

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

LENGUAJES FORMALES Y DE PROGRAMACIÓN

CATEDRÁTICO: ING. DAVID MORALES

AUXILIAR: HERBERTH ABISAI AVILA RUIZ



MANUAL TÉCNICO

DANIEL ANDREÉ HERNANDEZ FLORES

CARNÉ: 202300512

SECCIÓN: B+

GUATEMALA, 22 DE AGOSTO DEL 2,024

ÍNDICE

CONTENIDO

ÍNDICE	1
INTRODUCCIÓN	2
ESTRUCTURA DE MÓDULOS Y DEL PROGRAMA.....	3
1. MÓDULO “InventarioMod”	3
2. ESTRUCTURA DEL PROGRAMA PRINCIPAL	3
SUBROUTINAS Y FUNCIONES CLAVE.....	4
“RedimensionarArrays”	4
“ExtraerTokens”	4
“ExtraerTokensMov”	5
“MostrarMenu”	5
“CargarInventario”	6
“CargarMovimientos”	6
“CrearInforme”	7
MANEJO DE ARCHIVOS.....	8
• APERTURA DE ARCHIVOS	8
• LECTURA DE ARCHIVOS	8
• ESCRITURA DE ARCHIVOS	8
GESTIÓN DE MEMORIA.....	9
ASIGNACIÓN DINÁMICA DE MEMORIA	9
LIBERACIÓN DE ARRAYS.....	9
INTERACCIÓN CON EL USUARIO	10

INTRODUCCIÓN

Este programa está diseñado para gestionar un inventario de equipos, permitiendo cargar inventarios iniciales, registrar movimientos de stock, y generar informes. Utiliza Fortran, un lenguaje de programación eficiente para cálculos numéricos y operaciones con datos.

El objetivo principal es proporcionar una herramienta sencilla pero poderosa para la administración de inventarios, que pueda manejar dinámicamente el almacenamiento de datos en arrays y permitir interacciones fáciles con el usuario a través de un sistema de menús.

ESTRUCTURA DE MÓDULOS Y DEL PROGRAMA

1. MÓDULO “InventarioMod”

Este módulo contiene todas las variables globales y subrutinas esenciales para la gestión del inventario. Variables Globales: numEquipos, nombres, ubicaciones, cantidades, precios, inventarioCargado. Entre las principales componentes del módulo se encuentran:

- 1.1. numEquipos: Un entero que mantiene el número de equipos registrados.
- 1.2. nombres, ubicaciones: Arrays dinámicos que almacenan los nombres y ubicaciones de los equipos.
- 1.3. cantidades: Array que guarda la cantidad disponible de cada equipo.
- 1.4. precios: Array que contiene el precio unitario de cada equipo.
- 1.5. inventarioCargado: Un valor lógico que indica si el inventario ya ha sido cargado.

2. ESTRUCTURA DEL PROGRAMA

El programa principal (PROGRAM Inventario) es responsable de manejar la interacción con el usuario a través de un menú principal. Dependiendo de la opción seleccionada por el usuario, llama a las subrutinas correspondientes para cargar el inventario, procesar movimientos o generar un informe.

3. INFRAESTRUCTURA DEL PROGRAMA

- El programa fue desarrollado en un equipo con sistema operativo Windows 11
- Se desarrolló en el IDE de Visual Studio Code
- Usando la versión de Fortran: GNU Fortran (GCC) 14.1.0

SUBROUTINAS Y FUNCIONES CLAVE

“RedimensionarArrays”

Esta subrutina se encarga de redimensionar dinámicamente los arrays que almacenan los datos del inventario cuando se necesita más espacio. Primero, crea arrays temporales con el nuevo tamaño, copia los datos existentes a estos nuevos arrays, y luego libera la memoria de los arrays originales antes de asignar los temporales.

```
1 SUBROUTINE RedimensionarArrays(nombres, ubicaciones, cantidades, precios, newSize)
2     ! Redimensiona los arrays de nombres, ubicaciones, cantidades, y precios.
3     ! ... (código de la subrutina)
4 END SUBROUTINE RedimensionarArrays
```

“ExtraerTokens”

Esta subrutina toma una línea de texto de un archivo de inventario y la divide en sus componentes básicos: nombre del equipo, cantidad, precio unitario, y ubicación. Se utiliza en CargarInventario para procesar cada línea del archivo de inventario.

```
1 SUBROUTINE ExtraerTokens(linea, nombre, cantidad, precio_unitario, ubicacion)
2     ! Extrae los tokens de una línea del archivo de inventario.
3     ! ... (código de la subrutina)
4 END SUBROUTINE ExtraerTokens
```

“ExtraerTokensMov”

Similar a ExtraerTokens, pero diseñada para manejar líneas de un archivo de movimientos, donde se procesan las acciones como agregar stock o eliminar equipos.

```
1 SUBROUTINE ExtraerTokensMov(linea, accion, nombre, cantidad, ubicacion)
2     ! Extrae los tokens de una línea del archivo de movimientos.
3     ! ... (código de la subrutina)
4 END SUBROUTINE ExtraerTokensMov
```

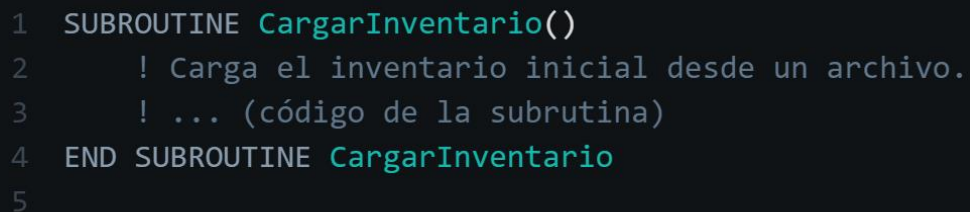
“MostrarMenu”

Muestra las opciones del menú principal al usuario, permitiéndole seleccionar una acción.

```
1 SUBROUTINE MostrarMenu()
2     ! Muestra el menú principal al usuario.
3     ! ... (código de la subrutina)
4 END SUBROUTINE MostrarMenu
5
```

“CargarInventario”

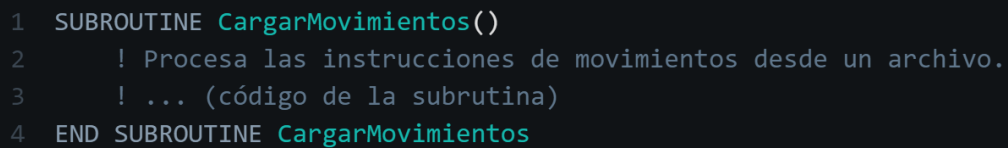
Carga el inventario desde un archivo .inv, leyendo cada línea y almacenando los datos en arrays dinámicos. También verifica si ya se ha cargado un inventario para evitar duplicaciones.



```
1 SUBROUTINE CargarInventario()  
2     ! Carga el inventario inicial desde un archivo.  
3     ! ... (código de la subrutina)  
4 END SUBROUTINE CargarInventario  
5
```

“CargarMovimientos”

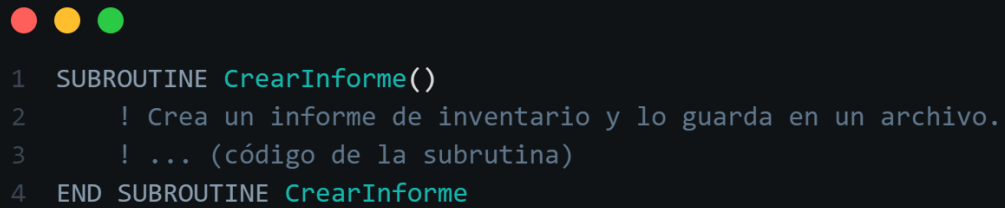
Procesa un archivo de movimientos (.mov), aplicando las acciones especificadas (como agregar stock o eliminar equipos) a los datos de inventario ya cargados.



```
1 SUBROUTINE CargarMovimientos()  
2     ! Procesa las instrucciones de movimientos desde un archivo.  
3     ! ... (código de la subrutina)  
4 END SUBROUTINE CargarMovimientos
```

“CrearInforme”

Genera un informe detallado del inventario actual, escribiendo la información en un archivo de texto (Reporte.txt), incluyendo el nombre del equipo, cantidad, precio unitario, valor total y ubicación.



```
1 SUBROUTINE CrearInforme()  
2     ! Crea un informe de inventario y lo guarda en un archivo.  
3     ! ... (código de la subrutina)  
4 END SUBROUTINE CrearInforme
```


MANEJO DE ARCHIVOS

- **APERTURA DE ARCHIVOS**

El programa utiliza la instrucción OPEN para abrir archivos necesarios para la lectura o escritura. Se emplea un sistema de control de errores (iostat) para manejar posibles problemas como archivos inexistentes o errores de acceso.

- **LECTURA DE ARCHIVOS**

Se utiliza READ para leer líneas del archivo y procesarlas. La subrutina ExtraerTokens se encarga de dividir estas líneas en datos útiles.

- **ESCRITURA DE ARCHIVOS**

La subrutina CrearInforme emplea la instrucción WRITE para escribir el informe de inventario en un archivo de texto.

GESTIÓN DE MEMORIA

ASIGNACIÓN DINÁMICA DE MEMORIA

Los arrays que almacenan los datos del inventario son dinámicos, es decir, su tamaño puede cambiar en tiempo de ejecución. Esto se maneja mediante las subrutinas `ALLOCATE` y `DEALLOCATE`.

LIBERACIÓN DE ARRAYS

Antes de redimensionar un array, el programa libera la memoria asignada al array anterior usando `DEALLOCATE` para evitar fugas de memoria.

INTERACCIÓN CON EL USUARIO

El usuario interactúa con el programa a través de un menú principal que se muestra en la consola. El menú permite seleccionar entre cargar inventarios, registrar movimientos o generar informes, lo que proporciona una interfaz sencilla para la administración del inventario.