Manual Técnico para el Sistema de Gestión de Inventarios en Fortran

Introducción

Este manual técnico describe la estructura y lógica del programa desarrollado en Fortran para gestionar un inventario de equipos de oficina. El programa lee y procesa archivos de texto que contienen instrucciones para la creación y actualización del inventario, y genera un informe final con el estado del inventario.

1. Estructura del Programa

El programa se organiza en módulos y subrutinas para manejar distintas tareas como la lectura de archivos, el procesamiento de datos y la generación de informes. A continuación, se describe la estructura principal:

- **Módulo Principal:** Contiene el menú principal y las llamadas a las subrutinas correspondientes.
- **Subrutinas:**
    - cargar_inventario_inicial
    - procesar_movimientos
    - generar_informe

Cada subrutina tiene un propósito específico y está diseñada para manejar una parte particular del proceso de gestión del inventario.

2. Descripción de la Lógica

2.1 Menú Principal

El programa inicia presentando un menú principal que permite al usuario seleccionar la operación que desea realizar. Dependiendo de la opción seleccionada, se invoca la subrutina correspondiente.

```
SUBROUTINE menu_principal()
  ! Presenta el menú y gestiona la selección del usuario
  PRINT *, 'Seleccione una opción:'
  PRINT *, '1. Cargar Inventario Inicial'
```

```
 PRINT *, '2. Cargar Instrucciones de Movimientos'
 PRINT *, '3. Crear Informe de Inventario'
 PRINT *, '4. Salir'
 READ *, opcion
 SELECT CASE(opcion)
 CASE(1)
  CALL cargar_inventario_inicial()
 CASE(2)
  CALL procesar_movimientos()
 CASE(3)
  CALL generar_informe()
 CASE(4)
  STOP
 END SELECT
END SUBROUTINE
```

2.2 Cargar Inventario Inicial

La subrutina cargar_inventario_inicial lee un archivo .inv línea por línea. Cada línea se divide en tokens utilizando los delimitadores definidos (;) y los datos se almacenan en una estructura de datos adecuada (como un arreglo o una lista enlazada).

```
SUBROUTINE cargar_inventario_inicial()
 OPEN(UNIT=10, FILE='inventario.inv', STATUS='OLD')
 DO WHILE (.NOT. EOF(10))
  READ(10, '(A)') linea
  CALL procesar_linea_inventario(linea)
 END DO
 CLOSE(10)
END SUBROUTINE
```

```
SUBROUTINE procesar_linea_inventario(linea)
  ! Divide la línea en tokens y almacena los datos en la estructura de inventario
  CALL dividir_linea(linea, nombre, cantidad, precio, ubicacion)
  ! Aquí se almacena el equipo en la estructura de datos correspondiente
END SUBROUTINE
```

2.3 Procesar Movimientos

La subrutina procesar_movimientos sigue una lógica similar para leer un archivo .mov. Dependiendo de la instrucción (agregar_stock o eliminar_equipo), se actualiza la cantidad en el inventario.

```
SUBROUTINE procesar_movimientos()
  OPEN(UNIT=11, FILE='movimientos.mov', STATUS='OLD')
  DO WHILE (.NOT. EOF(11))
    READ(11, '(A)') linea
    CALL procesar_linea_movimiento(linea)
  END DO
  CLOSE(11)
END SUBROUTINE


SUBROUTINE procesar_linea_movimiento(linea)
  ! Identifica la instrucción y la procesa adecuadamente
  CALL dividir_linea(linea, nombre, cantidad, ubicacion)
  IF (instruccion == 'agregar_stock') THEN
    ! Código para agregar stock
  ELSE IF (instruccion == 'eliminar_equipo') THEN
    ! Código para eliminar equipo
  END IF
```

```
END SUBROUTINE
```

2.4 Generar Informe de Inventario

La subrutina generar_informe crea un archivo de texto informe.txt que contiene el estado actual del inventario. Cada entrada del inventario se escribe en el archivo con el formato especificado.

```
SUBROUTINE generar_informe()
 OPEN(UNIT=12, FILE='informe.txt', STATUS='NEW')
 WRITE(12, '(A)') 'Equipo      Cantidad    Precio Unitario   Valor Total    Ubicación'
 DO i = 1, numero_de_equipos
  ! Escribe cada equipo y sus detalles en el archivo
  WRITE(12, '(A, I5, F8.2, F10.2, A)') equipo(i)%nombre, equipo(i)%cantidad, &
                                       equipo(i)%precio_unitario, &
                                       equipo(i)%valor_total, equipo(i)%ubicacion
 END DO
 CLOSE(12)
END SUBROUTINE
```

3. Estructuras de Datos

El programa utiliza estructuras de datos para almacenar la información del inventario. Por ejemplo, se puede definir un tipo de datos equipo_t para almacenar las características de cada equipo.

fortran

```
TYPE equipo_t
 CHARACTER(LEN=20) :: nombre
 INTEGER :: cantidad
 REAL :: precio_unitario
 REAL :: valor_total
 CHARACTER(LEN=10) :: ubicacion
```

END TYPE

TYPE(equipo_t), ALLOCATABLE :: inventario(:)

4. Manejo de Errores

El programa incluye un manejo básico de errores, como verificar si un equipo existe antes de actualizar su stock y validar las cantidades antes de eliminarlas.

5. Restricciones

El código ha sido diseñado sin el uso de expresiones regulares, implementando funciones propias para la manipulación de cadenas y la extracción de tokens.

Conclusión

Este manual técnico ofrece una visión detallada de la lógica del programa, desde la estructura del código hasta el manejo de datos y errores. Con esta información, cualquier desarrollador con conocimientos básicos de Fortran debería ser capaz de entender y modificar el programa según sea necesario.

4o