Universidad Tecnológica de Durango

Tecnologías de la Información

Fundamentos de Programación

Actividades

“Evidencias de Activiades y Tareas”

Alumnos:

- Barraza Torres Jesús Daniel

1°A BIS

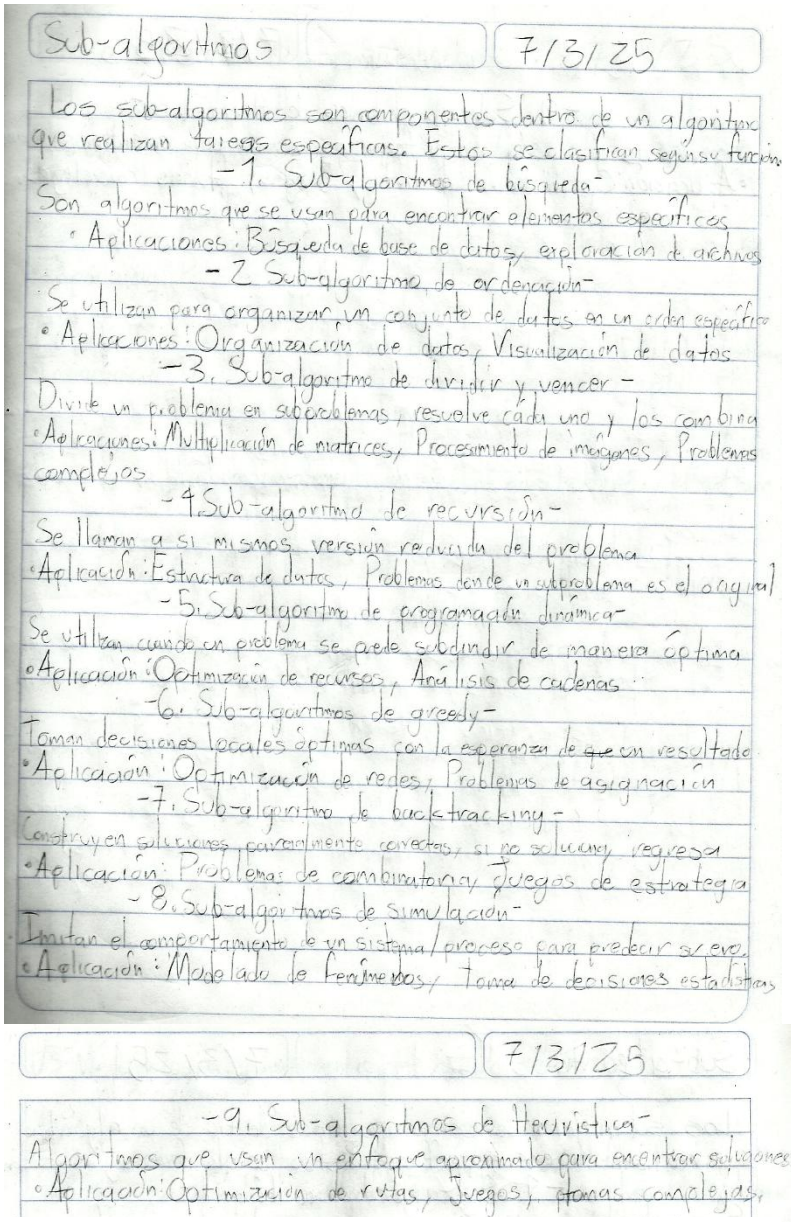
Docente:

- Ing. Dagoberto Fiscal Gurrola, M.T.I.

Marzo 2025

Tabla de Ilustraciones

Actividad 1


Sub-algoritmos 7/3/25

Los sub-algoritmos son componentes dentro de un algoritmo que realizan tareas específicas. Estos se clasifican según su función

-1. Sub-algoritmos de búsqueda-

Son algoritmos que se usan para encontrar elementos específicos

• Aplicaciones: Búsqueda de base de datos, exploración de archivos

-2 Sub-algoritmo de ordenación-

Se utilizan para organizar un conjunto de datos en un orden específico

• Aplicaciones: Organización de datos, Visualización de datos

-3. Sub-algoritmo de dividir y vencer -

Divide un problema en subproblemas, resuelve cada uno y los combina

• Aplicaciones: Multiplicación de matrices, Procesamiento de imágenes, Problemas complejos

-4 Sub-algoritmo de recursión-

Se llaman a si mismos versión reducida del problema

• Aplicación: Estructura de datos, Problemas donde un subproblema es el original

-5. Sub-algoritmo de programación dinámica-

Se utilizan cuando un problema se puede subdividir de manera óptima

• Aplicación: Optimización de recursos, Análisis de cadenas

-6. Sub-algoritmos de greedy-

Toman decisiones locales óptimas con la esperanza de que un resultado

• Aplicación: Optimización de redes, Problemas de asignación

-7. Sub-algoritmo de backtracking-

Construyen soluciones parcialmente correctas, si no solucionan regresan

• Aplicación: Problemas de combinatoria y juegos de estrategia

-8. Sub-algoritmos de simulación-

Imitan el comportamiento de un sistema/proceso para predecir su evo.

• Aplicación: Modelado de fenómenos, Toma de decisiones estadísticas

7/3/25

-9. Sub-algoritmos de Heurística-

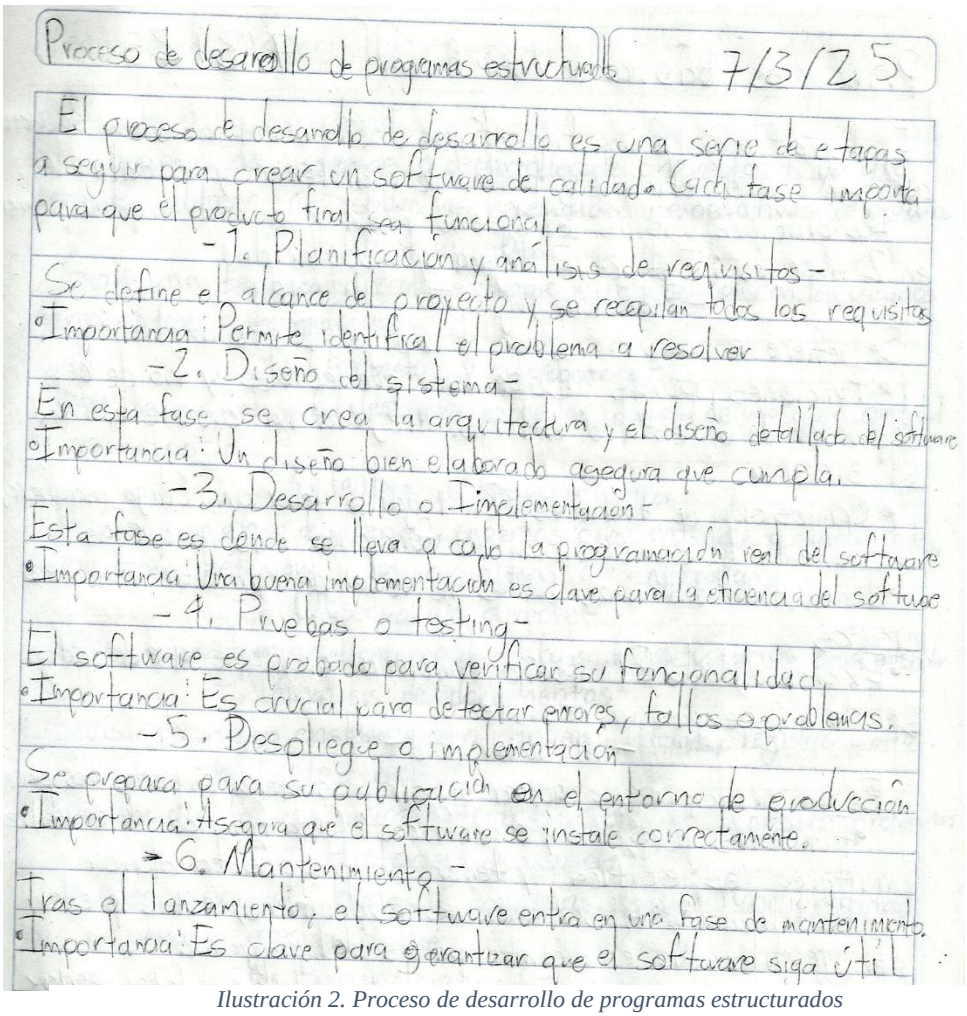
Algoritmos que usan un enfoque aproximado para encontrar soluciones

• Aplicación: Optimización de rutas, Juegos, Tareas complejas,

Ilustración 1. Sub-algoritmos

En esta actividad se muestran los 9 tipos de sub-algoritmos basados en su utilidad: Búsqueda, ordenación, dividir y vencer, recursión, programación dinámica, “greedy”, “backtracking”, simulación y heurística junto a sus aplicaciones.

Actividad 2


Proceso de desarrollo de programas estructurados 7/3/25

El proceso de desarrollo de desarrollo es una serie de etapas a seguir para crear un software de calidad. Cada fase importa para que el producto final sea funcional.

-1. Planificación y análisis de requisitos-

Se define el alcance del proyecto y se recopilan todos los requisitos

•Importancia: Permite identificar el problema a resolver

-2. Diseño del sistema-

En esta fase se crea la arquitectura y el diseño detallado del software

•Importancia: Un diseño bien elaborado asegura que cumpla.

-3. Desarrollo o Implementación-

Esta fase es donde se lleva a cabo la programación real del software

•Importancia: Una buena implementación es clave para la eficiencia del software

-4. Pruebas o testing-

El software es probado para verificar su funcionalidad

•Importancia: Es crucial para detectar errores, fallos o problemas.

-5. Despliegue o implementación

Se prepara para su publicación en el entorno de producción

•Importancia: Asegura que el software se instale correctamente.

-6. Mantenimiento-

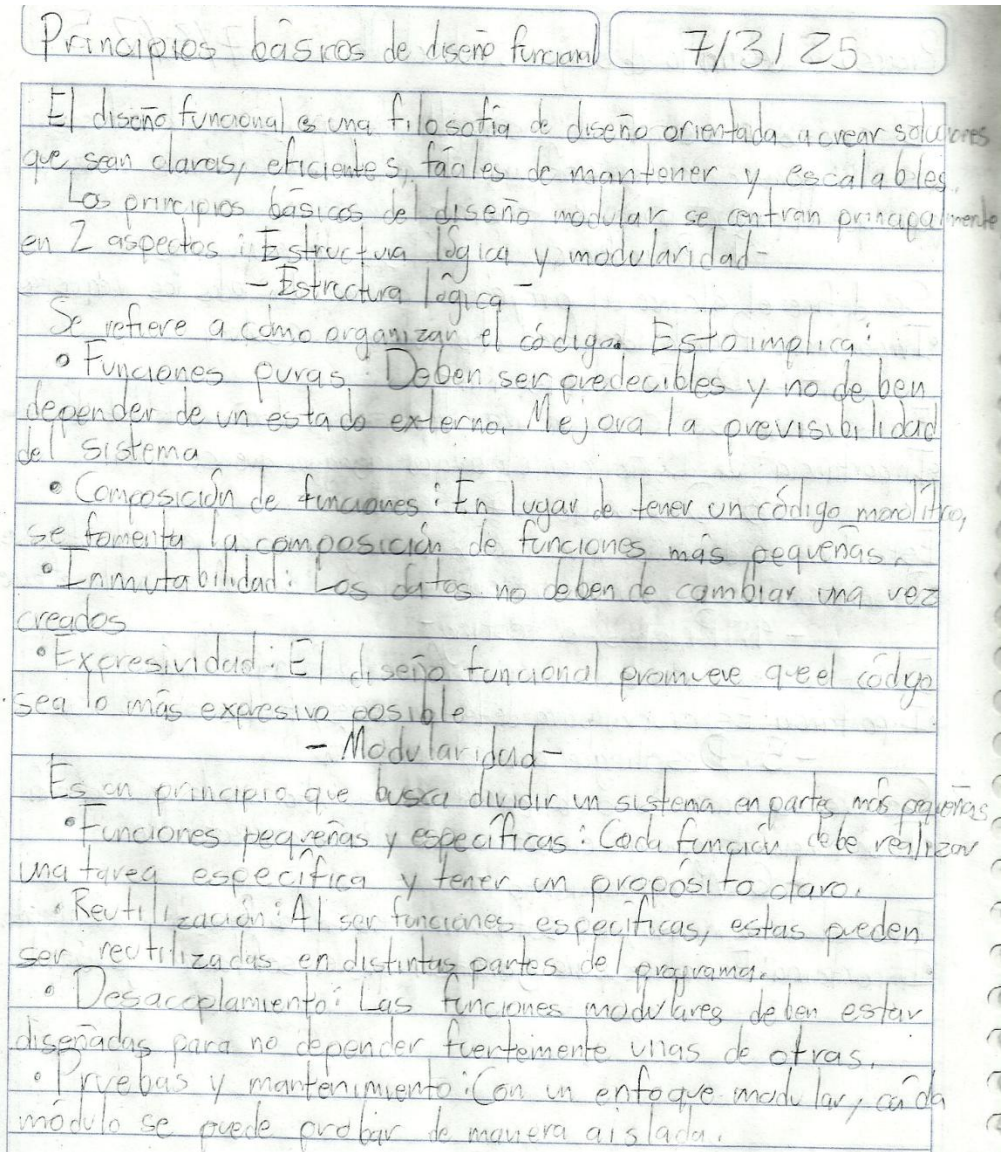
Tras el lanzamiento, el software entra en una fase de mantenimiento.

•Importancia: Es clave para garantizar que el software siga útil

Ilustración 2. Proceso de desarrollo de programas estructurados

En esta actividad se muestran las 6 fases del proceso del desarrollo del programa, los cuales son vitales para crear y mantener (en el caso del paso 6) un software de calidad. Muy usados dentro de las diversas industrias de la programación.

Actividad 3



Principios básicos de diseño funcional 7/3/25

El diseño funcional es una filosofía de diseño orientada a crear soluciones que sean claras, eficientes, fáciles de mantener y escalables.

Los principios básicos del diseño modular se centran principalmente en 2 aspectos: Estructura lógica y modularidad.

- Estructura lógica -

Se refiere a cómo organizar el código. Esto implica:

- Funciones puras: Deben ser predecibles y no deben depender de un estado externo. Mejora la previsibilidad del sistema
- Composición de funciones: En lugar de tener un código monolítico, se fomenta la composición de funciones más pequeñas.
- Inmutabilidad: Los datos no deben de cambiar una vez creados
- Expresividad: El diseño funcional promueve que el código sea lo más expresivo posible

- Modularidad -

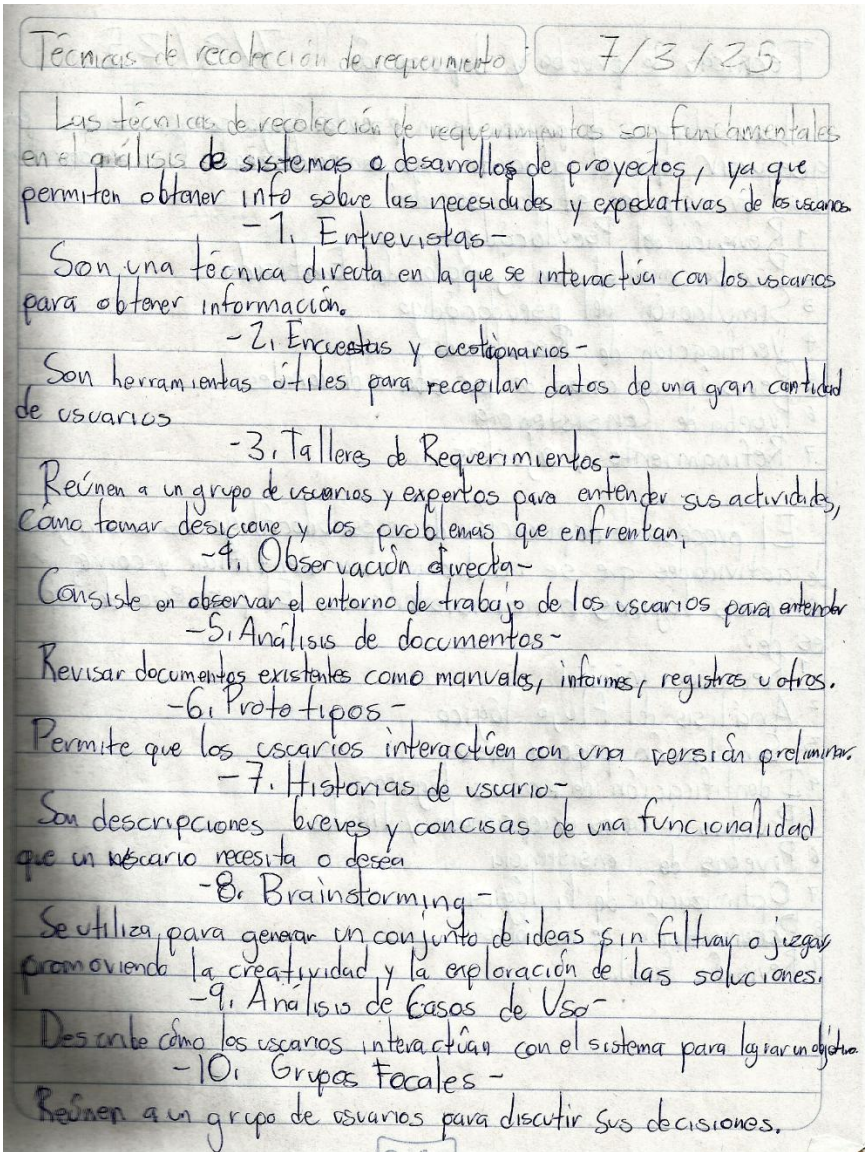
Es un principio que busca dividir un sistema en partes más pequeñas

- Funciones pequeñas y específicas: Cada función debe realizar una tarea específica y tener un propósito claro.
- Reutilización: Al ser funciones específicas, estas pueden ser reutilizadas en distintas partes del programa.
- Desacoplamiento: Las funciones modulares deben estar diseñadas para no depender fuertemente unas de otras.
- Pruebas y mantenimiento: Con un enfoque modular, cada módulo se puede probar de manera aislada.

Ilustración 3. Principios básicos del diseño funcional

En esta ilustración podemos ver los 2 principios del diseño funcional: Estructura lógica y modularidad. Esta ideología de construcción de diseños permite momentos de calidad de vida que dan bastante flexibilidad a un documento.

Actividad 4


Técnicas de recolección de requerimiento 7/3/25

Las técnicas de recolección de requerimientos son fundamentales en el análisis de sistemas o desarrollos de proyectos, ya que permiten obtener info sobre las necesidades y expectativas de los usuarios.

-1. Entrevistas-

Son una técnica directa en la que se interactúa con los usuarios para obtener información.

-2. Encuestas y cuestionarios-

Son herramientas útiles para recopilar datos de una gran cantidad de usuarios

-3. Talleres de Requerimientos-

Reúnen a un grupo de usuarios y expertos para entender sus actividades, cómo tomar desiciones y los problemas que enfrentan.

-4. Observación directa-

Consiste en observar el entorno de trabajo de los usuarios para entender

-5. Análisis de documentos-

Revisar documentos existentes como manuales, informes, registros u otros.

-6. Prototipos-

Permite que los usuarios interactúen con una versión preliminar.

-7. Historias de usuario-

Son descripciones breves y concisas de una funcionalidad que un usuario necesita o desea

-8. Brainstorming-

Se utiliza para generar un conjunto de ideas sin filtrar o juzgar promoviendo la creatividad y la exploración de las soluciones.

-9. Análisis de Casos de Uso-

Describe cómo los usuarios interactúan con el sistema para lograr un objetivo

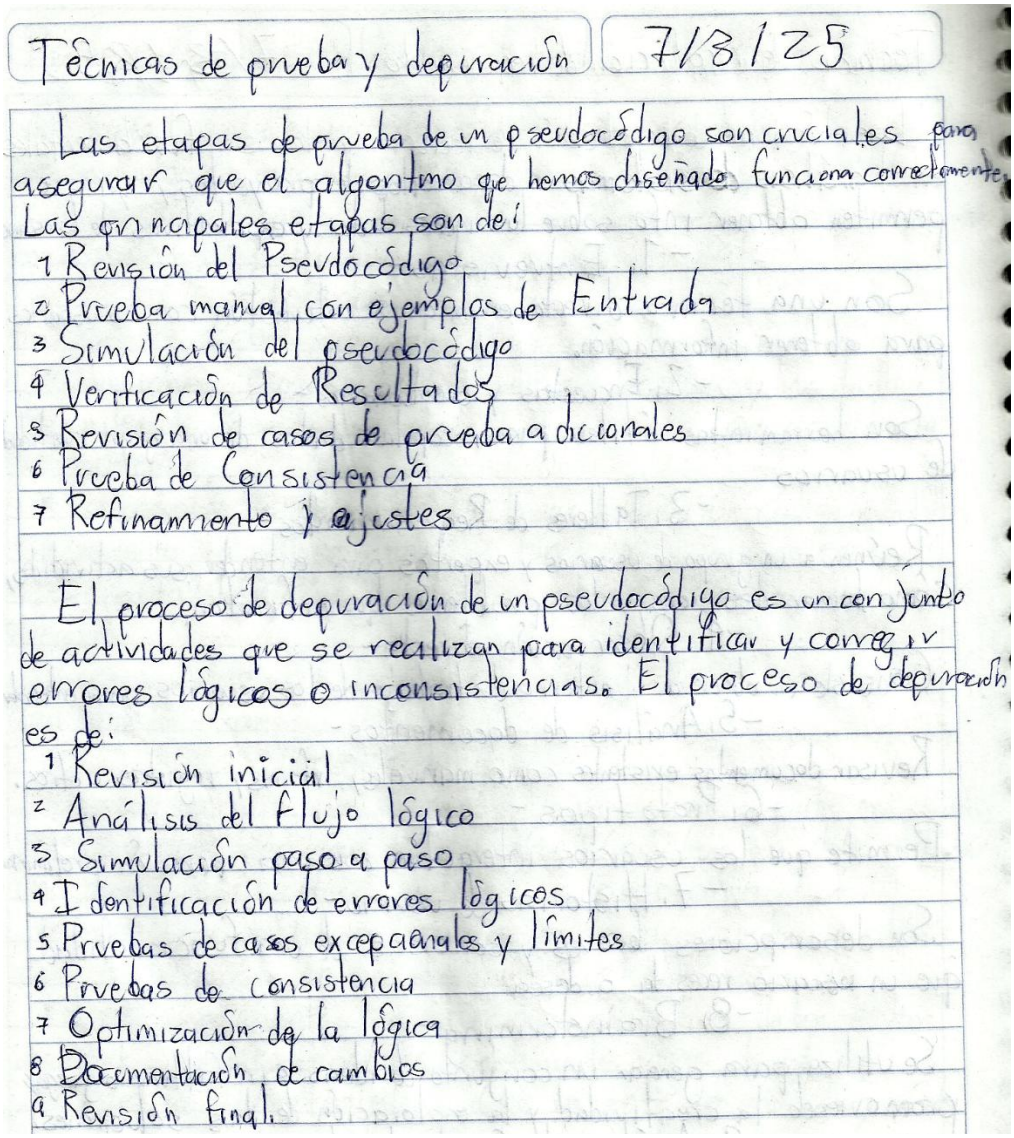
-10. Grupos Focales-

Reúnen a un grupo de usuarios para discutir sus decisiones.

Ilustración 4. Técnicas de recolección de requerimientos

En esta ilustración podemos encontrar las 10 técnicas de recolección de requerimientos los cuales sirven para poder encontrar una idea al momento de antes, durante o incluso después de realizar un código. Fundamental para poder crear algo del gusto del público.

Actividad 5



Técnicas de prueba y depuración 7/3/25

Las etapas de prueba de un pseudocódigo son cruciales para asegurar que el algoritmo que hemos diseñado funciona correctamente. Las principales etapas son de:

1 Revisión del Pseudocódigo
2 Prueba manual con ejemplos de Entrada
3 Simulación del pseudocódigo
4 Verificación de Resultados
5 Revisión de casos de prueba adicionales
6 Prueba de Consistencia
7 Refinamiento y ajustes

El proceso de depuración de un pseudocódigo es un conjunto de actividades que se realizan para identificar y corregir errores lógicos o inconsistencias. El proceso de depuración es de:

1 Revisión inicial
2 Análisis del flujo lógico
3 Simulación paso a paso
4 Identificación de errores lógicos
5 Pruebas de casos excepcionales y límites
6 Pruebas de consistencia
7 Optimización de la lógica
8 Documentación de cambios
9 Revisión final

Ilustración 5. Técnicas de prueba y depuración

En esta ilustración podemos encontrar los pasos de prueba y depuración de un pseudocódigo, diseñados específicamente para corregir errores y que el código sea funcional. Esto nos permite encontrar errores por arreglar y cosas por mejorar (en caso de optimización de archivos).

Retroalimentación

Este tema mencionado recapitula todo lo visto anteriormente en los 2 parciales anteriores y enriquece la información con pasos y metodologías útiles para poder revisar, corregir y armar pseudocódigos de calidad que permitan resolver de manera óptima los problemas futuramente proyectados.

En mi caso, si bien parecía que ya ando aplicando este tipo de datos debido a que es bastante lógico sirve como técnicas de pensamiento para lograr una metodología interesante e importante en temas óptimos.

En términos de nuevas ideas sin embargo, este tema no introduce nada nuevo, mas sin embargo utiliza lo ya conocido para poder armar estrategias de lo visto. Quizás a alguien que apenas vaya entrando al mundo de programación (como yo) pueda serle de utilidad esta información.