UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
LENGUAJES FORMALES Y DE PROGRAMACIÓN
CATEDRÁTICO: ING. DAVID MORALES
TUTOR ACADÉMICO: HERBERTH ABISAI AVILA

MANUAL TÉCNICO DE PRÁCTICA 1

ENNER ESAÍ MENDIZABAL CASTRO
CARNÉ: 202302220
SECCIÓN: B+

GUATEMALA, 13 DE AGOSTO DEL 2,024

ÍNDICE

INTRODUCCIÓN

Este manual se creó con el fin de proporcionar la información técnica, necesario para la comprensión del funcionamiento lógico del programa, esto con un lenguaje más técnico que permita una comprensión más profunda de las funciones, variables y todo lo utilizado para la creación del programa.

OBJETIVOS

1. GENERAL

1.1. Describir el funcionamiento del programa desde una perspectiva más allegada a su lógica y a sus algoritmos.

2. ESPECÍFICOS

2.1. Especificar las funciones y procedimientos que se llevaron a cabo para realizar cada una de las funcionalidades del programa

2.2. Explicar la manera en la que se resolvieron los problemas que surgieron durante la creación del programa.

ALCANCES DEL SISTEMA

Este manual se creó con el fin de propiciar una comprensión profunda sobre la manera en la cual se desarrolló la creación del programa y la solución de los problemas durante su elaboración. Se pretende proporcionar un conocimiento técnico tal que permita enseñar su funcionamiento con las suficientes explicaciones para que, si se desea, pueda ser replicado a posterior.

ESPECIFICACIÓN TÉCNICA

- **REQUISITOS DE HARDWARE**
  - Procesador con arquitectura x86
  - Teclado (opcional)
- **REQUISITOS DE SOFTWARE**
  - Editor de texto compatible con Fortran
  - Compilador de Fortran

LÓGICA Y DESCRIPCIÓN DE LA SOLUCIÓN

- **Creación del menú principal:** Al iniciarse el programa se muestra un menú que permite acceder a cada una de las funciones del programa Para la creación de este, se decidió utilizar un ciclo *while*, del cual no dejaría de estar en ciclo hasta que la variable *booleana* pasara a ser verdadera, lo cual únicamente sucedería si se presionaba el número 4, el cual era para salir. Para acceder a cada unas de las distintas funcionalidades, a excepción de la 4, se crearon distintas subrutinas que se ejecutarían de acuerdo con la opción seleccionada en el switch en ciclado dentro.

```
9   program sistema_inventario
10      use globales
11      implicit none
12      logical :: Salir !Declarando las Variables
13      integer :: opcion
14      salir=.false.
15      print *, "Bienvenido al sistema de inventario :)" !Bienvenida
16      do while (.not. Salir) !Iniciando el ciclo
17          print *, "SELECCIONE UNA OPCION:"
18          print *, "#1. Cargar inventario inicial"
19          print *, "#2. Cargar Instrucciones de movimiento"
20          print *, "#3. Crear informe de inventario"
21          print *, "#4. Salir"
22          read *, opcion
23          select case(opcion)
24              case(1)
25                  call cargar_inventario()
26              case(2)
27                  call cargar_instrucciones()
28              case(3)
29                  call crear_informe()
30              case(4)
31                  salir = .true.
32                  print *, "Saliendo del programa..."
33              case default
34                  print *, "Opcion no valida"
35          end select
36      end do
37  end program sistema_inventario
```

- **Almacenamiento de todos los datos del programa para luego imprimirlos:** Para conseguir almacenar correctamente los datos que se van cargando y modificando durante la ejecución del programa, se decidió

utilizar como solución un arreglo estático declarado de forma global, puesto a que no existe los arreglos dinámicos dentro de fortran, en el cual se podrían almacenar todos los datos deseados hasta el límite de cada uno de estos arreglos bidimensionales. Fue creado uno para las variables de tipo *string* y otro para las variables de tipo *real*.

```
module globales
    implicit none
    character(len=15), dimension (3, 50) :: datos !Declarando el arreglo que contiene TODO el archivo
    real, dimension(3,50) :: datosN
    integer :: productos !Declarando la variable que contiene la cantidad de productos que
contains
end module globales
```

- **Análisis léxico para la separación de los enunciados:** Para realizar esto, primero se recorrió cada una de las líneas del archivo.

```
53          !Abriendo el archivo
54          open(unit=1, file="inventario.inv", status="old", action="read")
55          do
56              read(1, '(A)', iostat=iostat) linea! Leyendo cada línea
57              if(iostat/=0) then
58                  exit
59              end if
```

Posteriormente, dentro del ciclo mismo que recorre cada una de las líneas del archivo, se separa cada línea en dos a partir de espacio (“ ”) que hay entre la instrucción y los parámetros.

```
60              contador=contador+1
61              p=index(linea, ' ')!Separando la instrucción de los parámetros
62              if(p>0) then
63                  read(linea(1:p-1), '(A)', iostat=iostat) instruccion
64                  parametros=linea(p+1:)
65              end if
```

Una vez teniendo estos separados, se verifica si la instrucción es valida y, en dado caso esta esté bien declarada, se comienza a separar cada uno de los parámetros y se guardan en variables, así como se hizo al inicio separando las instrucciones de los parámetros.

```
66          if(instruccion=="crear_equipo") then
67              exitos=exitos+1
68              p=index(parametros, ';')!Separando el nombre de los demás parámetros
69              if(p>0) then
70                  read(parametros(1:p-1), '(A)', iostat=iostat) nombre
71                  parametros=parametros(p+1:)
72              end if
73              p=index(parametros, ';')!Separando la cantidad de los demás parámetros
74              if(p>0) then
75                  read(parametros(1:p-1), '(I6)', iostat=iostat) cantidad
76                  parametros=parametros(p+1:)
77              end if
78              p=index(parametros, ';')!Separando el precio de la ubicación
79              if(p>0) then
80                  read(parametros(1:p-1), '(F6.2)', iostat=iostat) precio !El precio debe estar en formato 6.2 [dos decimales]
81                  ubicacion=parametros(p+1:)
82              end if
```

Ya con todo separado dentro de cada unas de sus respectivas variables declaradas previamente, se coloca todo dentro de los dos arreglos bidimensional creados en global para poder guardar los todos estos datos y que puedan ser recorridos y obtenidos nuevamente; y sin en dado caso, la instrucción está mal desde el inicio, indicará que esta está mal y no se reproducirá nada. Este ciclo se copio y pegó en la segunda subrutina (la que se ejecuta al presionar el 2) adaptándola a la cantidad de variables.

```
83              !print *,"nombre:",nombre,"cantidad:",cantidad,"precio:",precio,"ubicacion:",ubicacion
84              datos(1,exitos)=instruccion !Guardando los datos en el arreglo
85              datos(2,exitos)=nombre
86              datosN(2,exitos)=cantidad
87              datosN(3,exitos)=precio
88              datos(3,exitos)=ubicacion
89              datosN(1,exitos)=contador
90              print *, "INSTRUCCION: " ,trim(datos(1,exitos)),"; NOMBRE: ",trim(datos(2,exitos)),"; CA
91          else
92              print *, "INSTRUCCION: ", instruccion ," de la linea", contador, " no valida"
93          end if
94      end do
```

- **Cargar Instrucciones:** Esto se realiza mediante una subrutina la cual utiliza el procedimiento de la separación de los enunciados para obtener los parámetros del archivo *inventario.inv* el cual se abre previamente y se le asigna un *unit* con el cual se identificará este archivo. Luego se guarda todo en los arreglos globales y se va mostrando información en la consola

conforme se va realizando cada análisis sintáctico.

```
use globales
implicit none
!Declarando las variables para la carag del inventario
character(len=50) :: linea,parametros
character(len=15) :: nombre, ubicacion, instruccion
integer:: cantidad, iostat, contador,p,exitos
real :: precio
logical :: e
contador=0
exitos=0
!Verifico la existencia del archivo
inquire(file="inventario.inv", exist=e)
if(e) then
```

- **Cargar instrucciones:** Para esta, como se menciona anteriormente, se toma la lógica para la separación de los enunciados y se separa todo, así como se mostro nuevamente, adaptándolo a la cantidad de variables para esta subrutina y colocando un condicional adicional para verificar si la instrucción es para eliminar o para adicionar productos.

```
128 >   if(instruccion=="agregar_stock") then !PARA AGREGAR STOCK-----------------------
158     else
159         if(instruccion=="eliminar_equipo") then !PARA ELIMINAR EQUIPO---------------
160             p=index(parametros, ';')!Separando el nombre de los demás parámetros
161         if(p>0) then
```

Una vez realizado todo esto, en vez de colocar todo dentro de los arreglos globales, se modifican primeramente buscándolos dentro de estos arreglos y luego colocándolos o eliminándolos en la posición encontrada para así poder luego confirmar que se pudo realizar la operación deseada con el producto deseado.

```
171                 do while (i<50) !Ciclo para recorrer el arreglo y eliminar una cantidad
172                     i=i+1
173                     if (nombre==datos(2,i)) then
174                         if (ubicacion==datos(3,i) ) then
175                             if (cantidad>datosN(2,i)) then
176                                 print *,"ERROR, la cantidad a eliminar del producto", trim(nombre), " es mayor a la cantidad en stock"
177                                 v1=.false.
178                             else
179                             datosN(2,i)=datosN(2,i)-cantidad
180                             exitos=exitos+1
181                             v1=.true.
182                             pos=i
183                             end if
184                         else
185                             v1=.false.
186                             print *,"ERROR, el producto ", trim(nombre) ," no se encuentra en ", trim(ubicacion)
187                         end if
188                     else
189                         end if
190                 end do
191                 i=0 !Reinicio de la cuenta del ciclo
192                 if(v1) then
193                     print *, "Producto: ", trim(datos(2,pos)), " actualizo su cantidad a: ", datosN(2,pos)
194                 end if
```

Adicionalmente, si se encuentra el producto, mas no se encuentra en la ubicación especificada, indicará que no está donde se indica y no se realizará ningun procedimiento.

```
184                         else
185                             v1=.false.
186                             print *,"ERROR, el producto ", trim(nombre) ," no se encuentra en ", trim(ubicacion)
187                         end if
188                     else
189                         end if
```

- **Creación de informe:** Para realizar el informe, primero se crea un archivo con el nombre de *informe.txt* (en dado caso ya exista, se sobrescribirá este mismo ya existente), luego se abrirá para poder escribir en él mediante un ciclo el cual repasa cada uno de los valores almacenados dentro de los arreglos globales y los escribe en el con el formato deseado. Al finalizar indica en la consola que se generó exitosamente el informe.

```
208 subroutine crear_informe() !función para crear el informe
209     use globales
210     implicit none
211     character(len=15) :: informe
212     integer :: i
213     i=0
214     informe="informe.txt"
215     open(unit=22, file=informe,status="unknown", action="write") !Abriendo el archivo y creandolo
216     write(22,*) "------------------------| INFORME DE INVENTARIO |------------------------" !Escribiendo en el archivo
217     write(22,*) "|    EQUIPO    |   CANTIDAD   |  PRECIO UNITARIO  | VALOR TOTAL | UBICACION |"
218     do while(i<productos) !Ciclo para recorrer el arreglo y escribir en el archivo
219         i=i+1
220         write(22,*) datos(2,i), datosN(2,i), datosN(3,i), datosN(2,i)*datosN(3,i), trim(datos(3,i))
221     end do
222     write(22,*) "-------------------------------------------------------------------------------" !Mensaje de confirmación
223     close(22); !Cerrando el archivo
224     print *, "Informe generado exitosamente :)" !Mensaje de confirmación
225 end subroutine crear_informe !Terminando la funcion para leer el archivo
226
```

- **Salir del programa:** Para esto simplemente se cambia el valor de la variable de tipo booleano que mantiene el programa en ciclado, para que este deje estarlo y se ejecute hasta acabarse. Además, si no se escogió ninguna de las opciones validas, se indicará que no se ingreso la una opción válida.

```
30              case(4)
31                  salir = .true.
32                  print *, "Saliendo del programa..."
33              case default
34                  print *, "Opcion no valida"
35          end select
36      end do
```