INGENERIA EN SISTEMA COMPUTACIONALES

MONBRE DEL ALUMNO: FRANCISCO LOPEZ ARGUETA

NOMBRE DEL PROFESOR: ANDRES ALEJANDRO REYES MOLINA

MATERIA: ELEMENTOS DE PROGRAMACION ESTRUCTURADA

NOMBRE DE LA ACTIVIDAD: Super nota.

CUATRIMETRE: 4TO

INTRODUCCION:

Un ordenador o computadora está, desde que se enciende hasta que se apaga totalmente, ejecutando un algoritmo. Por lo general, estos algoritmos, escritos para que los entienda una máquina, terminan siendo vagos y confusos para la mayoría de quienes no han estudiado programación. Una máquina no puede entender "escribe Hola Mundo!" porque no sabe lo que es "escribe" ni lo que es una letra o un espacio, ni lo que es una pantalla. En cambio, puede entender "mov eax, 0x23afb31" (escribir en el registro eax el número 0x23afb31), aunque nosotros no. Un ordenador es solo un circuito electrónico, no funciona a base de magia ni nada por el estilo.

Debido a lo difícil que es escribir en lenguaje máquina, e incluso en ensamblador, se crearon diferentes lenguajes de programación, más o menos parecidos al inglés actual y a cómo se redacta un algoritmo. Estos lenguajes proveen de cosas tan complejas para una máquina como los bucles for. Los compiladores se encargan de traducir esos ficheros al lenguaje ensamblador que corresponda, el ensamblador de traducirlos a lenguaje máquina y el enlazador de juntar todo ese código máquina en un solo archivo, el programa. Y el microprocesador, se encarga de ir encendiendo o apagando transistores según lo que le diga el código máquina.

-ALGORITMOS

En informática, se llama algoritmo a una secuencia de instrucciones u operaciones específicas que permiten controlar determinados procesos. Se trata de conjuntos finitos y ordenados de pasos, que nos conducen a resolver un problema o tomar una decisión.

Por ejemplo, una acción simple y cotidiana como encender la luz de la habitación puede describirse como un conjunto ordenado de pasos, como son:

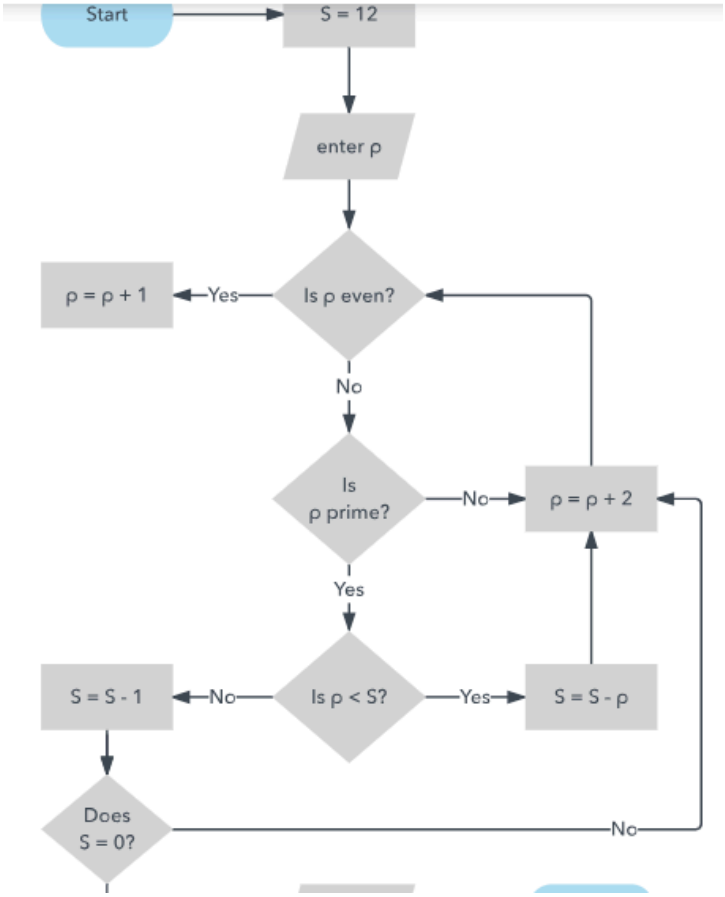
2) ¿La luz está apagada?
NO: FIN
SÍ: ve al paso 2

2) Presiona el interruptor y vuelve al paso 1.

Hoy en día, la presencia de los algoritmos es muy conocida gracias a la automatización digital. Algoritmos sumamente complejos y especializados controlan el funcionamiento de las redes sociales y de los buscadores de internet, entre otras piezas de software, para permitirle al usuario una experiencia personalizada.

-DIAGRAMA DE FLUJO

Un diagrama de flujo es un diagrama que describe un proceso, sistema o algoritmo informático. Se usan ampliamente en numerosos campos para documentar, estudiar, planificar, mejorar y comunicar procesos que suelen ser complejos en diagramas claros y fáciles de comprender. Los diagramas de flujo emplean rectángulos, óvalos, diamantes y otras numerosas figuras para definir el tipo de paso, junto con flechas conectoras que establecen el flujo y la secuencia. Pueden variar desde diagramas simples y dibujados a mano hasta diagramas exhaustivos creados por computadora que describen múltiples pasos y rutas. Si tomamos en cuenta todas las diversas figuras de los diagramas de flujo, son uno de los diagramas más comunes del mundo, usados por personas con y sin conocimiento técnico en una variedad de campos. Los diagramas de flujo a veces se denominan con nombres más especializados, como "diagrama de flujo de procesos", "mapa de procesos", "diagrama de flujo funcional", "mapa de procesos de negocios", "notación y modelado de procesos de negocio (BPMN)" o "diagrama de flujo de procesos (PFD)". Están relacionados con otros diagramas populares, como los diagramas de flujo de datos (DFD) y los diagramas de actividad de lenguaje unificado de modelado (UML).



-PSEUDOCODIGO

El pseudocódigo es un vocablo que se maneja en las áreas relacionadas con algoritmos y la programación de computadoras. Es un método que faculta a cualquier programador constituir con simplicidad la elaboración de un algoritmo. Tal como lo indica el nombre, es un código falso o la representación de un código, que puede ser sencillo de entender incluso por alguien que solo tenga cierta noción de programación a nivel primario. Los algoritmos se escriben muchas veces con el apoyo de un pseudocódigo, ya que así lograrán ser descifrados por los programadores, sin importar la experiencia o conocimiento que tengan en programación. Por tanto, el pseudocódigo no es más que la implementación de un algoritmo en forma de textos informativos y anotaciones, escrito en un lenguaje sencillo.

Características de un pseudocódigo

– **Herramienta para diseñar algoritmos**. El pseudocódigo es un lenguaje no formal que sirve para que los programadores puedan desarrollar algoritmos. Es una herramienta para diseñar algoritmos que se basa en textos. El uso de pseudocódigo tiene como propósito que un algoritmo sea efectivo. Sirve para concebir un algoritmo a través de un esquema, como paso previo a su codificación en lenguaje de programación.

– **Se escribe en cualquier formato**. Se puede escribir pseudocódigo en cualquier formato deseado. Por ejemplo, se podría usar un formato de academia, que sea sumamente detallado y estructurado, involucrando mucha matemática. Por otro lado, también se puede escribir como un resumen simple de lo que se espera que realice el código.

– **Paso previo a la programación real**. El pseudocódigo no es realmente un lenguaje de programación. Para escribir esta especie de código se utiliza una sintaxis sencilla en español, que luego será reformado a la sintaxis correcta de un lenguaje de programación en particular. Esto se hace para reconocer errores en el flujo y para vislumbrar el flujo de datos que utilizará el programa final. Esto favorece en gran medida a no perder tiempo durante la programación real, ya que los errores conceptuales estarán ya corregidos.

– **Reglas.** Las reglas del pseudocódigo son razonablemente sencillas. Las declaraciones son normalmente secuencias, selecciones o iteraciones. Todas las declaraciones que tengan una "dependencia" deben sangrarse. Por ejemplo, en lenguaje C las declaraciones de secuencia son imperativas. La de selección es la declaración "if-then-else", y la iteración se satisface con un conjunto de declaraciones, como "while", "do" o "for". La declaración "En caso" se satisface con el comando "switch".

-COMENTARIO

Un **comentario en programación** es la línea de texto en nuestro código fuente que el compilador ignora. Sabemos que nuestro código fuente está compuesto de instrucciones, y el compilador traduce esas instrucciones del lenguaje de programación que estamos usando a lenguaje máquina. Cuando el compilador se encuentra con un comentario, esa línea, o varias de ellas, no las traduce, y continúa buscando en la instrucción siguiente. Los comentarios en programación se utilizan para poner aclaraciones del código, y así es más fácil de entender lo que hace, aunque también se utilizan para *anular* parte de un código. Aunque con un **sistema de control de versiones** como GIT no sería necesario hacer eso. En algunos lenguajes, como Java, se utilizan los comentarios con un formato específico para crear documentación aparte (llamado JavaDoc).

-VARIABLES

Una variable en programación es una unidad de datos que puede cambiar de valor. Es la forma más simple de almacenamiento, representando una zona de MEMORIA donde se almacena un elemento de datos. Si un programa de computadora fuera un edificio, entonces las variables serían los ladrillos que constituyen su base. Las variables son componentes críticos de cualquier programa. Este no podría ser efectivo sin variables. Una variable puede ser la temperatura del aire o los precios de las acciones. Todos estos son valores que pueden cambiar. Las variables tienen dos propósitos importantes, que son que el programador puede elegir los nombres de ellas, facilitando así la programación, y también que pueda escribir programas o funciones que trabajen con cualquier valor en ellas. Si ya se está familiarizado con las hojas de cálculo, se podría pensar que las variables son como las celdas, que luego podrán ser usadas en fórmulas, independientemente de los valores que contengan en ellas. Todos los lenguajes de programación procedimentales, como C, Basic y Pascal tienen variables, pudiendo admitir diferentes tipos y permitir manipularlas de diferentes maneras

-ESTRUCTURA DE CONTROL

Las estructuras de control son las herramientas que utilizan los programadores para controlar el flujo de un programa. Las estructuras de control permiten a los programadores tomar decisiones y realizar tareas repetitivas en un programa. En esencia, las estructuras de control son las reglas que gobiernan cómo un

programa se comporta en diferentes situaciones. Las estructuras de control son importantes porque permiten a los programadores crear programas más complejos y eficientes. Con las estructuras de control, los programadores pueden tomar decisiones y realizar tareas repetitivas de manera automática, lo que ahorra tiempo y reduce los errores. Además, las estructuras de control son esenciales para la creación de programas interactivos que responden a la entrada del usuario. En resumen, las estructuras de control son importantes porque permiten a los programadores crear programas más complejos y eficientes, tomar decisiones y realizar tareas repetitivas de manera automática y crear programas interactivos que responden a la entrada del usuario

-IMPORTANCIA DE LA PROGRAMACION DE COMPUTADORAS

El desarrollo de un programa consiste en una serie de pasos. El programador define un problema, planea una solución, codifica el programa, lo comprueba y finalmente documenta el programa. Por lo general, el programador define lo que sabe y el objetivo, selecciona un programa a utilizar, depura el programa en las fases posteriores a la terminación - para asegurar que no se introduzcan errores - y a continuación documenta el diseño, desarrollo y pruebas del programa. Con el rostro siempre cambiante de la tecnología informática, la programación es un ambiente emocionante y desafiante que pocos programadores sueñan con dejar.

-CLASIFICASSION DE LOS LENGUAJES DE PROGRAMACION

La clasificación de los lenguajes de programación es un tema central en el mundo del desarrollo de software. Esta categorización nos permite entender las características, fortalezas y limitaciones de cada lenguaje, facilitando la elección del más adecuado para cada proyecto o tarea específica.

Existen varios criterios para clasificar los lenguajes de programación, cada uno ofreciendo una perspectiva única sobre sus capacidades y usos. Vamos a explorar en detalle los principales métodos de clasificación:

1. Según su nivel de abstracción
2. Por paradigma de programación
3. Según el propósito del lenguaje
4. Basada en la forma de ejecución
5. Por generación

Cada una de estas clasificaciones aporta información valiosa sobre cómo funcionan los lenguajes y en qué situaciones son más efectivos.

-Lenguajes de bajo nivel

Los lenguajes de bajo nivel son aquellos que están más cerca del lenguaje de máquina, es decir, de las instrucciones que el hardware puede ejecutar directamente. Estos lenguajes ofrecen un control muy preciso sobre el hardware, pero son más difíciles de leer y escribir para los humanos.

Ejemplos de lenguajes de bajo nivel:

1. Lenguaje de máquina: Es el nivel más bajo de programación, consistente en secuencias de números vinarios de la computadora interpreta como instrucciones.

2. Lenguaje ensamblador: Es un paso por encima del lenguaje de máquina, usando nemotécnicos para representar operaciones.

Los lenguajes de bajo nivel son ideales cuando se necesita un control preciso sobre el hardware o cuando se busca la máxima eficiencia en términos de velocidad y uso de recursos. Sin embargo, programar en estos lenguajes es más complejo y propenso a error

-DISEÑO DE UN ALGORITMO

En el diseño de algoritmos hay un objetivo: **analizar en detalle y comprender la naturaleza del problema**. Esto es primordial para obtener una idea general y certera de lo que realmente hay que hacer o lo que se solicita. Otro punto a tener en cuenta es que los algoritmos son independientes tanto del lenguaje de programación como del ordenador donde se ejecuta.

A la hora de diseñar algoritmos hay que cumplir con una serie de **requisitos** que son clave para no cometer errores:

- **Deben tener un principio y un fin.** La finitud es una característica clave de estos procesos matemáticos.
- **Deben ser precisos.** Los algoritmos deben precisar el orden de realización de cada acción, de forma clara, sin ambigüedades.
- **Secuencia clara.** Esta sucesión de pasos debe tener un orden inalterable.
- **Ser repetibles.** Estos procesos se pueden repetir tantas veces como se desea, pero es necesario que devuelvan siempre los mismos resultados frente a la misma solicitud.

¿Cómo se verifica si el algoritmo cumple con los estándares de calidad? Para ello, es necesario que los algoritmos estén compuestos de las siguientes **características**, complementarias a las anteriormente descritas:

- **Validez.** El algoritmo diseñado responde exactamente a la solicitud concreta que se ha realizado, es decir, hace lo que se le pide que haga.
- **Eficiencia.** Debe responder al problema en el menor tiempo posible.
- **Optimización.** El algoritmo que se ha desarrollado es el mejor para resolver el problema que se desea.
- Otra de las claves del algoritmo es que en él se plasman **las tres partes de una solución informática**:
- **Entrada.** Conjunto de datos con los que el algoritmo procesa la información.
- **Proceso.** Cálculos necesarios para llegar a resolver el problema.
- **Salida.** Es el resultado o resultados finales obtenidos después de procesar los cálculos.

La importancia de los algoritmos radica principalmente en que mediante ellos se puede ofrecer soluciones óptimas al afrontar **cualquier tipo de problema procesado mediante un ordenador**.