LENGUAJES DE PROGRAMACIÓN

1. ¿Qué es un algoritmo? ¿Cuáles son las características que ha de cumplir un algoritmo?

Un algoritmo se define como un conjunto ordenado y finito de operaciones que te permite solucionar un problema. Un algoritmo debe ser preciso, definido y finito.

Bla bla bla definir mejor preciso, definido y finito.

2. ¿Qué es un lenguaje de programación?

Un lenguaje de programación es el medio por el cual nos comunicamos directamente con la máquina. Son lenguajes creados para facilitar la elaboración de programas de forma más sencilla que con el lenguaje máquina.

3. Indica las ventajas e inconvenientes del lenguaje máquina, ensamblador y de los lenguajes de alto nivel.

El lenguaje máquina es el lenguaje propio de las computadoras.
- Ventajas: La principal ventaja se encuentra en la velocidad en lo vinculado a la ejecución.
- Desventajas: Tiempo de programación, falta de portabilidad, dificultad para aprenderlo.

4. ¿Qué novedades aportaron los lenguajes de cuarta Generación?

Entre 1980 y 1990 aparecen los lenguajes orientados a objetos. Estos incorporan herramientas y mecanismos para restringir el acceso a los datos que forman parte de los objetos. Asimismo, estos lenguajes también determinan el concepto de herencia, que permite la reutilización del código.

5. Explica las semejanzas y diferencias que hay entre un compilador y un intérprete.

Solo se genera código ejecutable si el código fuente se encuentra sin error léxicos, sintácticos o semánticos. Para ello existen dos formas:
1. Compilación. El proceso de compilación de un programa se hace mediante dos herramientas/programas: el compilador y el enlazador. El primero traduce el código fuente (sin errores) en un lenguaje llamado *código objeto* y el enlazador inserta en dicho código las funciones de librerías necesarias para poder hacer un programa ejecutable.
2. Interpretación. Para esta forma solo se necesita un programa, que va realizando una traducción del código fuente cada vez que se quiere ejecutar.

6. Según su funcionalidad, el software se categoriza en software de Aplicación, de gestión, de programación y de sistemas. Busca ejemplos de cada uno de los tipos.

1. Software de aplicación. Son programas diseñados para que los usuarios realicen tereas específicas en sus dispositivos. Ejemplos: Google Chrome, Adobe Photoshop, Microsoft Word.
2. Software de gestión. Ayuda a las empresas a organizar, almacenar y acceder a sus documentos de manera eficiente. Ejemplos: Microsoft SharePoint, Odoo, SAP.
3. Software de programación. Son herramientas que permiten al usuario a crear, editar, depurar y mantener programas. Ejemplos: IntelliJ Idea, VSCode, Eclipse, etc.
4. Software de sistemas. Los sistemas operativos son ecosistemas que permite controlar e interactuar con el hardware y otros programas. Ejemplo: MacOS, Linux, Windows

7. ¿Qué se entiende por software libre? ¿Y por software de código abierto? ¿Y por software propietario?

Un software libre es aquel que su código fuente puede ser modificado, estudiado, cambiado e incluso alterado. Un ejemplo puede ser Linux.

El software propietario, es aquel que está en propiedad de una empresa, por ejemplo, Microsoft

y no permite acceder al código fuente ni mucho menos alterarlo.

El software de código abierto se asemeja mucho al software libre, pero tiene más libertades. En la práctica, todo software libre se puede clasificar como abierto, aunque no todo el software de código abierto tiene por qué ser libre.

8. ¿Qué se conoce como software *on premise*? ¿Cuál es la alternativa a este tipo de software?

Se refiere a software e infraestructuras alojados dentro de instalaciones o centro de datos de la empresa. Se instala y se ejecuta en computadoras dentro de estas instalaciones.

Una alternativa para esto puede ser los softwares en la nube, que son programas que se utilizan a través de internet, en lugar de instalarlo localmente en un ordenador

9. Una de las características de las metodologías ágiles de desarrollo de productos es la entrega continua. ¿Qué significa esto?

Es una práctica de desarrollo de software que automatiza y agiliza la entrega de código a producción, asegurando que el software esté siempre en un estado desplegable.

Entregas periódicas funcionales.

10. ¿En qué estructuras de control se basa la programación estructurada?

La programación estructurada mejora la claridad y calidad del código, utilizando solo tres estructuras de control: secuencia, condicionales e iteración.

11. Describe las fases de Desarrollo de software

Las fases son las siguientes.

1. Diseño. En la fase de diseño, los desarrolladores de software analizan los requisitos e identifican las mejores soluciones.
2. Desarrollo. Es la fase de programación /codificación.
3. Pruebas. Esta fase puede llegar a ser una de las más largas, una vez el software esté finalizado se somete a pruebas para poder comprobar fallos.
4. Implementación. Fase donde se entrega o se vende el programa
5. Mantenimiento. La fase final donde se busca mantener el software a través de actualizaciones.

12. ¿En qué consisten las pruebas de caja blanca? ¿Y las pruebas de caja negra?

Las pruebas de coja blanca son una forma de probar aplicaciones que proporciona al evaluador un conocimiento completo de la aplicación, incluyendo el acceso al código fuente y a los documentos del diseño.

En cambio, las pruebas de caja negra, es una forma de prueba que se realiza sin conocimiento de los componentes internos de un sistema, se pueden realizar para evaluar la funcionalidad, la seguridad, el rendimiento y otros aspectos de una aplicación.

13.¿Qué son las pruebas de carga? ¿Y las pruebas de estrés?

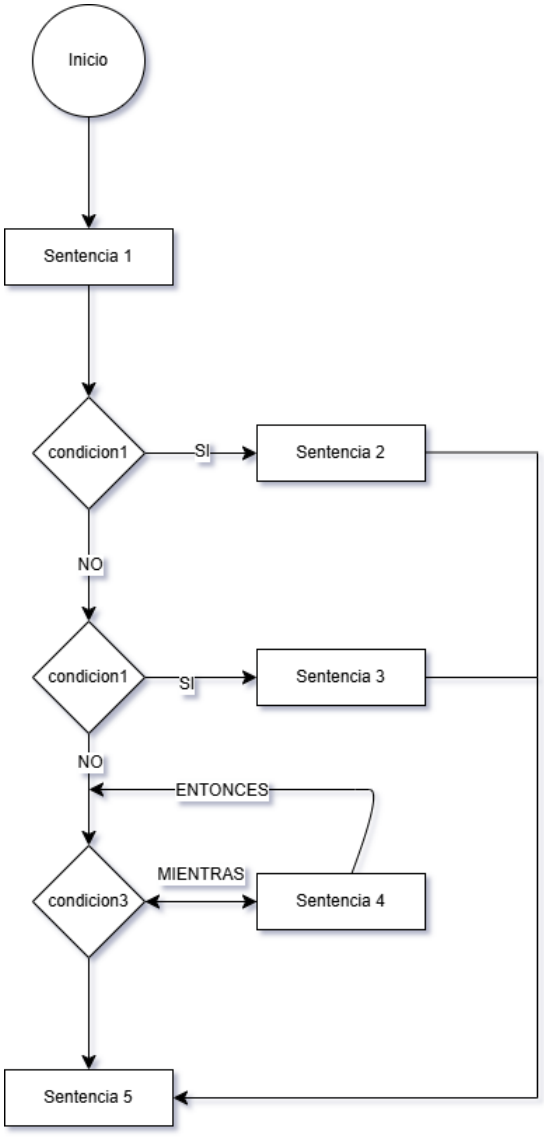
Las pruebas de carga se usan para evaluar el rendimiento bajo cargas normales o esperadas. Buscan encontrar fallos con un uso realista.

Mientras tanto, las pruebas de estrés buscan evaluar el comportamiento y el rendimiento de la aplicación bajo condiciones extremas o más allá de los límites esperados.

LENGUAJES DE PROGRAMACIÓN

14.Elabora el diagrama de flujo para el siguiente fragmento de Pseudocódigo.

```
Sentencia 1
IF condición1
      Sentencia 2
ELSE IF condición 2
            Sentencia 3
      ELSE
            WHILE condición 3
                  Sentencia 4
Sentencia 5
```



15.En POO, ¿qué se entiende por objeto?

Un objeto es algo a lo que se le puede enviar *mensajes* y que puede responder a los mismos, que tiene un *estado*, un *comportamiento* bien definido y una *identidad*.

16. ¿Qué abstracciones se contemplan en lenguajes de POO?

La abstracción surge de reconocer las similitudes entre objetos, situaciones o procesos en el mundo real y la decisión de concentrarse en esas similitudes e ignorar las diferencias. Los lenguajes de POO permiten definir interfaces comunes para comunicarse con conjuntos de objetos. Estas interfaces están compuestas por los métodos. ***

Las interfaces definen el método. Y cuando añades la implementación del método, eso ya es una clase

17.¿En qué consiste la encapsulación en la POO? Como norma general, ¿cómo deben ser los atributos de una clase? ¿Y los métodos?

La capacidad de encapsular permite mantener oculta la implementación de abstracción a los usuarios de esta, de esta forma; ninguna parte de un sistema complejo depende de cómo se ha implementado la otra parte.

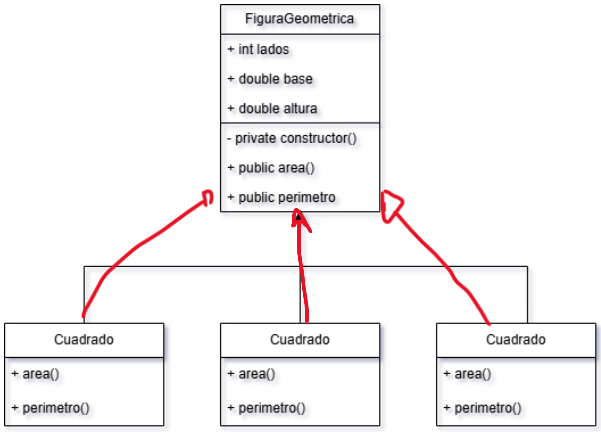
Los atributos y métodos pueden tener una serie de “modificadores” que se aplican a las propiedades y a los métodos.

- *(+) public:* se puede acceder libremente a unas partes de la clase
- *(-) private:* se prohíbe el acceso a unas partes de la clase
- *(#) protected:* se puede acceder a unas partes de la clase, pero con restricciones.

Los atributos de una clase deberían deberías ser “privados” y los métodos “públicos”.

LENGUAJES DE PROGRAMACIÓN

18.Dibuja un diagrama de clases que represente las figuras geométricas empleando la herencia.



19.Explica el concepto de polimorfismo. Documéntalo con algún ejemplo.

Es la propiedad que permite emplear el mismo nombre para nombrar métodos de diferentes abstracciones. La consecuencia de esto es que los efectos serán diferentes dependiendo del método que se ejecute.

El polimorfismo suele afectar en herencias entre interfaces y clases, por ejemplo:

Dos clases distintas: *Gato* y *Perro*, que las dos heredan de *Animal*.

La clase *Animal* tiene un método haceSonido().

A su vez, las interfaces tienen un método abstracto con el mismo nombre, pero en *Gato* hace *"miau"* y en *Perro* hace *"guau"*.

20. ¿Cuál es la función de un constructor?

La función de un constructor en una clase es inicializar un objeto cuando se crea, estableciendo su estado inicial y preparando el objeto para su uso.

21. ¿Qué es UML? Busca información sobre UML: quiénes lo diseñaron, qué tipos de diagramas permite desarrollar.

UML (Lenguaje Unificado de Modelado) no es un lenguaje de programación, fue creado para forjar un lenguaje de modelado visual común para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. Fue diseñado por "The Three Amigos", ya habiendo desarrollado otras metodologías (como los lenguajes POO) publicaron los documentos UML 0.9 y 0.91 en 1996. Y tras una buena recibida por parte de personas y compañías, publicaron en 1999 la Guía de usuario para el Lenguaje Unificado de Modelado. Sacaron una segunda edición (UML 2.0) en 2005. Diagramas de clase, casos de uso, diagrama de flujo, diagramas de secuencia...

22. ¿Qué es LoadView? ¿Y Taurus?

LoadView es un conjunto de pruebas de carga basadas en la nube que ayuda a los equipos de desarrollo a predecir cómo un sitio web, una aplicación web, una API responderán a varios niveles de tráfico y cargas de trabajo. Una prueba de estrés en otras palabras.

Taurus es otro conjunto de pruebas automatizadas de código abierto que se utiliza para pruebas de rendimiento.

Taurus prueba un trozo del código y hace una prueba de rendimiento. LoadView, pone a tope en la nube. Y lo hace en la nube para que el usuario tenga que pasar un cortafuegos.

LENGUAJES DE PROGRAMACIÓN

23. Escribe un algoritmo para saber si un número es múltiplo de 2, de 3 o de 6. Genera después una batería de casos de prueba para probar todos los caminos posibles.

Inicio

Usuario introduce un número.

Si num MOD 6 = 0 Entonces

Imprimir por pantalla: “El número es múltiplo de 6, por lo tanto, también de 2 y de 3”

Fin

Sino Si num MOD 2 = 0 Entonces

Imprimir por pantalla: “El número es múltiplo de 2”

Sino Si num MOD 3 = 0 Entonces

Imprimir por pantalla: “El número es múltiplo de 3”

Sino Entonces

Imprimir por pantalla: “El número no es múltiplo ni de 6, ni de 2, ni de 3.”

Fin

Casos de prueba:

Número de Prueba	**Número Introducido**	**Obtenido**
1	2	Es múltiplo de 2.
2	3	Es múltiplo de 3.
3	6	Es múltiplo de 6 y, por tanto, de 2 y de 3.
4	7	No es múltiplo de ninguno.

	A	B	C
1	1	0,8	
2	2	0,6	
3	3	0,5	
4	4	0,9	
5	5	0,7	
6	6	1	
7	7	1	
8	8	1	
9	9	0,7	
10	10	0,6	
11	11	1	
12	12	0,7	
13	13	1	
14	14	1	
15	15	0,6	
16	16	0,5	
17	17	1	
18	18	0,8	
19	19	1	
20	20	1	
21	21	1	
22	22	1	
23	23	1	
24	NOTAS	8,434782609	
25			