Nombre: Irene Frías Ramos

Grupo: 1º DAM

Fecha: 28/10/2024

1.- Explica detalladamente de qué forma se combinan y complementan el modelo en espiral y el modelo de construcción de prototipos.

Son los procesos incrementales en el que se van obteniendo versiones del software cada vez más completas hasta llegar al producto final.

El modelo evolutivo es equivalente al modelo en cascada, en vez de aplicarse sobre todo el software, se aplica sobre los incrementos del software o versiones de la aplicación más completas. De forma que se saca provecho de las mejores características de varios modelos de ciclo de vida.

En conclusión, ambos modelos utilizan las ventajas en varios enfoques, formando lo mejor de los modelos iterativos y evolutivos para adaptarse mejor a los cambios de requisitos y gestionar el riesgo de forma continua.

2.- Tipos de lenguaje de programación. Definición y aplicación.

Lenguaje máquina o de bajo nivel: es aquel que utiliza el código binario para establecer la comunicación entre el hardware y la máquina. Fue el primer lenguaje que se usó, pero debido a su dificultad fue sustituido por el lenguaje natural, más fáciles de aprender y utilizar.

Actualmente no se utiliza mucho por esa dificultad, debe tener un control preciso del hardware como en sistemas establecidos o en el desarrollo de controladores.

Lenguaje ensamblador o intermedio: son aquellos lenguajes que no utilizan códigos numéricos un direcciones de memoria, se representan mediante instrucciones de forma simbólica. A cada dato se le asigna un código en la que guarda relación con la operación o el dato al que representa. Utilizan direcciones binarias para referirse a esos datos. Los lenguajes ensambladores son más fáciles de entender por el ser humano, a través de esas instrucciones por ello se denomina lenguaje ensamblador.

Se suele utilizar para el desarrollo de los sistemas operativos, firmware y software de alto rendimiento.

Lenguaje de alto nivel o lenguajes evolucionados: su finalidad es liberar al programador de tareas aburridas y dificultosas en las que pueda frenar su productividad y eficiencia. Ofrecen que se conozcan ciertas características de la máquina y lo representen en estos lenguajes respecto al lenguaje natural.

Se usan habitualmente para el desarrollo de aplicaciones empresariales, web, inteligencia artificial,…

3.- ¿De qué tres tipos de lenguajes de programación se puede hablar en función de su cercanía al lenguaje que utiliza el ordenador o al lenguaje natural? ¿Qué tipo de le de programación son los más empleados en la actualidad y por qué?

Lenguaje máquina o de bajo nivel: son lenguajes cercanos al lenguaje de máquina formados por 0 y 1, es decir el código binario que el procesador puede ejecutar directamente.

Se utilizan normalmente para específicas tareas sobre el control del hardware para poderlo ejecutar. Los ejemplos de un lenguaje sería ensamblador y lenguaje máquina.

Lenguaje ensamblador o intermedio: son aquellos lenguajes que no utilizan códigos numéricos un direcciones de memoria, se representan mediante instrucciones de forma simbólica. A cada dato se le asigna un código en la que guarda relación con la operación o el dato al que representa.

Utilizan el desarrollo de los sistemas operativos software de rendimiento crítico y aplicaciones de bajo nivel, de las que dependen del hardware. Al algunos ejemplos son C y C++.

Lenguaje de alto nivel o lenguajes evolucionados: son más cercanos al lenguaje natural por lo que obtiene mayor productividad y eficiencia. Se basan en estructuras de programación más intuitivas y requieren de un compilador o intérprete para convertirse en código máquina.

Ofrecen que se conozcan ciertas características de la máquina y lo representen en estos lenguajes respecto al lenguaje natural. Como son los lenguajes Phyton, Java, JavaScript y Ruby.

4.- ¿Para qué sirven los compiladores y los intérpretes? ¿En qué se diferencian?

Compiladores: realizan la traducción para el código fuente. De lenguaje máquina o código objeto en un proceso. De esa manera puede ser ejecutado independientemente como un archivo ejecutable. Ese archivo se podrá ejecutar tantas veces como se quiera. El programa una vez ejecutado necesitará ser traducido.

Intérpretes: sirven para traducir el código fuente línea por línea o en bloques y lo ejecutan sin generar un archivo ejecutable independiente. A medida que se va traduciendo, se ejecutan las instrucciones inmediatamente. Los resultados son mostrados al instante, pero suele ser más lento ya que el programa compilado tiene un rendimiento más lento.

Diferencia de ambos: el compilador es un proceso único por el cual se genera una sola vez. Su ejecución es independiente del código fuente ni el compilador para poder ser ejecutado. Y su velocidad es rápida porque el código ya está traducido a lenguaje máquina. Por otro lado el intérprete es un programa que lee instrucciones en el código fuente y las ejecuta línea por línea o en bloques. Depende de qué se necesite leer y traducir. Su velocidad dependerá de la traducción cuando ocurra el programa que se ejecuta.

5.- Explica y razona el tipo de lenguaje de programación menos empleado en la actualidad.

El lenguaje de programación a bajo nivel es el que está orientado a máquina y su nivel al que lo puede observar el ser humano es prácticamente nulo.

Está unido al hardware de un programa que normalmente no es el que transporta de una máquina a otra. Por el hecho de que está compuesto por 0 y 1. Lo que es más complicado para el ser humano. Por otro lado tiene un conjunto limitado de instrucciones ejecutables. En la que se almacena un programa ensamblador para que la máquina lo pueda entender. Como por ejemplo C, facilita la comprensión y el uso sin perder totalmente el control sobre el hardware.

6.- Definición de compilador e interprete. Indicar un lenguaje de programación compilado y otro interpretado (No indicar Java).

Compiladores: realizan un único proceso en el que analiza el código fuente. Este genera un código objeto y almacenar el resultado. El código objeto generado de la compilación dependiendo del compilador, se podrá ejecutar directamente o puede que necesiten utilizar otros pasos ejecutable como ensamblado, enlazado y carga. El código ejecutable podrá ejecutar tantas veces que desee sin necesidad de tener que volver a hacer el proceso de compilación. Como puede ser C++.

Intérpretes: simulan el proceso de traducción y ejecución. De manera que analizan los bloques del código fuente, generan el código objeto que le corresponde y lo ejecutan; después lo repiten. Se pasa el control al código objeto que se genera y espera que termine la ejecución. Como puede ser Phyton.

Diferencia de ambos: el compilador es un programa que convierte el código fuente escrito en un lenguaje de programación a lenguaje máquina. Por otro lado el intérprete es un programa que lee instrucciones en el código fuente y las ejecuta inmediatamente.

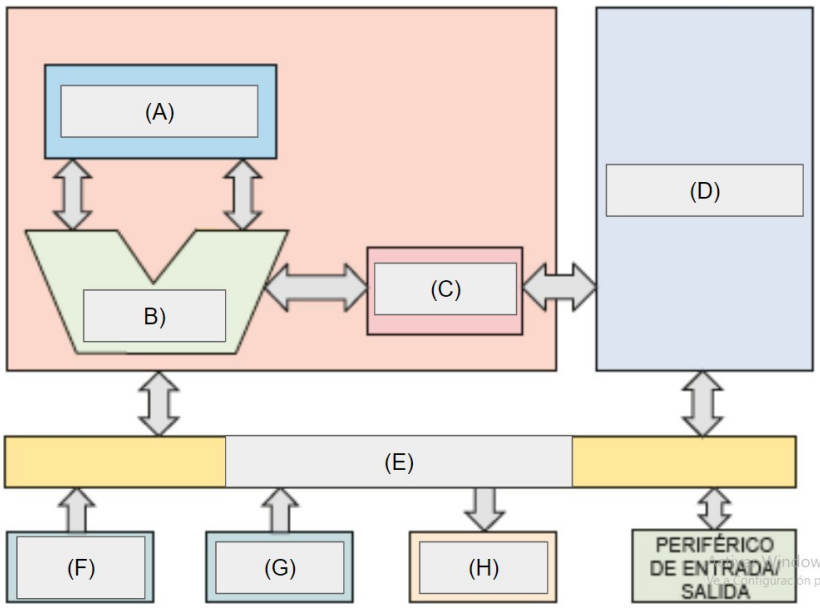
7.- ¿Qué dos tipos de máquinas virtuales existen y en qué se diferencian? ¿Qué tipo de máquina virtual es la máquina virtual de Java?

Máquinas virtuales de sistema: son las que pueden instalar en su interior otro sistema operativo con su propio disco duro, memoria,… En la que puede disponer de varios sistemas operativos en un mismo ordenador. Puede evaluar los nuevos sistemas operativos y probar aplicaciones en diferentes sistemas operativos.

Máquinas virtuales de proceso: son las que ejecutan un proceso dentro de otro sistema operativo. Este tipo de máquinas lanzan un proceso para ser ejecutado y detener cuando termine. Su finalidad es permitir al programa proporcionar un entorno independiente al hardware y el sistema operativo.

Máquina virtual de Java (JVM): es una máquina virtual de proceso la cual transporta el lenguaje, para que un programa sea compilado en Java y pueda ejecutarse en cualquier plataforma. Esto ocurre por el programa en el que está escrito en Java, es decir el entorno en desarrollo del que parte.

8.- Arquitectura de Von Neumann. Indica el nombre de los bloques principales.



A) Registros

B) ALU

C) Unidad de Control

D) Memoria Principal

E) Unidad Entrada Salida

F) Periférico de entrada

G) Periférico de entrada

H) Periférico de salida

9.- Indicar las fases del ciclo de vida en cascada.

1. Análisis
2. Diseño
3. Programación
4. Pruebas
5. Explotación
6. Mantenimiento

10.- Indica las características más relevantes de distinguen a los modelos de desarrollo ágil frente al resto de metodologías.

- Flexibilidad y adaptación a los cambios
- Desarrollo en ciclos cortos o iteraciones
- Colaboración con el cliente
- Documentación menos extensa
- Equipos autónomos y multifuncionariales
- Enfoque en la entrega continua