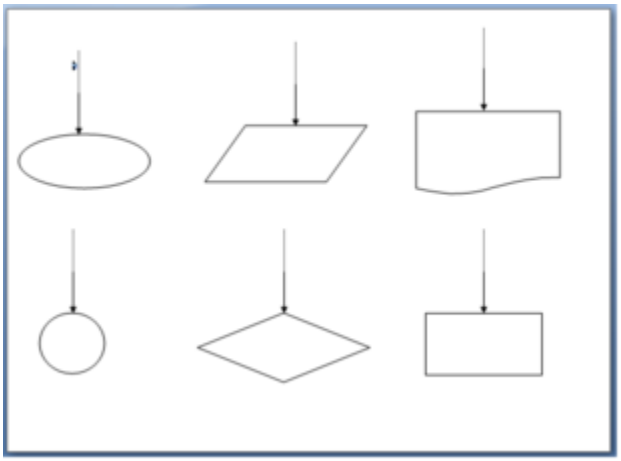
En un diagrama de flujo se representa de manera gráfica una serie de pasos a seguir para alcanzar la solución de un problema. Los símbolos presentados, colocados adecuadamente, permiten crear una estructura gráfica flexible que ilustra los pasos a seguir para alcanzar el resultado específico.
A continuación se presentan los pasos para la realización de un diagrama de flujo:

1. Todo diagrama debe de tener un inicio y un fin.



2. Las lineas utilizadas para indicar la direccion del flujo del diagrama deben ser rectas, verticales y horizontales.

3. Todas las lineas utilizadas para indicar la direccion del flujo del diagrama deben estar conectadas. la coneccion puede ser a un simbolo que exprese lectura, proceso, decision, impresion, conexion o fin de diagrama.



4. El diagrama de flujo debe ser construido de arriba hacia abajo y de izquierda a derecha

5. La notacion utilizada en el diagrama de flujo debe serindependiente del lenguaje de programacion. La solucion presentada en el diagrama puede escribirse posteriormente y facilmente en diferentes lenguajes de programacion.

6. Es conveniente cuando realizamos una tarea compleja poner comentarios que expresen o ayuden a en tender lo que hicimos.

7. Si el diagrama de flujo requiriera mas de una hoja para su construccion, debemos utilizar los conectores adecuados y enumerar las paginas convenientemente.

8. No puede llegar mas de una linea a un simbolo.

3.3 Pseudocódigo.

En ciencias de la computación, y análisis numérico el **pseudocódigo** (o falso lenguaje) es una descripción informal[1] de alto nivel de un algoritmo informático de programación, compacto e informal, que utiliza las convenciones estructurales de un lenguaje de programación verdadero[2] , pero que está diseñado para la lectura humana en lugar de la lectura mediante máquina, y con independencia de cualquier otro lenguaje de programación. Normalmente, el pseudocódigo omite detalles que no son esenciales para la comprensión humana del algoritmo, tales como declaraciones de variables, código específico del sistema y algunas subrutinas. El lenguaje de programación se complementa, donde sea conveniente, con descripciones detalladas en lenguaje natural, o con notación matemática compacta. Se utiliza pseudocódigo pues este es más fácil de entender para las personas que el código de lenguaje de programación convencional, ya que es una descripción eficiente y con un entorno independiente de los principios fundamentales de un algoritmo. Se utiliza comúnmente en los libros de texto y publicaciones científicas que se documentan varios algoritmos, y también en la planificación del desarrollo de programas informáticos, para esbozar la estructura del programa antes de realizar la efectiva codificación. No existe una sintaxis estándar para el pseudocódigo, aunque los ocho IDE's que manejan pseudocódigo tengan su sintaxis propia. Aunque sea parecido, el pseudocódigo no debe confundirse con los programas esqueleto que incluyen código ficticio, que

pueden ser compilados sin errores. Los diagramas de flujo y UML pueden ser considerados como una alternativa gráfica al pseudocódigo, aunque sean más amplios en papel.

3.4 Tipos de datos y expresiones.

Todos los datos tienen un tipo asociado con ellos. Un dato puede ser un simple carácter, tal como ´B´, aun valor entero tal como 35 o un numero real 141592. Una operación suma no tiene sentido con caracteres solo con números. Por consiguiente si el compilador detecta una operación de suma de dos caracteres, normalmente producirá un error. Incluso entre tipos numéricos, la operación de suma se almacena de modo distinto. Esto se debe a que números enteros y reales se almacenan de modos diferentes. A menos que el programa reconozca el tipo de datos, si es el valor entero a real, no puede ejecutar correctamente la operación de suma.
La asignación de tipos de datos tiene dos objetivos principales:

Detectar errores de operaciones en programas.

Determinar cómo ejecutar las operaciones.

3.5 Estructuras lógicas.

En álgebra universal y en teoría de modelos, una **estructura** consiste en un conjunto acompañado con una colección de funciones y relaciones finitas las cuales están definidas en él. El álgebra universal estudia estructuras que generalizan las estructuras algebraicas tales como grupos, anillos, campos, retículos y espacios vectoriales. El término álgebra universal es usado para estructuras sin símbolos de relaciones. La teoría de modelos tiene un alcance diferente que abarca teorías mas arbitrarias, incluyendo estructuras mas fundamentales como modelos de la teoría de conjuntos. Desde el punto de vista modelo-teórico, las estructuras son objetos usados para definir la semántica de lógicas de primer orden.