Avda. Arcadi Garcia, 1
12.600 La Vall d'Uixó
Tel. 964 738955
Mail: 12005751@gva.es
Web: iesbenigaslo.es

DAW, DAM .
ENTORNOS DE DESARROLLO

Unidad 0.

Ejercicio obligatorio. (Responded las siguientes cuestiones utilizando vuestras propias palabras**).**

1.- Explicad con vuestras palabras qué es un programa informático, qué son las instrucciones de un lenguaje, qué son los datos de un programa y qué relación existe entre estos tres conceptos.

Un programa informático es un conjunto de instrucciones lógicas escritas en un lenguaje de programación que le indican a una computadora que acciones debe realizar.
Estas instrucciones son como un conjunto de pasos que la computadora debe seguir para llevar a cabo tareas específicas.

Las instrucciones de un lenguaje son comandos que se utilizan para comunicarse con la computadora. Estos comandos están escritos en un lenguaje de programación comprensible tanto para los programadores humanos como para las maquinas. Los lenguajes de programación pueden ser un conjunto de reglas y palabras que permiten expresar las operaciones que se desean realizar en dicho programa.

Los datos de un programa son la información con la que el programa trabaja o sobra la que realiza las operaciones. Pueden ser números, texto, imágenes o cualquier tipo de información. Los datos son esenciales para que un programa realice tareas específicas, ya que las instrucciones del programa manipulan y procesan estos datos siguiendo la lógica de la programación.

La relación entre estos tres conceptos es lo más fundamental en la programación. Las instrucciones que se escriben en un lenguaje son el medio a través del cual se le dice a la computadora que tiene que hacer con los datos. El programa, que está compuesto por las instrucciones y los datos, trabaja en conjunto con ellos para realizar un proceso especifico y lograr un objetivo, ya sea realizar cálculos, procesar cierta información o realizar cualquier tarea definida por el programador.

Se podría resumir en que, un programa informático utiliza instrucciones para manipular datos y que estos realicen acciones específicas.

Avda. Arcadi Garcia, 1
12.600 La Vall d'Uixó
Tel. 964 738955
Mail: 12005751@gva.es
Web: iesbenigaslo.es

2.- Definid brevemente y utilizando vuestras palabras, qué es el «Código fuente», «Código objeto», «Código ejecutable» y el enlazador o «Linker».

CODIGO FUENTE: Es el conjunto de instrucciones escritas por un programador en un lenguaje de programación. Es legible para los humanos y necesita ser traducido a un formato ejecutable para que la computadora lo comprenda.

CODIGO OBJETO: Es el resultado de traducir el código fuente en un formato intermedio por un compilador. Este código no es ejecutable directamente, pero contiene la información necesaria para crear el código ejecutable.
CODIGO EJECUTABLE: Es el resultado final después de procesar el código objeto mediante un enlazador. Este código es la versión que la computadora puede ejecutar directamente para realizar las funciones programadas.

ENLAZADOR O "LINKER": Es una herramienta que combina y enlaza diferentes módulos de código objeto para crear un programa ejecutable. Se encargar de resolver las referencias entre módulos y enlazarlos para asegurar que todas las partes del programa estén integradas adecuadamente.

3.- Investiga los diferentes «tipos» de lenguajes de programación (1ª – 5ª generación), explica brevemente en qué consiste cada tipo e indica los lenguajes representativos de cada tipo.

1ª Generación – Lenguaje máquina:

- Descripción: Utiliza códigos numéricos que se interpretan directamente por la unidad central de procesamiento o CPU de la computadora en la que se ejecutan.
- Lenguajes: En este caso se utiliza el código binario más básico y difícil de leer por el humano. Estaríamos hablando de un lenguaje de bajo nivel
-

2ª Generación – Lenguajes ensambladores:

- Descripción: Estos lenguajes utilizan abreviaturas y representaciones mas entendibles por el humano que las instrucciones de bajo nivel, haciéndolas mas comprensibles que el lenguaje máquina.
- Lenguajes: Un ejemplo seria Assembly, que se utilizaría como lenguaje ensamblador, también hablaríamos en este caso de un lenguaje de bajo nivel.

3ª Generación – Lenguajes de Alto Nivel:

- Descripción: Se centran en la abstracción y la portabilidad, y sus instrucciones se asemejan bastante mas al lenguaje humano.
- Lenguajes: Aquí nos encontraríamos unos de los lenguajes mas conocidos como Java, C, Python, JavaScript, etc.

4ª Generación – Lenguajes de dominio especifico:

- Descripción: Estos lenguajes están diseñados para tareas específicas o dominios particulares, con mayor nivel de abstracción y facilidad de uso.
- Lenguajes: Aquí tendríamos el conocido lenguaje SQL, que se utiliza específicamente para bases de datos, también entrarían en este apartado otros lenguajes como PHP y MATLAB.

Avda. Arcadi Garcia, 1
12.600 La Vall d'Uixó
Tel. 964 738955
Mail: 12005751@gva.es
Web: iesbenigaslo.es

5ª Generación – Lenguajes basados en la logica:

- Descripción: Se centran en la resolución de problemas utilizando la lógica de programación declarativa.
- Lenguajes: Un ejemplo seria Prolog.

4.- Indicad la diferencia entre un compilador y un intérprete de órdenes. Citad un par de ejemplos de lenguajes interpretados.

Cuando hablamos de un compilador nos referimos a un programa informático que realiza la traducción del código fuente completo a código ejecutable en una fase que es única antes de la ejecución. Este programa genera un archivo ejecutable independiente que puede ejecutarse múltiples veces sin necesidad de recompilar, lo que habitualmente resulta mas eficiente en tiempo de ejecución, ya que el código ha sido traducido completamente antes de su ejecución.

En cambio, un intérprete de ordenes traduce y ejecuta el código en tiempo real línea por línea durante la ejecución del programa. Este no nos genera un archivo ejecutable independiente, lee y ejecuta el código directamente desde el código fuente. Esto nos proporciona flexibilidad al permitir realizar cambios en el código en tiempo real y ver los resultados inmediatamente, no cuenta con una fase de compilación anterior.

Python y JavaScript son ejemplos de lenguajes interpretados. Estos lenguajes permiten modificar y ejecutar el código directamente sin necesidad de un paso de compilación adicional anterior a la ejecución, lo que facilita experimentar y los hace mas interactivos.

5.- Explicad el concepto de máquina virtual aplicado a la programación y su relación con los bytecodes.

Cuando hablamos de una maquina virtual en programación estaríamos hablando de un “simulador” de una maquina dentro de otra. Esta maquina permite ejecutar programas escritos en un lenguaje especifico como si estuvieran corriendo en una maquina física, aunque en realidad están ejecutándose en la máquina virtual.

La maquina virtual utiliza los bytecodes como un puente entre el código fuente del programa y la ejecución en la maquina física. El programa se traduce en bytecodes y luego la maquina virtual los interpreta y los ejecuta. Esto permite que un mismo programa escrito en un lenguaje especifico pueda ejecutarse en diferentes plataformas, ya que la máquina virtual actúa como un intermediario que entiende los bytecodes independientemente del sistema que utilizen.

Avda. Arcadi Garcia, 1
12.600 La Vall d'Uixó
Tel. 964 738955
Mail: 12005751@gva.es
Web: iesbenigaslo.es

6.- Explicad con vuestras palabras los paradigmas de programación Estructurado, Funcional y Orientado a objetos. Intentad diferenciar en qué consiste cada uno, posibles ventajas e inconvenientes.

Programación Estructurada: El paradigma de programación estructurada se centra en la organización del código en estructuras lógicas y secuenciales. Utiliza control de flujo, como bucles y condicionales, para dirigir la ejecución del programa de manera clara y ordenada. La idea principal es dividir el programa en bloques más pequeños y manejables, fomentando la modularidad y la reutilización de código.

Ventajas: Facilita el mantenimiento del código, mejora la legibilidad y comprensión, y ayuda en la identificación de errores.

Inconvenientes: Puede ser limitado en la representación de ciertos problemas complejos y no siempre es la elección más natural para todos los escenarios.

Programación Funcional: La programación funcional trata los programas como evaluación de funciones matemáticas. Se centra en las funciones puras, que producen resultados basándose únicamente en sus entradas sin efectos secundarios. Se promueve el uso de funciones de orden superior, inmutabilidad y expresiones lambda.

Ventajas: Proporciona un enfoque declarativo, facilita la concurrencia y permite un código más compacto y elegante.

Inconvenientes: Puede requerir un cambio de mentalidad para quienes están acostumbrados a la programación imperativa. La inmutabilidad puede ser difícil de aplicar en ciertos contextos.

Programación Orientada a Objetos: En la programación orientada a objetos, los programas se modelan como conjuntos de objetos que interactúan entre sí. Cada objeto es una instancia de una clase y contiene datos y métodos. La herencia, la encapsulación y el polimorfismo son conceptos clave en este paradigma.

Ventajas: Mejora la modularidad, la reutilización del código y la escalabilidad del sistema. Refleja el mundo real de manera natural.

Inconvenientes: Puede resultar complejo para algunos problemas más simples y puede llevar a una jerarquía de clases demasiado compleja si no se gestiona adecuadamente.
En resumen, cada paradigma tiene sus propias fortalezas y debilidades. La elección del paradigma dependerá del problema a resolver y de las preferencias del programador. A menudo, una combinación de paradigmas se utiliza en proyectos más grandes para aprovechar lo mejor de cada enfoque.

7.- Explicad brevemente en qué consisten las diferentes fases de desarrollo de programas (Análisis, Diseño, Programación, pruebas y Mantenimiento

Análisis: La fase de análisis es el primer paso en el desarrollo de programas. Aquí, se identifican y comprenden los requisitos del sistema. Se analiza el problema que se va a resolver y se define qué funcionalidades debe tener el programa. El objetivo es entender completamente lo que se necesita antes de comenzar a diseñar o programar.

Diseño: En la fase de diseño, se planifica cómo se estructurará el programa para satisfacer los requisitos identificados en la fase de análisis. Se definen las estructuras de datos, la arquitectura del sistema y la interfaz de usuario. El diseño se centra en crear un plano detallado que sirva como guía para la implementación.

Programación: La fase de programación es donde se traduce el diseño en código fuente. Los desarrolladores escriben el código según las especificaciones del diseño. Es la fase en la que se crean las funciones y se implementan las estructuras de datos planificadas. La programación es el núcleo del desarrollo, donde el código se materializa.

Pruebas: Después de la programación, se lleva a cabo la fase de pruebas para asegurarse de que el programa funciona según lo previsto. Se realizan diferentes tipos de pruebas, como pruebas de unidad para componentes individuales, pruebas de integración para asegurar que los módulos funcionen bien juntos, y pruebas de sistema para evaluar el sistema completo. El objetivo es detectar y corregir errores antes de que el programa se despliegue.

Mantenimiento: La fase de mantenimiento ocurre después de que el programa está en funcionamiento. Involucra la corrección de errores, la introducción de mejoras y la adaptación a cambios en los requisitos del usuario. El mantenimiento garantiza que el programa siga siendo efectivo y útil a lo largo del tiempo, ajustándose a las necesidades cambiantes del entorno y los usuarios.

8.- Existen diferentes modelos de desarrollo de programas y estos cambian en función de la forma en la que se realizan las diferentes fases de desarrollo. Entre otras tenemos el desarrollo en cascada, en V y en espiral. Indicad para cada afirmación, el modelo al que se puede asociar de los tres citados anteriormente:

- Hasta que no termina una fase, no comienza la siguiente. **Modelo**: Cascada
- Un error de diseño en la etapa de pruebas requiere modificar el diseño del código y una nueva programación. **Modelo**: Cascada
- El programa se desarrolla en una serie de versiones incrementales. **Modelo**: Espiral
- Introduce dentro del modelo de desarrollo un análisis de riesgos asociados. **Modelo**:

  Espiral

Avda. Arcadi Garcia, 1
12.600 La Vall d'Uixó
Tel. 964 738955
Mail: 12005751@gva.es
Web: iesbenigaslo.es

Recursos:

http://www.tutorialspoint.com/sp/software_engineering/software_development_life_cycle.htm

http://modeloespiral.blogspot.com.es/

http://ingsoftware.weebly.com/ciclo-de-vida-en-v.html

Andreu Orenga Ramon 1ºDAW (tardes)