Guia de Resolucion

1. Tipos de Archivos en Pascal

Archivos de texto vs. binarios

- **Archivos de texto**:
  - Almacenan caracteres organizados en líneas.
  - Operaciones con conversión automática de tipos (ej: `ReadLn`, `WriteLn`).
  - Acceso secuencial (no permiten posicionamiento directo con `Seek`).
- **Archivos binarios**:
  - Almacenan registros de longitud fija (`File of <tipo>`).
  - Permiten acceso directo a registros mediante `Seek`.
  - Lectura/escritura sin conversión de tipos.

Operaciones Básicas

1. **Asignación (`Assign`)**: Vincular una variable de archivo con un archivo físico.
2. **Apertura**:
   - `Rewrite`: Crea un archivo nuevo (borra el existente).
   - `Reset`: Abre para lectura/escritura (posición inicial).
   - `Append`: Abre archivo de texto para añadir datos al final.
3. **Lectura/Escritura**:
   - `Read/ReadLn` (texto) o `Read` (binario).
   - `Write/WriteLn` (texto) o `Write` (binario).
4. **Cierre (`Close`)**: Libera recursos y guarda cambios.

---

2. Ejercicios Típicos y Pasos para Resolverlos

A. Creación de un archivo nuevo y escritura de datos

Pasos:

1. Asignar nombre al archivo con `Assign`.

2. Abrir en modo `Rewrite` (binario) o `Append` (texto).
3. Leer datos desde teclado o archivo de texto.
4. Escribir en el archivo hasta cumplir condición de fin (ej: entrada "zzz").
5. Cerrar el archivo.

   Ejemplo práctico: Crear un archivo de materiales de construcción con nombres ingresados por el usuario.

B. Lectura y muestra de contenido por pantalla

Pasos:

1. Abrir el archivo con `Reset`.
2. Leer registros secuencialmente con `Read`.
3. Mostrar datos en pantalla (ej: listar votantes de una provincia).
4. Cerrar el archivo.

   Consideración clave: Usar bucles `while not Eof(archivo)` para recorrer todo el contenido.

C. Búsqueda de registros específicos (ID, nombre)

Pasos:

1. Abrir el archivo con `Reset`.
2. Leer registros uno a uno hasta encontrar el dato buscado.
3. Validar coincidencia (ej: comparar código o nombre).
4. Cerrar el archivo.

   Nota: En archivos binarios ordenados, optimizar con búsqueda binaria usando `Seek`.

D. Modificación/Actualización de registros existentes

Pasos:

1. Abrir el archivo con `Reset`.
2. Buscar el registro a modificar.
3. Sobrescribir el registro con nuevos datos usando `Seek` y `Write`.

4. Cerrar el archivo.

   Ejemplo: Actualizar el stock de productos vendidos en un archivo maestro.

E. Fusión de archivos (Merge)

Pasos:

1. Abrir todos los archivos detalles y el maestro.
2. Leer registros de cada detalle secuencialmente.
3. Comparar claves (ej: código de producto) y consolidar datos (sumar cantidades).
4. Escribir el resultado en el archivo maestro.
5. Cerrar todos los archivos.

   Ejemplo: Generar un archivo maestro de ventas a partir de 3 detalles.

F. Manejo de Errores

- **Archivo no encontrado**: Verificar existencia antes de abrir con `Reset`.
- **Permisos insuficientes**: Usar bloques `try-except` (si el lenguaje lo permite).
- **Validación de datos**: Chequear formato al leer desde texto (ej: números vs. letras).

---

3. Algoritmos Clásicos

A. Actualización Maestro-Detalle

Pasos:

1. Abrir maestro y detalles (ordenados por clave).
2. Leer registro actual del maestro y detalles.
3. Comparar claves:
   - Si coinciden, actualizar maestro (ej: restar stock).
   - Si no, avanzar en el archivo correspondiente.
4. Repetir hasta procesar todos los detalles.

B. Corte de Control

Pasos:

1. Abrir archivo ordenado (ej: por provincia y localidad).
2. Leer registros agrupando por clave (ej: provincia).
3. Acumular totales por grupo y mostrar subtotales.
4. Mostrar total general al final.

   Ejemplo: Contabilizar hogares por provincia y localidad.

C. Bajas Físicas vs. Lógicas

- **Baja física**:
  - Copiar el último registro al lugar del eliminado y truncar el archivo.
- **Baja lógica**:
  - Marcar registro (ej: campo `descripción = "@"`).
  - Usar lista invertida para reutilizar espacio (registro 0 como cabecera).

---

4. Consideraciones Importantes

- **Manejo de punteros**: Usar `FilePos` y `Seek` para navegar en binarios.
- **Bucles**: `While not Eof(archivo)` para recorridos secuenciales.
- **Validación**: Chequear datos antes de escribir (ej: evitar registros corruptos).
- **Eficiencia**: En archivos grandes, preferir acceso directo si están ordenados.