**Algoritmica y**

**Programacion 1**

**Año 2019**

**GRUPO DOCENTE**

* **Profesor**: Dr. Diego FIRMENICH
* **JTP**: Esp. Lic. Lucy GUTIÉRREZ MARTICORENA
* **Auxiliares**: Lic. Leonardo MORALES. APU Lucas KRMPOTIC

Integrantes del Grupo:

Agustin Maximiliano Coronel | 37.067.605 |

Leandro David Espinoza | 38.800.816 |

algoritmo tp\_obligatorio2

(\*Integrantes: Agustin Coronel, Leandro Espinoza\*)

const

ARCHIVO\_BASE = "INSCRIPCIONES.CSV"

ARCHIVO\_ORDENADO = "INSCRIPCIONES\_ORDENADAS.DAT"

MAX\_ALUMNOS = 50

tipos

tFecha = arreglo [1..3] de entero

tAlumno = registro

legajo: cadena

dni: entero

apellido: cadena

nombre: cadena

fechaDeNacimiento: tFecha

carrera: cadena

fin registro

tArchivoBase = archivo de tTexto

tArchivoRegistro = archivo de tAlumno

tListado = arreglo [1..MAX\_ALUMNOS] de tAlumno

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento corteControl(E i :entero)

(\*

que hace: limita la cantidad de registros que se muestran en pantalla y muestra el numero de hoja

pre: i=I; I > 0

pos:--

\*)

var

enter:caracter

inicio

si i <> 0 entonces

mostrar("hoja numero : ",i div 3)

fin si

mostrar("presione enter para continuar")

ingresar(enter)

fin

(\*------------------------------------------------------------------------------------------------------------\*)

funcion separarCampos(E cad:cadena,p:entero):entero

(\*

que hace: encuentra la coma en una cadena de campos separados por coma.

pre: cad=C,p=P; P enRango(longitud(C))

pos: P= lugar donde se encuentra la coma

\*)

inicio

mientras ',' <> cad[p] y p < long(cad) hacer

p <- p + 1

fin mientras

separarCampos <- p

fin

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento ingresoFechaDeNacimiento (E cad:cadena,S fecha:tFecha)

(\*

que hace: dada una cadena conteniendo una fecha la transforma en entero y la ingresa

en un arreglo para luego poder ser prosesada

pre:cad=C; C es una cadena con una fecha en formato dia/mes/año

pos: fecha=F y F[1]= dia,F[2]=mes,F[3]=año

\*)

var

i,p:entero

inicio

p <- 1

j <- 1

desde i <- 1 hasta long(cad) hacer

si cad[i] = / o i = long(cad) entonces

fecha[j] <- cadenaAEntero(subcadena(cad,p,i-p))

si j=3 entonces

fecha [j] <- cadenaAEntero(subcadena(cad,p,i-p+1))

j <- j + 1

p <- i + 1

fin si

fin desde

fin

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento mostrarFecha(E fecha:tFecha)

(\*

que hace: muestra una fecha de nacimiento ingresada en un arreglo

pre: fecha=F;F[1]= dia,F[2]=mes,F[3]=año

pos:--

\*)

var

i:entero

inicio

mostrar("fecha de nacimiento:")

desde i <- 1 hasta 2 hacer

mostrar(fecha[i])

mostrar("/")

fin desde

i <- i+1

mostrar(fecha[i])

inicio

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento llenarRegistro(E cad:cadena,S alumno:tAlumno)

(\*

que hace: completa los campos de un registro a partir de una cadena separada por comas

pre: cad=C;C una cadena separada por comas

pos: alumno=A y A conteniendo los datos de C

\*)

var

p,j,campo:entero

inicio

p <- 1

desde campo <- 1 hasta 6 hacer

j <- separarCampos(cad,p)

segun campo hacer

1:alumno.legajo <- subcadena(cadAux,p,j-p)

2:alumno.dni <- cadenaAEntero(subcadena(cadAux,p,j-p))

3:alumno.apellido <- subcadena(cadAux,p,j-p)

4:alumno.nombre <- subcadena(cadAux,p,j-p)

5:ingresoFechaDeNacimiento(subcadena(cadAux,p,j-p),alumno.fechaDeNacimiento)

6:alumno.carrera <- subcadena(cadAux,p,j-p+1)

fin segun

p <- j + 1

fin desde

fin

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento llenarArreglo(E ruta:cadena,S listado:tListado,S dim:entero)

(\*

que hace: llena el arreglo de manera secuencial leyendo del archivo de texto especificado en la ruta

pre: ruta=R y R es la ruta del archivo de texto a leer

pos: listado=L,dim=D; listado[1..D]

\*)

var i:entero

Aux:cadena

archBase:tArchivoBase

inicio

abrir(archBase,ARCHIVO\_BASE)

dim <- tamaño(archBase)

desde i <- 1 hasta dim hacer

leerLN(archBase,cadAux)

llenarRegistro(cadAux,listado[i])

fin desde

cerrar(archBase)

fin

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento mostrarRegistro(E Alumno:tAlumno)

(\*

que hace:muestra los datos de los alumnos.

pre: A=alumno Y [1..A] con rango de tAlumno.

pos:--

\*)

var

i:entero

inicio

mostrar("nombre: ",alumno.nombre)

saltoLinea()

mostrar("apellido: ",alumno.apellido)

saltoLinea()

mostrar("dni: ",alumno.dni)

saltoLinea()

mostrar("legajo n°: ",alumno.legajo)

saltoLinea()

mostrar("fecha de nacimiento: ")

mostrarFecha(alumno.fechaDeNacimiento)

saltoLinea()

mostrar("carrera: ",alumno.carrera)

fin

(\*------------------------------------------------------------------------------------------------------------\*)

funcion compararNombres(E nuevo,actual:cadena):entero

(\*

que hace:compara dos nombres

pre: N1=nuevo,N2=actual

pos:compararNombres=C; C=-1 SI N1<N2; C=0 si N1=N2; C=1 si N1>N2.

\*)

var

i:entero

esFin:logico

inicio

si nuevo < actual entonces

compararNombres <- -1

sino

si nuevo > actual entonces

compararNombres <- 1

sino

compararNombres <- 0

fin si

fin si

fin

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento intercambio(E/S alumno1, alumno2: tAlumno)

(\*

que hace:intercambia dos valores

pre: alumno1=A; alumno2=B

pos: alumno1=B; alumno2=A

\*)

var

aux:tAlumno

inicio

aux <- alumno1

alumno1 <- alumno2

alumno2 <- aux

fin

(\*------------------------------------------------------------------------------------------------------------\*)

funcion fechaEnNumero (E fecha :tfecha):entero

(\*

que hace:compara dos fechas

pre: F=fecha;

pos: fechaEnNumero=N y N > 0 representa la fecha N.

\*)

inicio

fechaEnNumero <- fecha[3] \* 10000 + fecha[2] \* 100 + fecha[1]

fin

(\*------------------------------------------------------------------------------------------------------------\*)

funcion compararFechas(E alum1,alum2:tAlumno):entero

(\*

que hace:compara dos fechas de nacimiento

pre: A1=alumn1 , A2=alumn2

pos: compararFechas=F ;F=-1 si A1<A2; F=0 si A1=A2 ; F=1 si A1>A2.

\*)

var

a1,a2:entero

inicio

a1 <- fechaEnNumero (alum1.fechaDeNacimiento)

a2 <- fechaEnNumero (alum2.fechaDeNacimiento)

si a1=a2 entonces

compararFechas <- 0

sino

compararFechas <- (a1-a2) div abs(a1-a2)

fin si

fin

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento ordenarFechaNac (E/S listado:tListado, E principio,final:entero)

(\*

que hace: ordena los elmentos del arreglo de menor a mayor por fecha de nacimiento

pre: listado=L,Principio=P,final=F; [P..F] en rango(tListado)

pos: L=L', L' Esta ordenada en [P..F];

\*)

var

max,tope:entero

ordenado:logico

inicio

max <- final

repetir

ordenado<- verdadero

tope<- max-1

desde i <- principio hasta tope hacer

si compararFechas(listado[i].fecha,listado[i+1].fecha) >0 entonces

intercambio(listado[i],listado[i+1])

max <- i

ordenado <- falso

fin si

fin desde

hasta ordenado=verdadero

fin

procedimiento listarFechaNac(E listado:tListado,E dim:entero)

(\*

que hace: muestra un listado de los registros del arreglo de a 3 personas por orden de edad de menor a mayor

pre: listado=L,dim=D; L[1..D]

pos:--

\*)

var

i:entero

inicio

ordenarFechaNac(listado,1,dim)

desde i <- i hasta dim hacer

mostrarRegistro(listado[i])

si i mod 3 = 0 entonces

corteControl(i)

fin si

fin desde

fin

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento insertarAlumnoAlfabetico(E/S archRegistro:tArchivoRegistro,E nuevo:tAlumno)

(\*

que hace: agrega un nuevo registro al archivo respetando el orden

pre: archRegistro=f, nuevo=N; F esta abierto y ordenado.

pos: F=F' y F' esta ordenado y F' = F U {N}

\*)

var

pos: entero

aux: tAlumno

encontroLugar: logico

inicio

pos <- tamaño(archRegistro) - 1

encontroLugar <- falso

mientras (pos >= 0) y (no encontroLugar) hacer

posicionarse(archRegistro,pos)

leer(archRegistro,aux)

si compararNombres(nuevo.nombre+nuevo.apellido,aux.nombre+aux.nombre) < 0 entonces

escribir(archRegistro,aux)

pos <- pos - 1

sino

encontroLugar <- verdadero

fin si

fin mientras

posicionarse(archRegistro,pos + 1)

escribir(archRegistro,alumno)

fin

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento crearAlfabetico(E listado:tListado,E dim:entero)

(\*

que hace: genera un archivo .DAT conteniendo los registros del arreglo por orden Alfabetico

pre: listado=L,dim=D; L es un arreglo conteniendo los registros de alumnos y L[1..D]

pos:--

\*)

var

archRegistro: tArchivoRegistro

i:entero

inicio

crear(archRegistro,ARCHIVO\_ORDENADO)

desde i <- 1 hasta dim hacer

insertarAlumnoAlfabetico(archRegistro,listado[i])

fin desde

cerrar(archRegistro)

fin

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento listarAlfabetico(E ruta:cadena)

(\*

que hace: muestra un listado de los registros del archivo ordenado alfabeticamente de a 3 personas

pre: ruta=R; R es la ruta a donde esta el archivo .DAT

pos:--

\*)

var

alumno:tAlumno

i:entero

archRegistro:tArchivoRegistro

inicio

crearAlfabetico(listado,dim)

mostrar("archivo creado")

corteControl(0)

abrir(archRegistro,ARCHIVO\_ORDENADO)

i <- 1

mientras no eof(archRegistro) hacer

leer(archRegistro,alumno)

mostrarRegistro(alumno)

i <- 1 + 1

si i mod 3 = 0 entonces

corteControl(i)

fin si

fin mientras

cerrar(archRegistro)

fin

(\*------------------------------------------------------------------------------------------------------------\*)

funcion seleccion(E msg,error:cadena):entero

(\*

que hace: muestra un menu para que el usuario elija que operacion desea realizar

pre: msg=M,error=E; M es el mensaje para el usuario, E es el mensaje de error que se muestra

si el usuario no elige una opcion adecuada

pos: seleccion=S; S es la opcion elegida por el usuario y S enRango[1..4]

\*)

var

op:entero

inicio

repetir

mostrar(msg)

saltoLinea()

mostrar("1: listar alumnos por fecha de nacimiento")

saltoLinea()

mostrar("2: listar alumnos por carrera")

saltoLinea()

mostrar("3: listar y generar archivo con alumnos ordenados alfabeticamente")

saltoLinea()

mostrar("4: salir")

saltoLinea()

si op < 1 o op > 4 entonces

mostrar(error)

fin si

hasta op en [1..4]

seleccion <- op

fin

(\*------------------------------------------------------------------------------------------------------------\*)

procedimiento listarCarrera(E listado:tListado,E dim:entero)

(\*

que hace: muestra un listado de los registros del arreglo segun carrera de a 3 personas

pre: listado=L,dim=D; L[1..D]

pos:--

\*)

var

op,i,j: entero

c: caracter

inicio

repetir

mostrar("Elija carrera a listar: ")

mostrar("1: Analista Programador Universitario")

mostrar("2: Licenciatura en sistemas")

ingresar(op)

hasta op en [1..2]

segun op hacer

1: c <- 'A'

2: c <- 'L'

fin segun

desde i <- 1 hasta dim hacer

si c = listado[i].carrera[1] entonces

j <- j+1

mostrarRegistro(listado[i])

si j mod 3 = 0 entonces

cortecontrol(j)

fin si

fin si

fin desde

fin

(\*------------------------------------------------------------------------------------------------------------\*)

var

archBase: tArchivoBase

listado:tListado

dim:entero

inicio

llenarArreglo(ARCHIVO\_BASE,listado,dim)

repetir

op <- seleccion("Elija una Opcion","opcion erronea")

segun op hacer

1: listarFechaNac(listado,1,dim)

2: listarCarrera(listado,dim)

3: generarAlfabetico(listado,dim)

4: salir <- verdadero

fin segun

limpiarPantalla();

hasta salir

fin

(\*------------------------------------------------------------------------------------------------------------\*)