UTN – FACULTAD REGIONAL BUENOS AIRES
Materia: Algoritmos y Estructura de Datos – 2020
CURSO K1042

TRABAJO PRÁCTICO

Informe

Nombre:
Martin Fernando Fedorenko

N° Legajo:
1772399

Correo:
mfedorenko@est.frba.utn.edu.ar
martinfedorenkolk1288@gmail.com

Aclaraciones e hipótesis:

El buffer infracciones inf[100] del subprograma procesarlote(), tiene ese tamaño ya que se necesitaba un buffer "grande" según la consigna.

Asimismo en el subprograma procesarlote() se utiliza un getch junto con un con un condicional para poder salir de la iteración al final de cada "vuelta" en el lote de infracciones.

En el tipo de dato "conductor" se usan arrays de caracteres de tamaño 6 debido a lo establecido en los registros de la consigna. Lo mismo ocurre en el tipo de dato "infracción" (exceptuando el elemento fechahora[13]) y "listaconductores".

En el programa principal se agregó el subprograma creardatosdepruebaconductores() para crear datos de prueba con fines prácticos durante el desarrollo y las pruebas del programa.

En el subprograma guardarconductor() solo se permite ingresar ID de conductor, fecha de vencimiento de registro y email, evitando los datos como activo (ya que se asume que el nuevo conductor es activo al registrarse) y totalinfracciones (ya que dichas infracciones no aparecerían en el archivo de infracciones histórico "procesados.bin").

En varias partes del programa se agregan "\n" o "endl" de más con fines estéticos y para facilitar la visualización en la ejecución del programa.

En el subprograma buscarconductor(), se decidió aunar mediante un switch y un pequeño menú la capacidad de buscar, desactivar conductores y mostrar infracciones totales y montos totales, ya que como subprogramas separados tendrían una estructura similar y de esta manera se ahorra tiempo, esfuerzo y recursos.

En el programa listarporprovincia(), las provincias se obtienen del archivo "Procesados.bin".

Antes de mostrar el menú principal se pide al usuario ingresar la fecha del día en la que se realizan las operaciones, ya que facilita la verificación de registros vencidos y no cae en la inexactitud que puede tener el archivo "Procesados.bin", en cuanto a la fecha de las infracciones.

Es recomendable no ingresar datos muy rápido ni sobrepasarse del número de caracteres estipulado por pantalla, ya que podría impedir el normal funcionamiento del programa.

El tipo de dato listaconductores fue creado exclusivamente como medio para tener un array de Ids de conductores (que tienen varios caracteres) en la función listarporprovincia().

Para mayor comodidad se definen como "filename1" y "filename2" los archivos "Conductores.bin" y "Procesados.bin".

Exposición de los subprogramas:

Menu()

Funciona de nexo entre los demás subprogramas. Mediante un getch (el cual solo acepta los caracteres indicados junto con ESC para salir) que desemboca en un switch que activa las opciones deseadas.

Creardatosdepruebaconductores()

Subprograma encargado de crear datos de tipo "conductor" mediante asignaciones y la función strcpy (cuando se trata de copia de cadena de caracteres). Estos datos sirven de material para las pruebas llevadas a cabo durante el desarrollo. Cabe destacar que los datos creados son almacenados en el archivo "Conductores.bin".

Guardarconductor()

Subprograma que permite añadir registros de conductores individuales al archivo "conductores.bin", o en su defecto crear este mismo. Se compone de varias estructuras de salida (que le indican al usuario que dato ingresar) y de entrada (que datos ingresan al programa).

Mostrarconductores()

Subprograma inverso al anterior. En vez de escribir registros de conductores en el archivo "conductores.bin", los lee de este y luego los muestra por pantalla. Es importante mencionar que muestra todos los registros de conductores que hay en el archivo "conductores.bin" hasta que este llega a su fin.

Buscarconductor()

Subprograma con un menú independiente cuyo principal objetivo es permitir la búsqueda de registros de conductores particulares en el archivo "Conductores.bin" mediante el ingreso del mail o ID del conductor, para luego mostrar sus datos por pantalla o cambiarlos (más precisamente desactivar a los conductores). El programa luego de recibir el mail o ID los compara con los datos del archivo, de encontrarlos procede a los procesos mencionados, de lo contrario muestra por pantalla que el registro buscado no existe.

Listarporprovincia()

Subprograma encargado de buscar conductores que hayan cometido infracciones en determinadas provincias (que se ingresan por teclado), para luego mostrar dichos conductores por pantalla. Al comienzo el algoritmo pide que se ingrese el código de la provincia, el cual se comparara con los registros de infracciones históricos ("Procesados.bin"). Si las provincias coinciden, se selecciona el registro de infracción en el que se encuentra el cursor y se compara el código ID del conductor con un vector que servirá de "buffer" para mostrar por pantalla a los conductores (esta comparación se realiza para evitar que se repitan conductores en la misma provincia). De no coincidir las provincias en ningún momento (en la comparación de la provincia ingresada y el archivo histórico), implica que ningún conductor registrado ha cometido una infracción en dicha provincia.

Procesarlote()

Subprograma encargado de procesar los lotes de infracciones, escribiéndolos en el archivo "Procesados.bin" y actualizando los datos de los conductores de "Conductor.bin". Comienza pidiendo todos los elementos pertenecientes al tipo de dato infraccion. Luego compara el ID

de conductor ingresado, con todos los ID del archivo “Conductores.bin”, de haber coincidencia se procede a aumentar el número de infracciones y el monto total del conductor, para luego sobrescribir el registro individual con los nuevos datos. Sin embargo, sino se logrará encontrar el ID correspondiente al ingresado, se pide que se registre al nuevo conductor (invocando la función Guardarconductor()). Habiendo registrado al conductor se pide que se vuelvan a escribir los datos de la infracción para así, si poder actualizar los datos del registro de conductores. Después de lidiar con el archivo de “Conductores.bin”, se agregan al archivo “Procesados.bin” los datos del lote recolectados en esa “vuelta”. Por último, luego de la escritura en el archivo, se propone al usuario la opción de seguir registrando lotes o retirarse al menú principal. Cabe destacar que el buffer encargado de los lotes es relativamente grande por lo cual se implementó la funcionalidad anterior.

Finalizarjornada()

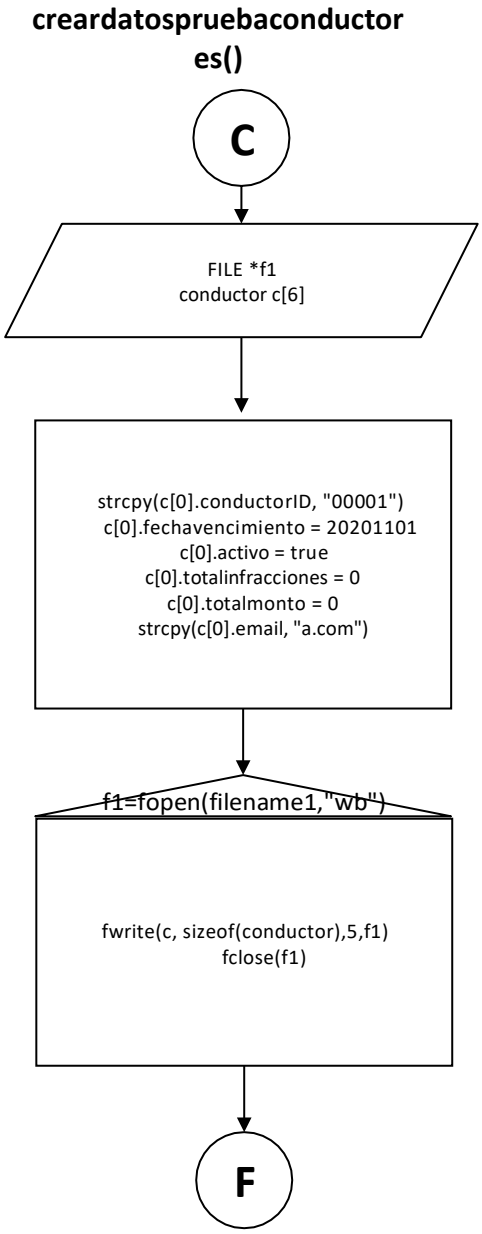
Única función a la cual se accede justo antes de finalizar la ejecución del programa. Esta se encarga de filtrar a todos los conductores que no tengan vencido el registro o no hayan sido desactivados, para luego ser reescritos en un archivo auxiliar (“auxiliar.bin”) que luego pasara a tener el nombre de “Conductores.bin” después de haberse borrado el original. Cabe destacar que esta función requiere que al comienzo del programa se ingrese la fecha actual, a fin de corroborar la fecha ingresada con la fecha de vencimiento de los registros, y así tener un filtro más confiable y preciso.

División de tareas:

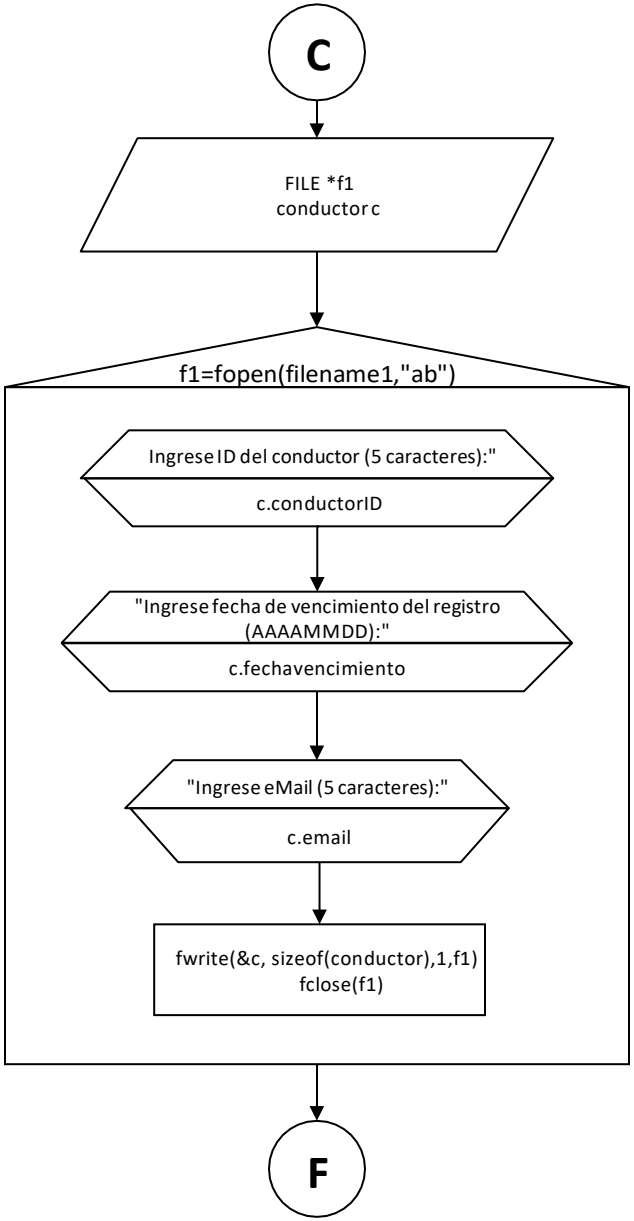
1. Estudio de la consigna, requerimientos y formulación de ideas preliminares.
2. Formulación formal de que subprogramas debe haber.
3. Comienzo de la escritura de cada subprograma junto con su pertinente prueba.
4. Pruebas generales y de adaptación entre subprogramas.

Programa realizado por Martin Fedorenko

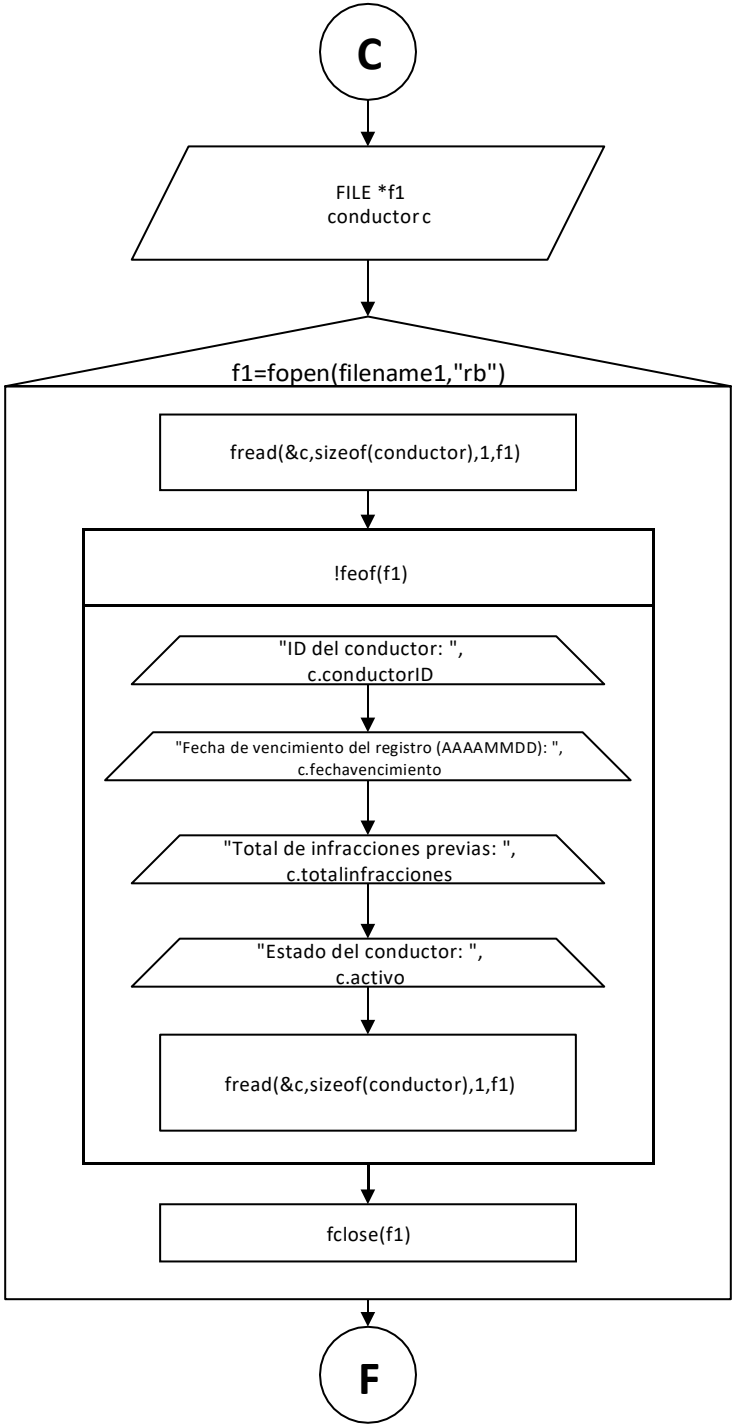
Diagramas de bloques (Lindsay)



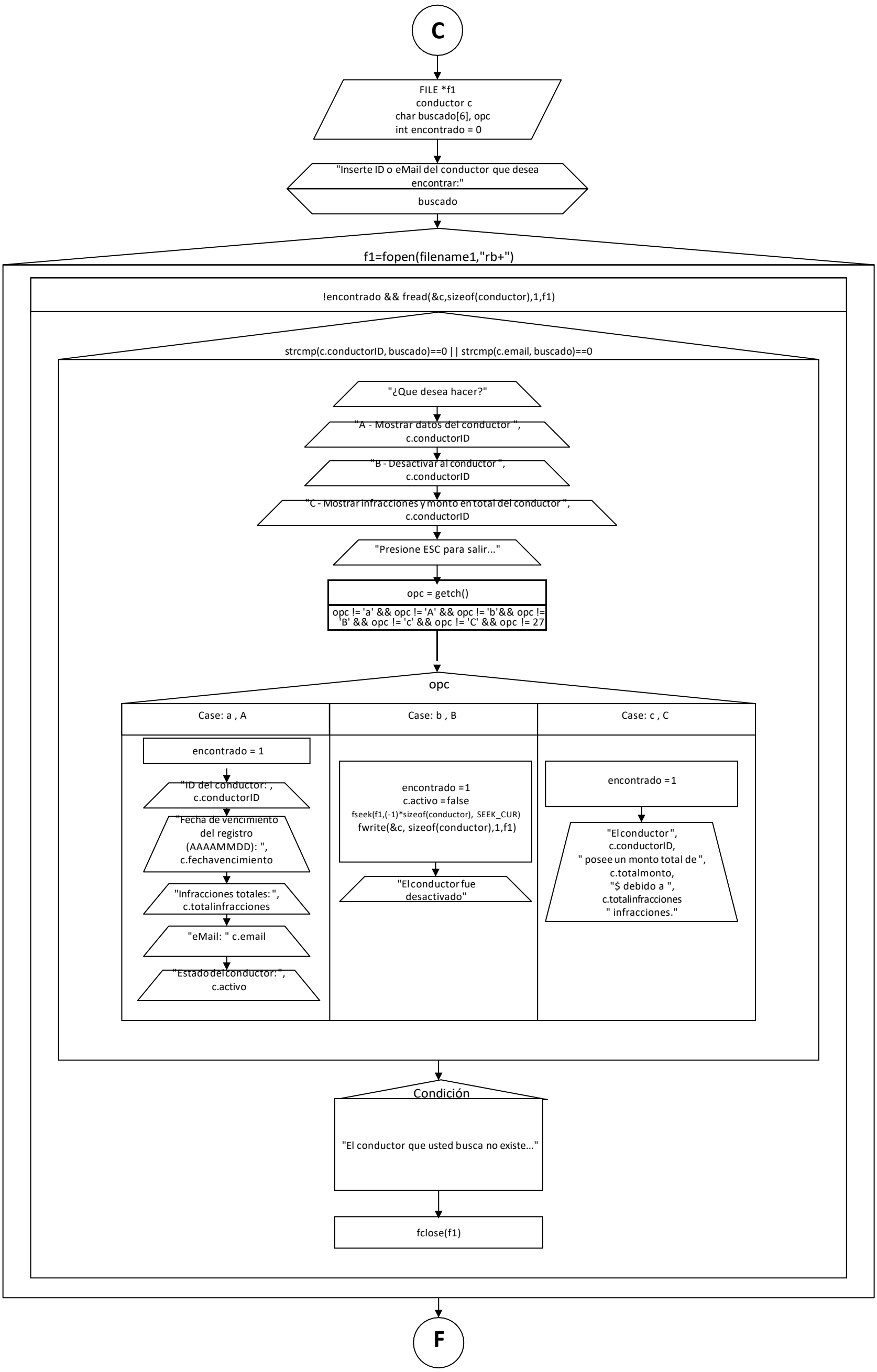
guardarconductor()



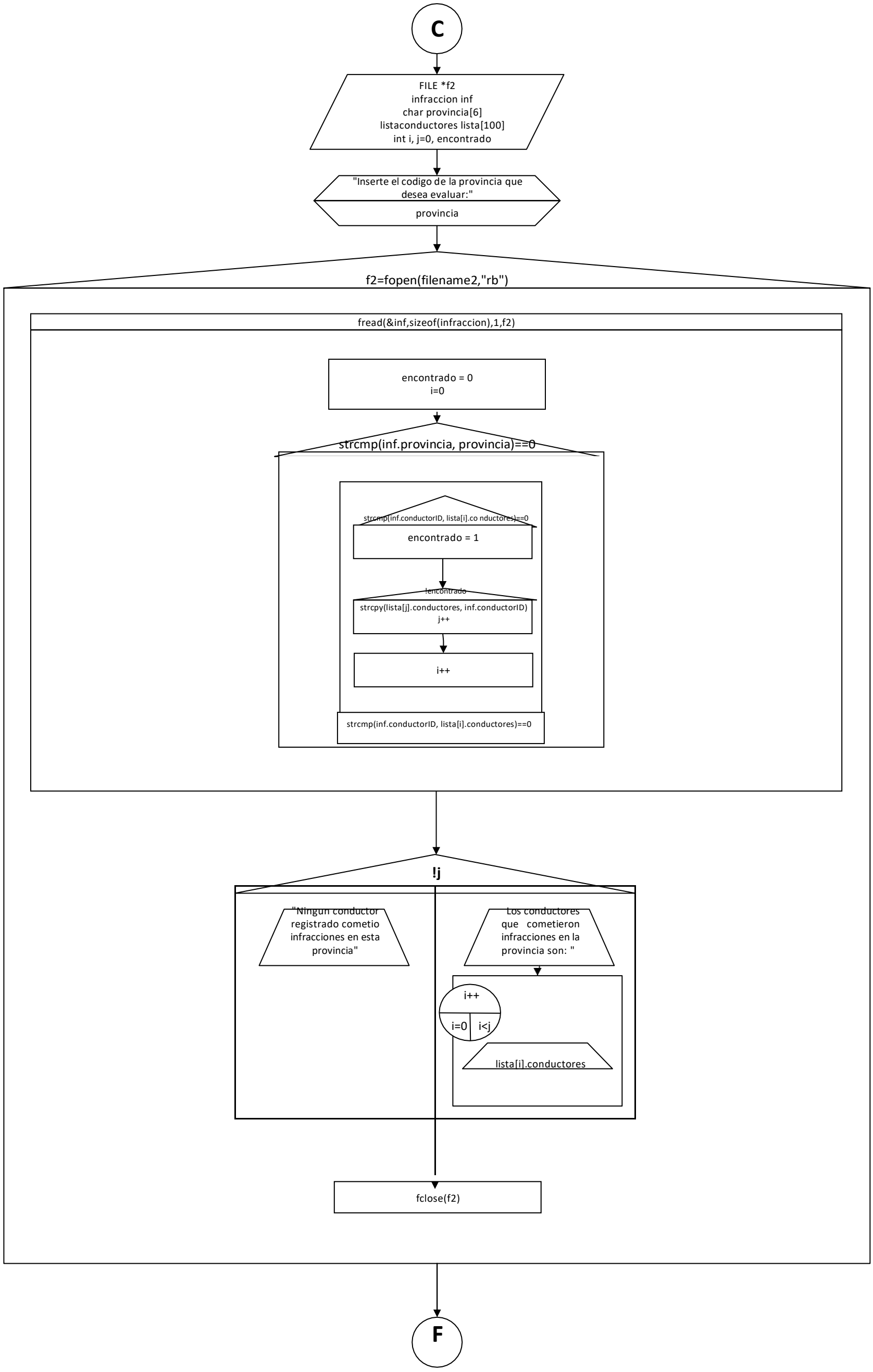
mostrarconductores()



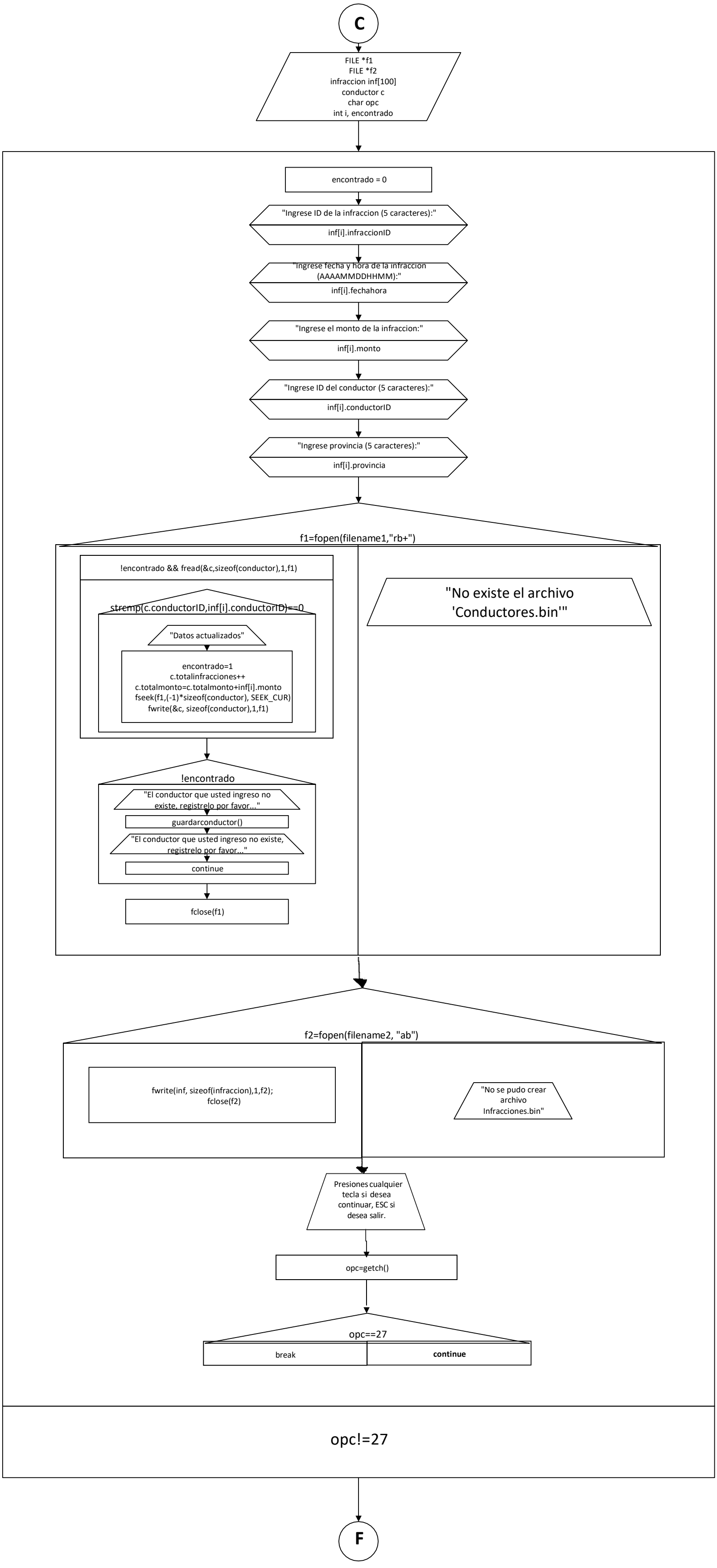
buscarconductor()



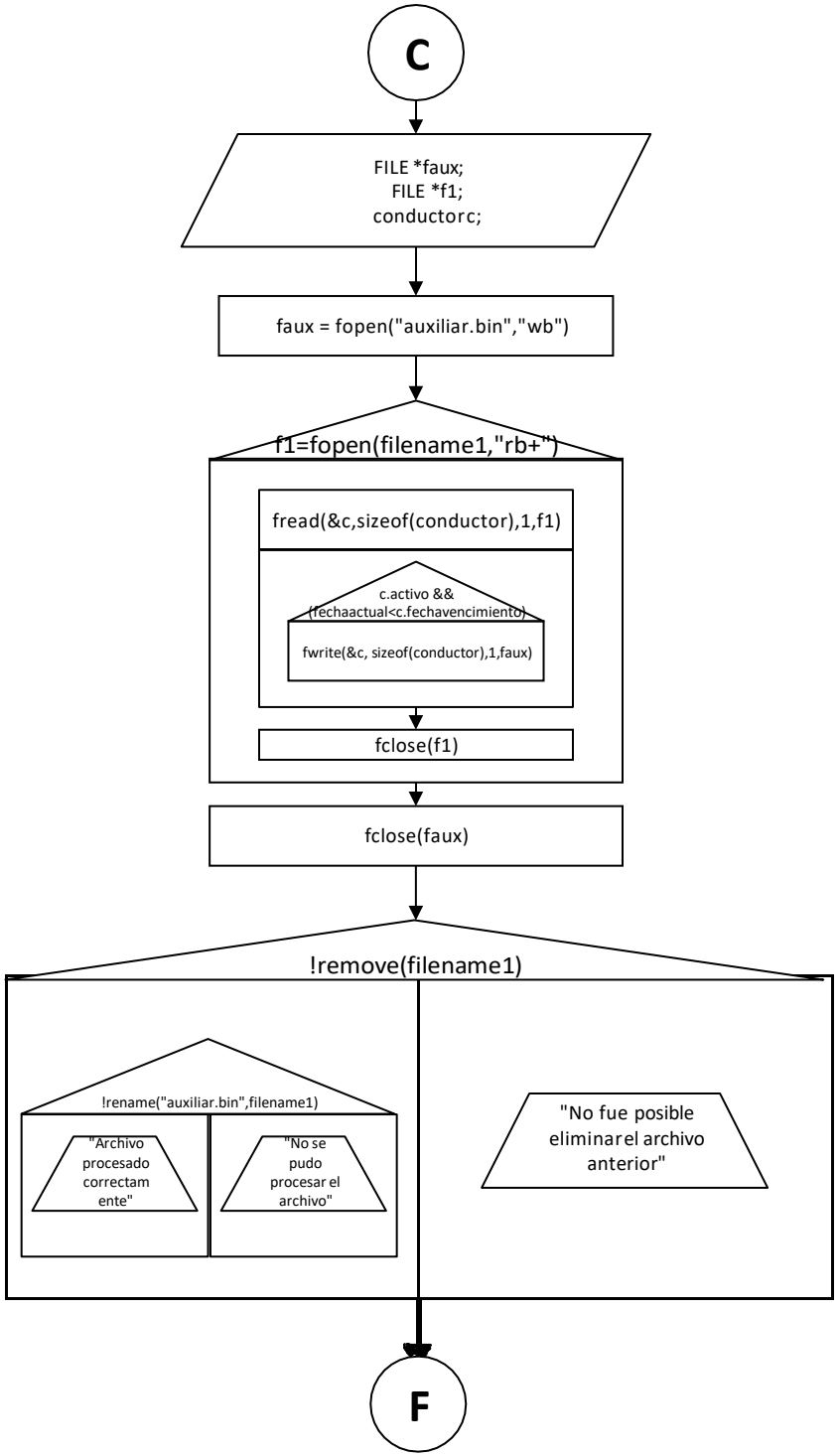
listarporprovincia()

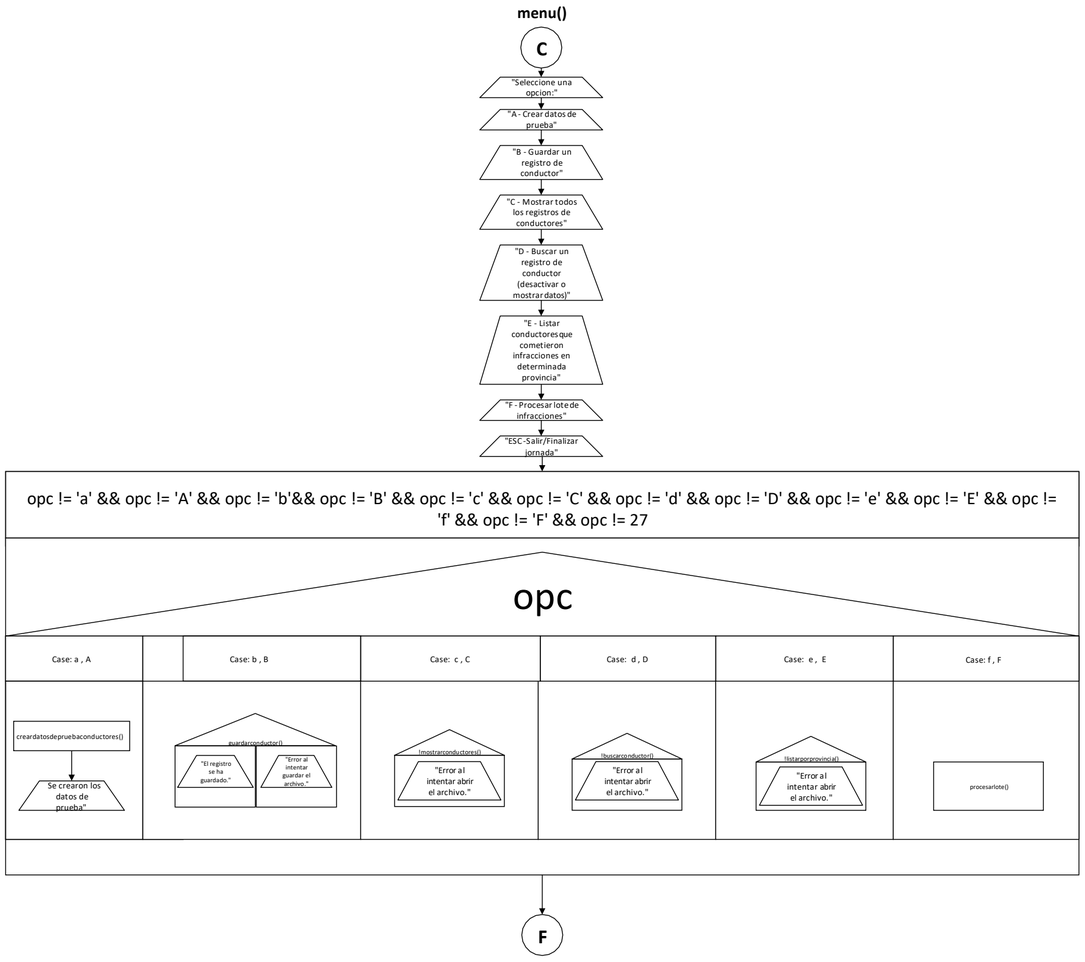


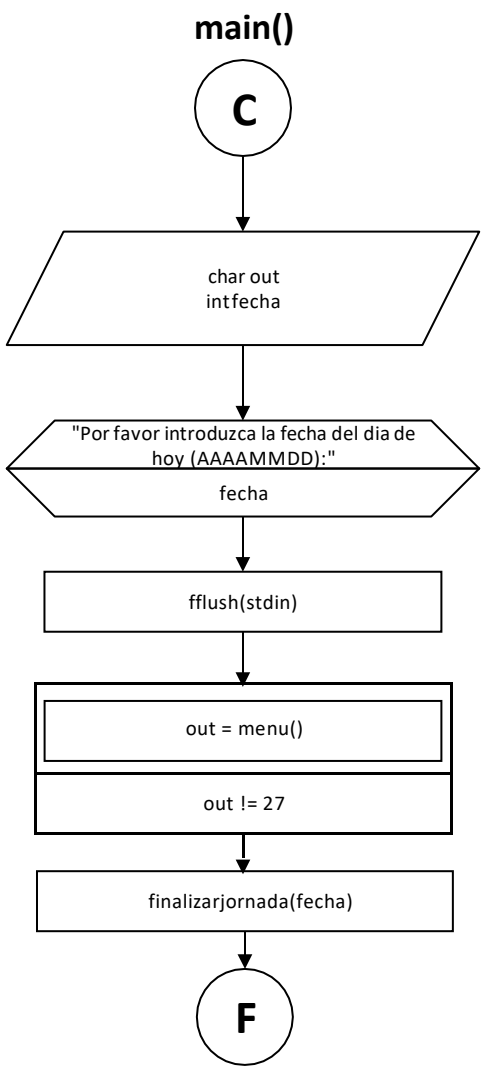
procesarlote()



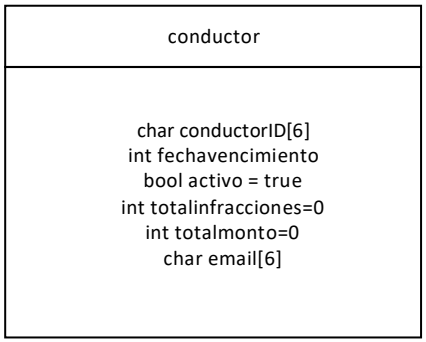
finalizarjornada()



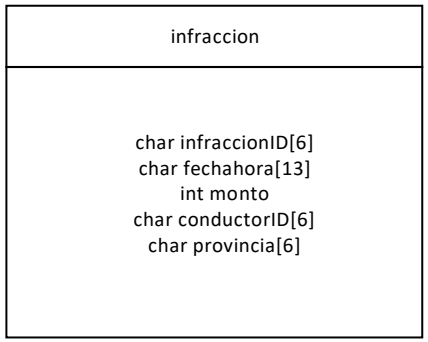

menu()
C
"Seleccione una opcion:"
"A - Crear datos de prueba"
"B - Guardar un registro de conductor"
"C - Mostrar todos los registros de conductores"
"D - Buscar un registro de conductor (desactivar o mostrar datos)"
"E - Listar conductores que cometieron infracciones en determinada provincia"
"F - Procesar lote de infracciones"
"ESC-Salir/Finalizar jornada"
opc != 'a' && opc != 'A' && opc != 'b'&& opc != 'B' && opc != 'c' && opc != 'C' && opc != 'd' && opc != 'D' && opc != 'e' && opc != 'E' && opc != 'f' && opc != 'F' && opc != 27
opc
Case: a , A
Case: b , B
Case: c , C
Case: d , D
Case: e , E
Case: f , F
creardatosdepruebaconductores()
Se crearon los datos de prueba"
guardarconductor()
"El registro se ha guardado."
"Error al intentar guardar el archivo."
[mostrarconductores()]
"Error al intentar abrir el archivo."
[buscarconductor()]
"Error al intentar abrir el archivo."
[listarporprovincia()]
"Error al intentar abrir el archivo."
procesarlote()
F


main()
C
char out
intfecha
"Por favor introduzca la fecha del dia de hoy (AAAAMMDD):"
fecha
fflush(stdin)
out = menu()
out != 27
finalizarjornada(fecha)
F

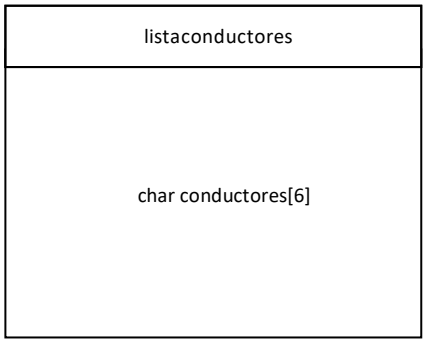
Structs creados y utilizados a lo largo del trabajo



Tipo de dato conductor que como elementos posee el ID de conductor, la fecha de vencimiento del registro de cada conductor, el estado del conductor (si está activo o no), el total de infracciones de cada conductor (que se inicia siempre en 0 al ingresar un nuevo conductor), el monto total por las infracciones (que también se inicializa en 0) y por último el email de cada conductor.
Este tipo de dato se utiliza a lo largo de todo el programa.



Tipo de dato infracción que como elementos posee el ID de la infracción, la fecha y hora en que se hizo, el monto que equivale la infracción, el ID del conductor que la realizo y por último en qué provincia se llevó a cabo la infracción.
Este dato al igual que conductores se usa en gran parte de programa.



Tipo de dato listaconductores con un único elemento, conductores.
Este tipo de dato solo se utiliza exclusivamente en la función listarporprovincia como medio para tener un array de IDs de conductores.