Proyecto Final de Algorítmica

Integrantes: Palacio Cristian, Quiroga Agustín

Algoritmo registrosMeteorologicos
Léxico
//REGISTRO
Tregdiario = <
ddmmyyyy E Z, **//fecha del registro, dd es el día, mm el mes, yyyy el año**
tmax E Z, **//temperatura máxima ocurrida ese día**
tmin E Z, **//temperatura mínima ocurrida ese día**
HUM E (0..100), **//humedad promedio del día registrado**
PNM E Z, **//valor es la presión atmosférica**
DV E (0..360), **//dirección del viento que se toma en grados y en valores enteros de 0 a 360**
FF E Z, **//velocidad máxima ocurrida en el dia**
PP E Z, **//precipitación pluvial acumulado en 24 hs**
borrado E Lógico **// baja lógica del registro**
>

TNodo = < info E arreglo [1..max] de Tregdiario
next E puntero a TNodo >

//VARIABLES
max = 1000
f E archivo de Tregdiario **//variable interna del archivo**
Tarreglo **=** arreglo [1..max] de Tregdiario
fechaNew E Tarreglo **//Registro**
nomArch E cadena **//nombre del archivo**
buffer2 E cadena **//cadena para agregarle el .dat al nombre del archivo**
respMenu E Z **//Respuesta del menu**
cantFechas E Z **//Cantidad total de fechas que hay**
searchDate E Z **//Fecha a buscar para el uso de funciones**

FUNCIONES

//Funcion que revisa si el archivo esta lleno.
Función Llena(dato fecha E Tarreglo) --> **Logico**
Léxico local
m E Z
Inicio

```
m <-- 0
mientras (fecha[m].ddmmyyyy <> 0) hacer
                m <-- m + 1
fmientras

si (m = max) entonces
                <-- verdadero
  sino
                <-- falso
fsi
```

```
Ffuncion

  //Funcion que revisa si el archivo esta vacio.
Funcion Vacia(dato fecha E Tarreglo) --> Logico
Lexico Local
m E Z
contador E Z
Inicio
        m <-- 1,
        contador <--    0

        mientras (fecha[m].ddmmyyyy <> 0) hacer
                        contador <-- contador + 1
                        m <-- m + 1
        fmientras

        si (contador = 0) entonces
                        <-- verdadero
        sino
                        <-- falso
         fsi
Ffuncion

// Funcion recursiva que busca una fecha en el registro
Funcion buscarFecha(dato fecha E Tarreglo, dato fechaBuscar E Z, dato n E Z) --> Z
Inicio
        si (fecha[n].ddmmyyyy <> fechaBuscar y n = cantFechas) entonces
                        <--    -1
        sino
                        si (fecha[n].ddmmyyyy = fechaBuscar) entonces
                                <-- n
                        sino
                                <-- buscarFecha(fecha, fechaBuscar, n + 1)
                        fsi
         fsi
Ffuncion


Funcion CrearNodo(dato fecha E Tarreglo, dato index E Z) --> puntero a TNodo
Lexico Local
nuevo E puntero a TNodo
Inicio
        Obtener(nuevo)
        Si(nuevo <> nil) entonces
                        (^nuevo).info[0] <-- fecha[index]
                        (^nuevo).next <-- nil
                          <-- nuevo
        Fsi
```

Ffuncion

ACCIONES

Accion Menu (resultado rp E Z)
Lexico Local
msg E cadena
Inicio
    mientras (rp < 1 ó rp > 10) hacer
        msg <-- "1- Alta de un registro diario
        2- Suprimir un registro diario
        3- Modificar un registro, se busca por la fecha
        4- Mostrar todos los registros activos
        5- Buscar registro de un día dado y mostrar todos los parámetros
        6- Listar el día o días de máxima temperatura en lo que va del año
        7- Listar el día o días de máxima precipitación en lo que va del año
        8- Listar las fechas de mayor a menor velocidad de viento
        9- Realizar una copia de seguridad del archivo del año en curso
        10-Salir "
        salida: msg
        entrada: rp
    mientras
FAccion

Accion ingresoDatos(dato-resultado fe E Tarreglo, dato index E Z)
Léxico local
dd e (1..31)
mm e (1..12)
anio e Z
ddText, mmText, anioText e Cadena
enteroNuevo e Z
Inicio

    entrada: anio, fe[index].tmax, fe[index].tmin, fe[index].FF

    Mientras (anio < 1000 ó anio> 9999) hacer
        entrada: anio
    Fmientras

    si(anio MOD 4 = 0 y anio MOD 100 <> 0 ó anio MOD 400 = 0)entonces
        Entrada: mm
        Segun
            mm = 1 ó 3 ó 5 ó 7 ó 8 ó 10 ó 12:
                Entrada: dd
            mm = 2:
                Mientras (dd > 29) hacer
                    Entrada: dd
                Fmientras

```
                mm = 4 ó 6 ó 9 ó 11:
                        mientras (dd > 30 ) hacer
                                Entrada: dd
                        Fmientras
        Fsegun
sino
        Entrada: mm
        Segun
                mm = 1 ó 3 ó 5 ó 7 ó 8 ó 10 ó 12:
                        Entrada: dd
                mm = 2:
                        Mientras ( dd > 28 ) hacer
                                Entrada: dd
                        Fmientras
                mm = 4 ó 6 ó 9 ó 11:
                        mientras (dd > 30 ) hacer
                                Entrada: dd
                        Fmientras
        Fsegun
fsi
```

//Usariamos la libreria "sprintf" para transformar los enteros de la fecha, en cadenas separadas, luego las uniriamos y convertiriamos la union en un entero con la libreria "strtol" quedando en formato ddmmyyyy

```
ddText <-- sprintf(dd)
mmText <-- sprintf(mm)
anioText <-- sprintf(anio)

ddText <-- ddText + mmText + anioText
```

//Nos quedaria la fecha en una cadena y la pasamos a entero ahora

```
enteroNuevo <-- strtol(ddText)
fe[index].ddmmyyy <-- enteroNuevo


Mientras ( fe[index].Hum < 0 ó fe[index].Hum > 100 ) hacer
        entrada: fe.HUM,
Fmientras

 Mientras ( fe[index].PNM < 900 ó fe[index].PNM > 3500 ) hacer
        entrada:    fe[index].PNM
Fmientras

Mientras ( fe[index].PP < 0 ó fe[index].PP > 1000 ) hacer
        entrada: fe[index].PP
Fmientras

Mientras ( fe[index].DV < 0 ó fe[index].DV > 360 ) hacer
```

```
                        entrada: fe[index].DV
        Fmientras
FAccion

//Accion que revisa si el archivo tiene previamente elementos  cargados y los agrega al arreglo.
Accion revisoPreCargado(dato-resultado fecha E Tarreglo, dato nameArchivo E cadena)
Lexico Local
arch E archivo de Tregdiario
i E Z
Inicio
        i <-- 0
        Abrir(nameArchivo, arch, l)
                        Mientras( not(EOF(arch)) hacer //Si tiene elementos cargados los cuento.
                                i <-- i+1
                                Leer(fecha[i], arch)
                        fmientras
        Cerrar(arch)
        cantFechas <-- i
Faccion

Accion backupFile(dato fecha E Tarreglo, dato nameFile E cadena)
Lexico Local
arch E archivo de Tregdiario
i E Z
Inicio
        nameFile <-- nameFile + "-copia.dat"
                Abrir(nameFile, arch, e)
                        para (i <-- 1, i < cantFechas, i <-- i+1) hacer
                                si(fecha[i].borrado <> verdadero)entonces
                                Escribir(arch, fecha[i])
                        fsi
                        fpara
                Cerrar(arch)
Faccion

Accion Insertar(dato fecha E Tarreglo)
Inicio
        ingresoDatos(fecha, cantFechas)
        cantFechas <-- cantFechas + 1
Faccion

//Muestra por salida todos los campos de la fecha actual
Accion muestroText(dato fecha E Tarreglo, dato numActual E Z)
Inicio
salida: fecha[numActual].ddmmyyyy          //Fecha actual
                fecha[numActual].tmax      //Temperatura Máxima
                fecha[numActual].tmin      //Temperatura Mínima
                fecha[numActual].HUM //Hum
```

```
                fecha[numActual].PNM //PNM
                fecha[numActual].DV          //DV
                fecha[numActual].FF          //FF
                fecha[numActual].borrado     //Borrado
Faccion

//Accion que muestra los elementos del archivo.
Accion Mostrar(dato fecha E Tarreglo)
Lexico Local
i E Z
Inicio
        Para (i <-- 1, i < cantFechas, i <-- i+1) hacer
            si(fecha[i].borrado <> verdadero)entonces
                    muestroText(fecha, i) //Llamo y muestro los datos de ese registro
                fsi
        Fpara
Faccion

Accion guardoLista(dato fecha E Tarreglo, dato nameArchivo E cadena)
Lexico Local
arch E archivo de Tregdiario
i E Z
Inicio
        Abrir(nameArchivo, arch, a)
            Para    (i <-- 1, i < cantFechas, i <-- i+1) hacer
                        Escribir(fecha[i], arch)
            Fpara
        Cerrar(arch)
Faccion

//Accion que permite modificar una fecha dada.
Accion modificoFecha(dato-resultado fecha E Tarreglo, dato fechaBuscar E Z)
Lexico Local
n E Z
msg E cadena
Inicio
      n <-- 1
       Mientras (fecha[n].ddmmyyyy <> fechaBuscar y n <= cantFechas) hacer
            n <-- n + 1
      Fmientras

      Si (fecha[n].ddmmyyyy <> fechaBuscar ó fecha[n].borrado = verdadero) entonces
                    msg <-- "¡Esa fecha no existe!"
                    Salida: msg
      Sino
                    ingresoDatos(fecha, n)
       Fsi
Faccion
```

```
//Accion que borra una fecha dada.
Acción borroFecha(dato-resultado fecha E Tarreglo, dato fechaBuscar E Z)
Léxico Local
n E Z
msg E cadena
Inicio
        n <-- 1
        Mientras (fecha[n].ddmmyyyy <> fechaBuscar y n <= cantFechas) hacer
                        n <-- n+1          //buscamos para ver si la fecha existe dentro del arreglo
        Fmientras

        Si (fecha[n].ddmmyyyy <> fechaBuscar) entonces
                        msg <-- "¡Esa fecha no existe!"   //si no existe mostramos un mensaje
                        Salida: msg
        Sino
                //de existir la fecha, se borra, asignándole 1 al campo lógico(borrado) del registro de
esa fecha
                         fecha[n].borrado <-- verdadero
        Fsi
Faccion

Accion InsertarC(resultado primero E puntero a TNodo, dato fecha E Tarreglo, dato index E Z)
Lexico Local
nuevo E puntero a TNodo
Inicio
        nuevo    <-- CrearNodo(fecha, index)
        Si (primero = nil) entonces
                        primero <-- nuevo
        Sino
                        (^nuevo).next <-- primero
                        primero <-- nuevo
        Fsi
Faccion

Accion maxTemperature(dato fecha E Tarreglo)
Lexico Local
n, index, mayor, cantTempMax E Z
primero E puntero a TNodo
Inicio
                n <-- 1
                mayor <-- fecha[0].tmax
                index <-- 0
                cantTempMax <-- 0

                Mientras (n <= cantFechas) hacer
                        Si    (fecha[n].tmax >= mayor) entonces      //temperatura maxima
                                        mayor <-- fecha[n].tmax
```

```
                                        index <--    n                    //guardamos la posicion en
index
                                Fsi
                        n <-- n+1
                 Fmientras

                n <-- 1  //inicializamos de nuevo n en 1

                Mientras (n <= cantFechas) hacer
                //vemos si hay alguna otra temperatura igual a la maxima
                        Si (fecha[n].tmax = mayor) entonces
                        //si hay alguna la insertamos a la cabeza en la lista
                                InsertarC(primero, fecha, n)
                        Fsi
                n <-- n+1
                Fmientras

                mientras ((^primero).next <> nil) hacer
                        Salida: (^primero).info[0].ddmmyyyy    //finalmente mostramos la fecha
                        Salida: (^primero).info[0].tmax) // y tambien la temperatura
                        primero <-- (^primero).next
                Fmientras
Faccion

Accion speedWind(dato fecha E Tarreglo)
Lexico Local
i E Z
j E Z
temp E Tarreglo
aux E Tarreglo
topTen e Z
limit e Z
Inicio
        topTen <-- 10
        limit <-- 0
        Para (i <-- 1, i < cantFechas, i <-- i+1) hacer
                        aux[i] <-- fecha[i]
        Fpara

        Para (i <-- 1, i < cantFechas, i <-- i+1) hacer
                        Para (j <-- 1, j < cantFechas - 1, j <-- j+1) hacer
                                Si (aux[j].FF < aux[j + 1].FF) entonces
                                        temp[0] <-- aux[j];
                                        aux[j] <-- aux[j + 1];
                                        aux[j + 1] <-- temp[0];
                                Fsi
                        Fpara
        Fpara
```

```
        si(cantFechas < 10)entonces
                limit <-- cantFechas
        fsi

        Para (i <-- 1, i < cantFechas, i <-- i+1) hacer
                        Si (aux[i].borrado <> verdadero y topTen < 10) entonces
                                        Salida: aux[i].ddmmyyyy aux[i].FF
                        topTen <-- topTen + 1
                        Fsi
        Fpara
Faccion

Accion orderPrecipitation(dato fecha E Tarreglo)
Lexico Local
i E Z
a E Z
topTen e Z
limit e Z
aux E Tarreglo
index Tarreglo
Inicio
        topTen <-- 10
        limit <-- 0
        Para (i <-- 1, i < cantFechas, i <-- i+1) hacer
                        aux[i] <-- fecha[i]
          Fpara

        Para    (i <-- 1, i < cantFechas, i <-- i+1) hacer
                        index[i] <-- aux[i]
                        a <-- i - 1
                        Mientras (a >= 0 ^ aux[a].PP > index[i].PP)     hacer
                                aux[a + 1] <-- aux[a]
                                a <-- a-1
                        Fmientras
                        aux[a + 1] <-- index[i]
          Fpara

        si(cantFechas < 10)entonces
                limit <-- cantFechas
        fsi

        Para (i <-- cantFechas - 1, i >= 0, i <-- i - 1) hacer
                        Si (aux[i].borrado <> verdadero y topTen < 10) entonces
                                Salida: aux[i].ddmmyyyy, aux.[i].PP
                        topTen <-- topTen + 1
                        Fsi
        Fpara
```

Faccion

//||||||||||||||||||INICIO DEL PROGRAMA||||||||||||||||||
Inicio

```
        cantFechas <-- 0

        //el ususario le da nombre externo al archivo, de no existir se crea, sino se abre
        Entrada: nomArch

        buffer2 <-- ".dat"

        //le concatenamos la cadena ".dat" al nombre          ingresado por el usuario
        nomArch <-- nomArch + buffer2

        //reviso si el archivo tiene registros cargados
        revisoPreCargado(fecha, nomArch)

    Mientras(respMenu <> 10) hacer    //Selecciono una opcion.
            Menu(respMenu)
            según
                    (respMenu = 1):
                            Si (Llena(fechaNew) = verdadero) entonces
                                    msg <-- "¡La lista esta llena, no puedes hacer esto!"
                                    Salida: msg
                            Sino
                                    Insertar(fechaNew)
                            Fsi
                    (respMenu = 2):
                            Si (Vacia(fechaNew) = verdadero) entonces
                                    msg <-- "¡La lista esta vacia, no puedes hacer esto!"
                                    salida: msg
                            Sino
                                    entrada: searchDate
                                    borroFecha(fechaNew, searchDate)
                            Fsi

                    (respMenu = 3):
                            Si (Vacia(fechaNew) = verdadero) entonces
                                    msg <-- "¡La lista esta vacia, no puedes hacer esto!"
                                    salida: msg
                            Sino
                                    entrada: searchDate
                                    modificoFecha(fechaNew, searchDate)
                            Fsi
                    (respMenu = 4):
                            Si (Vacia(fechaNew) = verdadero) entonces
                                    msg <-- "¡La lista esta vacia, no puedes hacer esto!"
                                    salida: msg
```

```
                    Sino
                        Mostrar(fechaNew)
                    Fsi
                (respMenu = 5):
                    si (Vacia(fechaNew) = verdadero) entonces
                        msg <-- "¡La lista esta vacía, no puedes hacer esto!"
                        Salida: msg
                    sino
                        Entrada: searchDate
                        Si (buscarFecha(fechaNew, searchDate, 1) = -1)
entonces
                            msg <-- "¡Esa fecha no existe!"
                            salida: msg
                          Sino
                            muestroText(fechaNew,
buscarFecha(fechaNew, searchDate, 1))
                        fsi
                    fsi
                (respMenu = 6):
                        Si (Vacia(fechaNew) = verdadero) entonces
                        msg <-- "¡La lista esta vacia! "
                        Salida: msg
                    Sino
                        maxTemperature(fechaNew)
                    Fsi
                (respMenu = 7):
                    Si (Vacia(fechaNew) = verdadero) entonces
                        msg <-- "¡La lista esta vacia!"
                        Salida: msg
                    Sino
                        orderPrecipitation(fechaNew)
                    Fsi
                (respMenu = 8):
                    Si (Vacia(fechaNew) = verdadero) entonces
                        msg <-- "¡La lista esta vacia!"
                        Salida: msg
                    Sino
                        speedWind(fechaNew)
                    Fsi
                (respMenu = 9):
                    Si (Vacia(fechaNew) = verdadero) entonces
                        msg <-- "¡La lista esta vacia!"
                        Salida: msg
                    Sino
                        backupFile(fechaNew, nomArch)
                    Fsi
                (respMenu = 10):
                    msg <-- Gracias..."
```

```
                                    salida: msg
                Fsegun
                guardoLista(fechaNew, nomArch)
        Fmientras
Fin

//||||||||||||||||||||FIN DEL PROGRAMA||||||||||||||||||||
```