Manual Tecnico

[LFP] PRACTICA 1

Manual tecnico

En este manual se podran ver los metodos utilizados para la creacion del programa de manejo de archivos para la creacion de un sistema de inventario que deja agregar un nuevo inventario, actualizar los datos del inventario y crear un nuevo informe con el inventario que se desee utilizar.

- Nombre del programa: Sistema de Inventario
- Fecha de elaboración: 22 de Agosto del 2024
- Responsable: Victor Andres Estrada Eguizabal

Objetivos

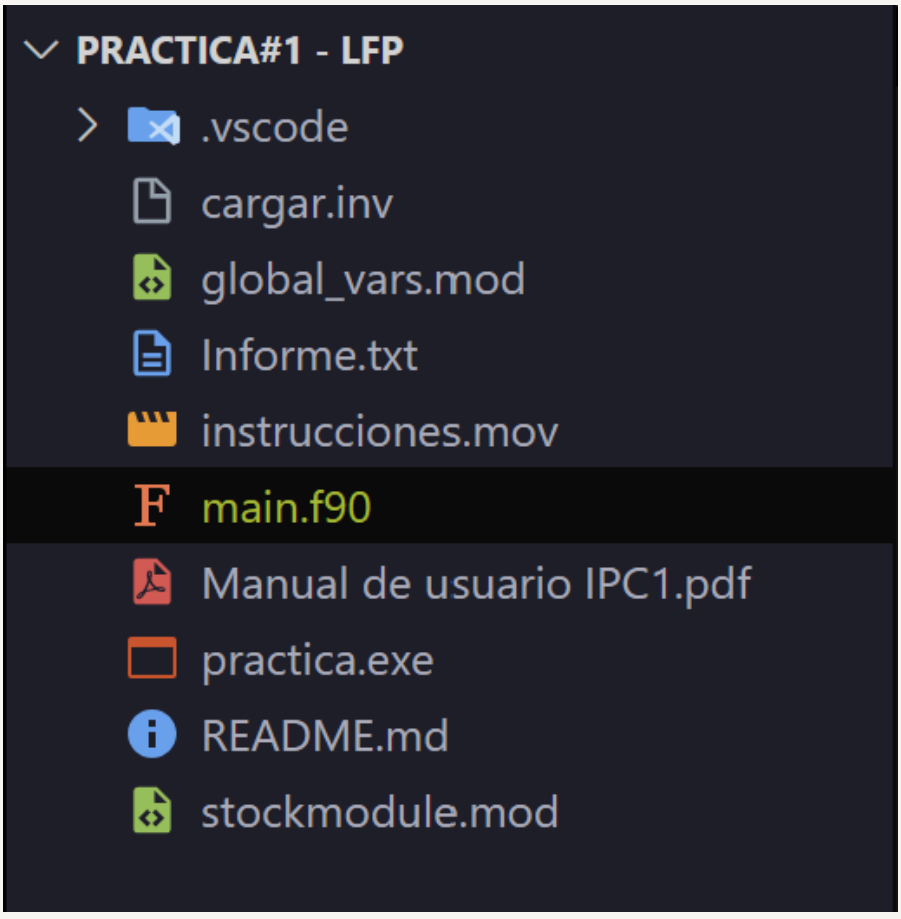
- Proporcionar una descripción clara y concisa del proyecto y sus características principales.

- Describir en detalle los modulos del proyecto asi como sus funcionalidades y como es capaz de realizar este proyecto.

- Presentar la arquitectura del proyecto, incluyendo los componentes principales y su interacción.

Logica del programa

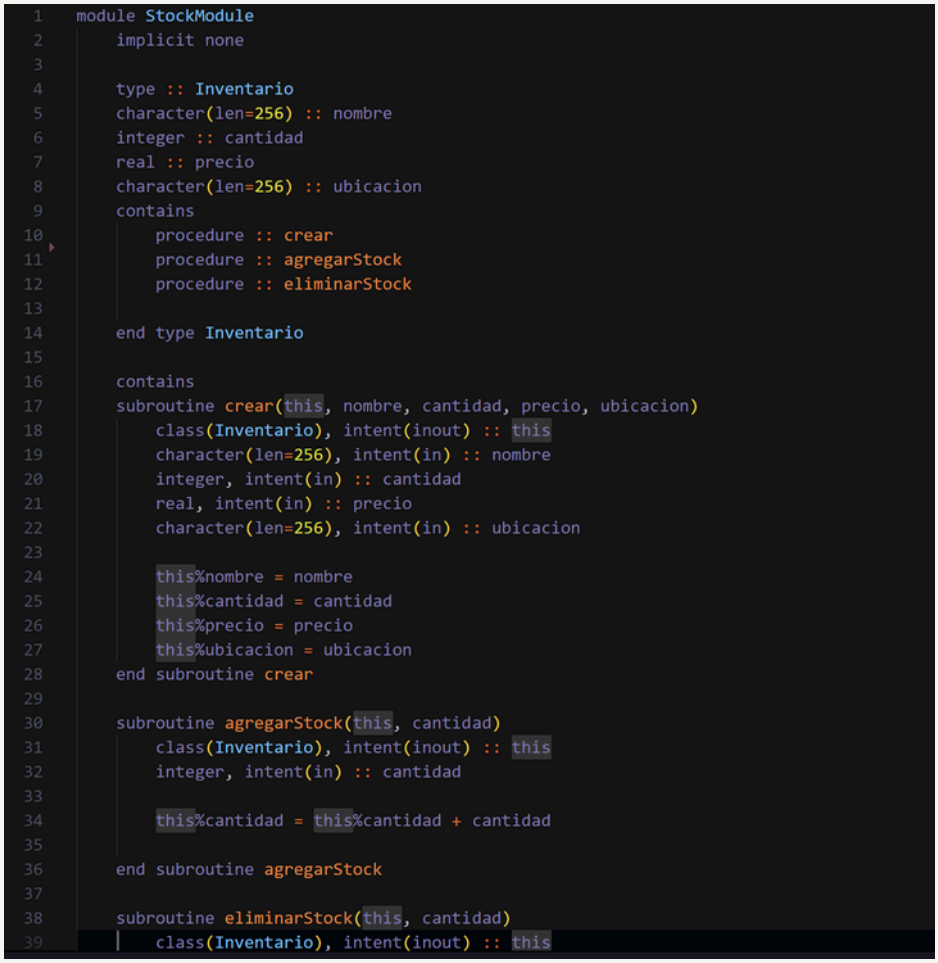
Archivos Utilizados



Se hace uso de 2 archivos para el funcionamiento del inventario, .inv para cargar el nuevo inventario y .mov para las instrucciones de agregar y eliminar articulos, y se crea un nuevo archivo txt para el informe.

Logica del programa

StockModule



```
module StockModule
    implicit none

    type :: Inventario
    character(len=256) :: nombre
    integer :: cantidad
    real :: precio
    character(len=256) :: ubicacion
    contains
        procedure :: crear
        procedure :: agregarStock
        procedure :: eliminarStock

    end type Inventario

    contains
    subroutine crear(this, nombre, cantidad, precio, ubicacion)
        class(Inventario), intent(inout) :: this
        character(len=256), intent(in) :: nombre
        integer, intent(in) :: cantidad
        real, intent(in) :: precio
        character(len=256), intent(in) :: ubicacion

        this%nombre = nombre
        this%cantidad = cantidad
        this%precio = precio
        this%ubicacion = ubicacion
    end subroutine crear

    subroutine agregarStock(this, cantidad)
        class(Inventario), intent(inout) :: this
        integer, intent(in) :: cantidad

        this%cantidad = this%cantidad + cantidad

    end subroutine agregarStock

    subroutine eliminarStock(this, cantidad)
        class(Inventario), intent(inout) :: this
```

- Define un tipo de datos Inventario, que representa un artículo en el inventario con atributos como nombre, cantidad, precio y ubicación.
- El módulo contiene métodos para crear un artículo (crear), agregar stock (agregarStock), y eliminar stock (eliminarStock).

Logica del programa

global_vars

```
57  module global_vars
58      use StockModule
59      integer :: num = 1
60      integer :: i
61      type(Inventario), dimension(100) :: articulos
62
63  end module global_vars
```

- Define variables globales, incluyendo un arreglo para almacenar los artículos (articulos) y un contador (num) para seguir la cantidad de artículos en el inventario.
- También define una lista de ubicaciones válidas (ubicaciones_validas) que se utilizan para validar la ubicación de los artículos.

Logica del programa

program main

```
65  program main
66      use StockModule
67      use global_vars
68      implicit none
69
70      integer :: opcion
71
72      do
73
74          print *, "----------------------------------------------------------------"
75          print *, "Practica 1 - Lenguajes Formales y de Programacion - Seccion: A-"
76          print *, "----------------------------------------------------------------"
77          print *, "# Sistema de Inventario"
78          print *, " "
79          print *, "1. Cargar Inventario Inicial"
80          print *, "2. Cargar instrucciones de movimiento"
81          print *, "3. Crear Informe de Inventario"
82          print *, "4. Salir"
83          print *, " "
84          print *, "Ingrese una opcion: "
85          read *, opcion
86          select case (opcion)
87              case(1)
88                  call cargarInventario()
89
90              case(2)
91                  call cargarInstrucciones()
92
93              case(3)
94                  call crearInforme()
95
96              case(4)
97                  print *, "Saliendo del sistema..."
98                  print *, "Presione Enter para salir del programa"
99                  read *
100                 call system ("cls")
101                 stop
102
```

- Presenta un menú interactivo al usuario con opciones para cargar inventario inicial, procesar instrucciones de movimiento (agregar o eliminar stock), crear un informe del inventario, o salir del programa.
- Dependiendo de la opción seleccionada, se llama a las subrutinas correspondientes para realizar las acciones solicitadas.

Logica del programa

cargarInventario

```
116  subroutine cargarInventario()
117      integer :: iunit, ios, pos, cantidad_int
118      real :: precio_real
119      character(len=256) :: nombre, cantidad, precio, ubicacion, linea, comando
120      character(len=512) :: ruta_archivo
121
122      print *, "Ingrese la ruta del archivo"
123      read(*, '(A)') ruta_archivo
124
125      iunit = 10
126
127      open(unit = iunit, file = trim(ruta_archivo), status = 'old', action = 'read', iostat = ios)
128
129      if ( ios /= 0 ) then
130          print *, "Error al abrir el archivo"
131          stop
132      end if
133
134      do
135          read(iunit, '(A)', iostat=ios)linea
136          if ( ios /= 0 ) exit
137              linea = trim(linea)
138
139              print *, linea
140
141
142
143              pos = index(linea, ' ')
144          if(pos > 0)then
145              comando = linea(1: pos-1)
146              linea = trim(linea(pos+1:))
147              pos = index(linea, ';')
148              if ( pos > 0 ) then
149                  nombre = linea(1:pos-1)
150                  linea = trim(linea(pos+1:))
151                  pos = index(linea, ';')
152                  if ( pos > 0 ) then
153                      cantidad = linea(1:pos-1)
154                      read(cantidad, '(I10)', iostat=ios) cantidad_int
```

- Lee un archivo con información sobre los artículos y los agrega al inventario.
- Cada línea del archivo contiene un comando y los detalles del artículo (nombre, cantidad, precio, ubicación).

Logica del programa

cargarInstrucciones

```
191  subroutine cargarInstrucciones()
192      use StockModule
193      use global_vars
194      integer :: iunit, ios, pos, cantidad_int
195      character(len=256) :: nombre, cantidad, ubicacion, linea, comando
196      character(len=512) :: ruta_archivo
197
198      print *, "Ingrese la ruta del archivo"
199      read(*, '(A)') ruta_archivo
200
201      iunit = 11
202
203      open(unit = iunit, file = trim(ruta_archivo), status = 'old', action = 'read', iostat = ios)
204
205      if ( ios /= 0 ) then
206          print *, "Error al abrir el archivo"
207          stop
208      end if
209
210      do
211          read(iunit, '(A)', iostat=ios)linea
212          if ( ios /= 0 ) exit
213              linea = trim(linea)
214
215              print *, linea
216
217              pos = index(linea, ' ')
218          if(pos > 0)then
219              comando = linea(1: pos-1)
220              linea = trim(linea(pos+1:))
221              pos = index(linea, ';')
222              if ( pos > 0 ) then
223                  nombre = linea(1:pos-1)
224                  linea = trim(linea(pos+1:))
225                  pos = index(linea, ';')
226                  if ( pos > 0 ) then
227                      cantidad = linea(1:pos-1)
228                      read(cantidad, '(I10)', iostat=ios) cantidad_int
```

- Lee un archivo con instrucciones para agregar o eliminar stock de los artículos en las bodegas.
- Verifica que las ubicaciones sean válidas antes de realizar cualquier operación y maneja los errores de ubicación incorrecta.

Logica del programa

crearInforme

```
278  subroutine crearInforme()
279      use StockModule
280      use global_vars
281
282      integer iunit, ios
283
284      iunit = 20
285
286      open(unit = iunit, file = "Informe.txt", status = 'replace', action = 'write', iostat = ios)
287
288      if ( ios /= 0 ) then
289          print *, "Error al abrir el archivo"
290          stop
291      end if
292
293      write(iunit, '(A30, A30, A30, A30)')"Nombre", "Cantidad", "Precio", "Ubicacion"
294
295      do i=1, num-1
296          write(iunit, '(A30, I30, F30.2, A30)')trim(articulos(i)%nombre), articulos(i)%cantidad, articulos(i)%precio, tr
297      end do
298
299      print *, "Informe creado con exito"
300      print *, "Presione Enter o cualquier tecla para continuar"
301
302      close(unit = iunit)
303
304
305  end subroutine crearInforme
```

Genera un informe de texto que contiene todos los artículos en el inventario con sus respectivas cantidades, precios y ubicaciones.