Informe tecnico

esto se usa para llamar el modulo a utilizar importando de esta manera las funciones del modulo

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESTRUCTURA DE DATOS

CATEDRÁTICO: ZULMA AGUIRRE

TUTOR ACADÉMICO:JONATAN LEONEL GARCIA ARANA

Natalia Suceli Garrido Montenegro CARNÉ: 202201385

INDICE

Introduccion

Se presenta un programa donde el se desea conocer ante todo el funcionamiento del lenguaje de Fortran como sus funciones basicas usadas en este programa y por que se uso tal programa. tambien veremos el manejo de distintos tipos de archivos, tanto su lectura como su escritura.

Objetivos

general

- entender conceptos basicos del lenguaje de fortran

especifico

- poder leer y escribir distintos tipos de archivos
- entender el funcionamiento de herramientas básicas de fortran

Requisitos del sistema

hardware

- computadora suficiente potente para ejecutar un software de desarrollo asi como un procesador moderno y buen almacenamiento
- pantalla de buena calidad y buen tamaño

software

- IDE
- sistema operativo que soporte la IDE y sea compatible en este caso el uso es de vsCode
- tener el compilador de Fortran

DESCRIPCIÓN DE LA SOLUCIÓN

- primero se creo un menu de manera recursiva
- se habia planteado la implementacion de varios modulos para la creacion del programa asi teniendo mas ordenado el codigo pero por temas de compilacion se prefirio trabajar solo un modulo
- primero se trabajo la lectura de archivo de inventario. guardandolo en un array
- luego se creo un imprimirInventario con el fin de ir viendo y confirmando como se trabajan los datos en el inventario
- el leer archivos de movimientos fue mas complicado por el hecho que habia que segun el comando se debia enviar a diferente lado. y con la reutlizicacion de el separar datos

Main

```
PROGRAM Main

use cargarInventarios

IMPLICIT NONE

TYPE(Inventario), DIMENSION(:), ALLOCATABLE :: listInventario

INTEGER :: opcion

CHARACTER(LEN=100) :: filename

CALL menu()

CONTAINS

RECURSIVE SUBROUTINE menu()

PRINT *, '1. Cargar inventario'

PRINT *, '2. Cargar movimientos'

PRINT *, '3. Mostrar inventario'

PRINT *, '4. Salir'

READ(*,*) opcion

SELECT CASE (opcion)

CASE (1)

PRINT *, 'Escriba la ruta'

READ *, filename
```

```
CALL leerInventario(filename,listInventario)

CALL menu()

CASE (2)

PRINT *,'escriba la ruta de  los movimientos'

READ *, filename

call leerCambios(filename,listInventario)

CALL menu()

CASE (3)

PRINT *, 'Se imprime lista'

call imprimirInventario(listInventario)

CALL menu()

CASE (4)

PRINT *, 'Saliendo del programa.'

CASE DEFAULT

PRINT *, 'Opción inválida, por favor ingrese un número válid
o.'

CALL menu()

END SELECT
```

```
END SUBROUTINE menu

END PROGRAM main
```

siendo este el codigo de main entraremos mas a profundidad a la parte posterior del codigo explicando paso a paso ya que la otra parte se basa a como se imprimio el menu eso si tener en cuenta que para usar recursividad en fortran se utiliza el `RECURSIVE SUBROUTINE`

```
use cargarInventarios
```

esto se usa para llamar el modulo a utilizar importando de esta manera las funciones del modulo

```
TYPE(Inventario), DIMENSION(:), ALLOCATABLE :: listInventario
```

aca se crea como un arraylist de tipo Inventario llamado listInventario

El `ALLOCATABLE` es una característica que permite a un arreglo o matriz ser asignado dinámicamente en memoria durante la ejecución del programa

MODULO [cargarInventarios]

```
MODULE cargarInventarios
    TYPE :: Inventario
        CHARACTER(LEN=50) :: nombre
        INTEGER :: cantidad
        REAL :: precio
        CHARACTER(LEN=50) :: ubicacion
    END TYPE Inventario
```

```
    CONTAINS
```

en esta parte de aca podemos ver el modulo cargar Inventarios se crea un objeto de este tipo

el `TYPE` sirve para crear un *datos compuestos* de tipo inventario

ahora el codigo contiene varios subroutines como lo es

leer Inventario

```
SUBROUTINE leerInventario(filename, listInventario)
        TYPE(Inventario), DIMENSION(:), ALLOCATABLE :: listIn
ventario
        CHARACTER(LEN=*), INTENT(IN) :: filename
        INTEGER :: i, ios, count, pos
        CHARACTER(LEN=50) :: line
        CHARACTER(LEN=50) :: dataTemp(4)
        CHARACTER(LEN=50) :: comando


        count = 0
        OPEN(UNIT=10, FILE=filename, STATUS='OLD', ACTION='RE
AD', iostat=ios)
        IF (ios /= 0) THEN
            PRINT *, 'Error al abrir el archivo'
            STOP
        END IF


        DO
            READ(10, '(A)', IOSTAT=ios) line
            IF (ios /= 0) EXIT
```

```
            count = count + 1
        END DO


        REWIND(10)
        ALLOCATE(listInventario(count))


        DO i = 1, count
            READ(10, '(A)') line


            pos = INDEX(line, ' ')
            comando = line(1:pos-1)
            line = line(pos+1:)

            CALL separarDatos(line,4,dataTemp)


            listInventario(i)%nombre = TRIM(dataTemp(1))
            READ(dataTemp(2), *) listInventario(i)%cantidad
            READ(dataTemp(3), *) listInventario(i)%precio
            listInventario(i)%ubicacion = TRIM(dataTemp(4))
        END DO
        print*, 'se leyo el inventario de entrada'

        CLOSE(10)
    END SUBROUTINE leerInventario
```

aca se muestra el codigo completo pero lo explicare por martes

- al principio se muesta la asignacion de variables

```
  OPEN(UNIT=10, FILE=filename, STATUS='OLD', ACTION='READ', i
ostat=ios)
  !aca se presenta en caso que no se pueda leer el archivo
```

```
        IF (ios /= 0) THEN
            PRINT *, 'Error al abrir el archivo'
            STOP
        END IF


        DO
            READ(10, '(A)', IOSTAT=ios) line
            IF (ios /= 0) EXIT
            count = count + 1
        END DO


        REWIND(10)
        !aca se asigna que el tama;o de listInventario sera i
gual al numero de lineas contadas
        ALLOCATE(listInventario(count))
```

en la otra parte de la subrutina en esta parte se van llenando los datos en el listInventario

```
 DO i = 1, count
            READ(10, '(A)') line


            pos = INDEX(line, ' ')
            comando = line(1:pos-1)
            line = line(pos+1:)

            CALL separarDatos(line,4,dataTemp)


            listInventario(i)%nombre = TRIM(dataTemp(1))
            READ(dataTemp(2), *) listInventario(i)%cantidad
```

```
            READ(dataTemp(3), *) listInventario(i)%precio
            listInventario(i)%ubicacion = TRIM(dataTemp(4))
        END DO
        print*, 'se leyo el inventario de entrada'

        CLOSE(10)
```

se retira primero el comando inicial con un `INDEX` , este funciona de manera de un buscador de caracteres aca se llama a otra subrutina pasando la linea que se trabaja en ese momento para poder separarla en un vector y esta ir guardandola en el “arraylist”

separar datos

```
SUBROUTINE separarDatos(line, numCampos, dataTemp)
        CHARACTER(LEN=*), INTENT(IN) :: line
        INTEGER, INTENT(IN) :: numCampos
        CHARACTER(LEN=50), DIMENSION(:), INTENT(OUT) :: dataT
emp
        CHARACTER(LEN=256) :: local_line
        INTEGER :: pos, j


        local_line = line


        DO j = 1, numCampos
            dataTemp(j) = ''
        END DO


        DO j = 1, numCampos
            pos = INDEX(local_line, ';')
            IF (pos > 0) THEN
                dataTemp(j) = TRIM(local_line(1:pos-1))
```

```
                local_line = local_line(pos+1:)
            ELSE
                dataTemp(j) = TRIM(local_line)
                EXIT
            END IF
        END DO
    END SUBROUTINE separarDatos
```

esta subrutina esta encargada de ir separando los datos por cada (;) que encuentre y guardandolo en un vector temporal

leer cambios

```
SUBROUTINE leerCambios(filename, listInventario)
        CHARACTER(LEN=*), INTENT(IN) :: filename
        TYPE(Inventario), DIMENSION(:), ALLOCATABLE, INTENT(I
NOUT) :: listInventario
        CHARACTER(LEN=50) :: line, comando, nombre, ubicacion
        INTEGER :: cantidad, ios, pos

        CHARACTER(LEN=50) :: dataTemp(4)

        OPEN(UNIT=20, FILE=filename, STATUS='OLD', ACTION='RE
AD', IOSTAT=ios)
        IF (ios /= 0) THEN
            PRINT *, 'Error al abrir el archivo de cambios'
            STOP
        END IF

        DO
            READ(20, '(A)', IOSTAT=ios) line
            IF (ios /= 0) EXIT
```

```
            pos = INDEX(line, ' ')
            comando = line(1:pos-1)
            line = line(pos+1:)


            CALL separarDatos(line,3,dataTemp)

            nombre = TRIM(dataTemp(1))
            READ(dataTemp(2), *) cantidad
            ubicacion = TRIM(dataTemp(3))


            IF (TRIM(comando) == 'agregar_stock') THEN
                CALL agregarStock(listInventario, nombre, can
tidad, ubicacion)
            ELSE IF (TRIM(comando) == 'eliminar_equipo') THEN
                CALL eliminarEquipo(listInventario, nombre, c
antidad, ubicacion)
            ELSE
                PRINT *, 'Comando no reconocido:', TRIM(coman
do)
            END IF
        END DO

        CLOSE(20)
    END SUBROUTINE leerCambios
```

se muestra una lectura normal como el primer archivo la diferencia es que la lectura de comandos llama a otras subrutinas

agregar stock

```
 SUBROUTINE agregarStock(listInventario, nombre, cantidad, ub
icacion)
    TYPE(Inventario), DIMENSION(:), ALLOCATABLE, INTENT(INOU
T) :: listInventario
```

```
    CHARACTER(LEN=50), INTENT(IN) :: nombre, ubicacion
    INTEGER, INTENT(IN) :: cantidad
    INTEGER :: i
    LOGICAL :: encontrado

    encontrado = .FALSE.
    DO i = 1, SIZE(listInventario)
        IF (TRIM(listInventario(i)%nombre) == TRIM(nombre) .A
ND. &
            TRIM(listInventario(i)%ubicacion) == TRIM(ubicaci
on)) THEN
            listInventario(i)%cantidad = listInventario(i)%ca
ntidad + cantidad
            encontrado = .TRUE.
            PRINT *, 'Stock actualizado para:', TRIM(nombre),
'en', TRIM(ubicacion)
            EXIT
        END IF
    END DO

    IF (.NOT. encontrado) THEN
        PRINT *, 'Equipo no encontrado para agregar:', TRIM(n
ombre), TRIM(ubicacion)
    END IF
END SUBROUTINE agregarStock
```

- el intent(out) sirve para especificar que la lista ListInventario tendra modificaciones, si se encuentra nombre y ubicacion entonces se actualiza la cantidad

eliminar stock

```
SUBROUTINE eliminarEquipo(listInventario, nombre, cantidad, u
bicacion)
    TYPE(Inventario), DIMENSION(:), ALLOCATABLE, INTENT(INOU
T) :: listInventario
```

```
    CHARACTER(LEN=50), INTENT(IN) :: nombre, ubicacion
    INTEGER, INTENT(IN) :: cantidad
    INTEGER :: i
    LOGICAL :: encontrado

    encontrado = .FALSE.
    DO i = 1, SIZE(listInventario)
        IF (TRIM(listInventario(i)%nombre) == TRIM(nombre) .A
ND. &
            TRIM(listInventario(i)%ubicacion) == TRIM(ubicaci
on)) THEN
            encontrado = .TRUE.

            ! Comprobando si la cantidad a eliminar es mayor
que la existente
            IF (cantidad > listInventario(i)%cantidad) THEN
                PRINT *, 'Error: La cantidad a eliminar (', c
antidad, ') es mayor que la existente (', &
                         listInventario(i)%cantidad, ') par
a:', TRIM(nombre), 'en', TRIM(ubicacion)
            ELSE
                ! Actualizar la cantidad o eliminar el equipo
si la cantidad es 0 o menor
                listInventario(i)%cantidad = listInventario
(i)%cantidad - cantidad
                IF (listInventario(i)%cantidad <= 0) THEN
                    listInventario(i)%cantidad = 0
                    PRINT *, 'Equipo eliminado:', TRIM(listIn
ventario(i)%nombre)
                ELSE
                    PRINT *, 'Stock actualizado para:', TRIM
(nombre), 'en', TRIM(ubicacion), &
                             'Nueva cantidad:', listInventari
o(i)%cantidad
                END IF
            END IF
```

```
            EXIT
        END IF
    END DO

    IF (.NOT. encontrado) THEN
        PRINT *, 'Error: Equipo no encontrado para elimina
r:', TRIM(nombre), TRIM(ubicacion)
    END IF
END SUBROUTINE eliminarEquipo
```

es parecida a la subrutina anterior con la diferencia que trabaja mas condiciones

imprimir inventario

```
    SUBROUTINE imprimirInventario(listInventario)
        TYPE(Inventario), DIMENSION(:), INTENT(IN) :: listInv
entario
        INTEGER :: i, ios
        REAL :: valorTotal
        CHARACTER(LEN=100) :: nombreArchivo


        nombreArchivo = 'informeInventario.txt'


        OPEN(UNIT=10, FILE=nombreArchivo, STATUS='REPLACE', A
CTION='WRITE', IOSTAT=ios)
        IF (ios /= 0) THEN
            PRINT *, 'Error al abrir el archivo:', TRIM(nombr
eArchivo)
            RETURN
        END IF
```

```
        WRITE(10, '(A)') 'Informe de Inventario:'
        WRITE(10, '(A)') '-----------------------------------
-------------------------'
        WRITE(10, '(A, A, A, A, A)') 'Equipo', '    Cantida
d', '    Precio Unitario', '    Valor Total', '    Ubicación'
        WRITE(10, '(A)') '-----------------------------------
-------------------------'


        DO i = 1, SIZE(listInventario)
            valorTotal = listInventario(i)%cantidad * listInv
entario(i)%precio
            WRITE(10, '(A20,5X, I6,5X,  F10.2, 5X, F10.2,5X,
A20)') TRIM(listInventario(i)%nombre), &
                  listInventario(i)%cantidad, listInventario
(i)%precio, valorTotal, TRIM(listInventario(i)%ubicacion)
        END DO

        CLOSE(10)

        PRINT *, 'Informe de inventario generado en:', TRIM(n
ombreArchivo)
    END SUBROUTINE imprimirInventario
```

aca se creara un arhivo llamado informeInventario.txt llamando a los datos de list inventario por medio de un Do

para esto se usara la `ACTION='WRITE' escribiendo datos`

primero se coloco las columnas de la trabla y luebo se escribieron los datos de la lista

```
 DO i = 1, SIZE(listInventario)
            valorTotal = listInventario(i)%cantidad * listInv
entario(i)%precio
            WRITE(10, '(A20,5X, I6,5X,  F10.2, 5X, F10.2,5X,
```

```
A20)') TRIM(listInventario(i)%nombre), &
                  listInventario(i)%cantidad, listInventario
(i)%precio, valorTotal, TRIM(listInventario(i)%ubicacion)
```

en esta parte de aca se llenan datos la parte de `'(A20,5X, I6,5X,  F10.2, 5X, F10.2,5X, A20)'` significa que datos imprimira el (A) hace referencia a datos tipos caractec el (I) a nuemros enteros el (F) a numeros reales y el (x) a cadena de espacios, los numeros de al lado represental la cantidad de caracters que puede contener