SEMANA 9 – PROGRAMACIÓN

• Programar = resolver problemas mediante algoritmos traducidos a código.
• Proceso: Analizar → Diseñar → Codificar → Probar → Documentar.
• Lenguajes compilados (C, C++, Java): mayor velocidad y rendimiento.
• Lenguajes interpretados (Python, JS): mayor flexibilidad y prueba rápida.
• Paradigmas:
• - Estructurado: uso de secuencias, condicionales y bucles. Fuerte en lógica básica.
• - Orientado a Objetos (POO): clases, objetos, atributos y métodos. Mantiene y organiza sistemas grandes.
• - Funcional: funciones puras, evita efectos secundarios, útil en análisis de datos.
• Buenas prácticas:
• - Código modular: divide en funciones/módulos.
• - Nombres claros en variables y funciones.
• - Comentarios útiles (no obvios).
• - Evitar duplicación de código (principio DRY).

SEMANA 10 – PRUEBAS DE SOFTWARE

• Objetivo: asegurar calidad y detectar errores antes de liberar el software.
• Verificación: ¿Se construyó correctamente según el diseño? (calidad interna)
• Validación: ¿Se construyó lo que el usuario realmente necesita? (calidad externa)
• Caja Negra: prueba entradas y salidas sin ver código interno. Se enfoca en requisitos funcionales.
• Caja Blanca: analiza la lógica interna, decisiones, branch y flujo del código.
• Tipos de pruebas relevantes:
• - Pruebas Funcionales: validan procesos y resultados.
• - Prueba de Instalación: confirma correcta instalación y funcionamiento inicial.
• - Prueba de Documentación: verifica claridad y coherencia del manual.
• Errores comunes si no se realizan pruebas:
• - Fallos en producción.
• - Pérdida de clientes.
• - Costos de mantenimiento elevados.

SEMANA 11 – MANTENIMIENTO DE SOFTWARE

• Mantenimiento = modificaciones después de la entrega.
• Razones: corregir fallos, adaptar a cambios, mejorar rendimiento.
• Tipos:
• - Correctivo: arreglar errores detectados.
• - Adaptativo: ajustar software a nuevos sistemas o tecnologías.
• - Perfectivo: añadir o mejorar características.
• - Preventivo: reestructurar antes de que aparezcan fallas.
• Problemas frecuentes en mantenimiento:
• - Código heredado difícil de entender.
• - Falta de documentación actualizada.
• - Cambios apresurados bajo presión.
• - Acumulación de "parches" reduce calidad y aumenta complejidad.
• Importante: El mantenimiento puede costar más que el desarrollo inicial.

RESUMEN FINAL PARA EXAMEN

• Programación: elegir el paradigma y lenguaje según el tipo de solución.
• Pruebas: se realizan para evitar errores costosos y asegurar calidad.
• Mantenimiento: mantiene vivo el software y asegura continuidad.
• Si puedes explicar Verificación vs Validación y los 4 tipos de mantenimiento, ya aseguraste puntos clave.