TRABAJO PRÁCTICO
"INFORME"
ALGORITMOS Y ESTRUCTURA DE DATOS

UNIVERSIDAD TECNOLÓGICA NACIONAL
CURSO: K1024
PROFESOR: INGENIERO PABLO D. MENDEZ

Integrantes:

- Diego Hernández Canetti
  - Legajo: 176.198-5
  - Mail: dhernandezcanetti@est.frba.utn.edu.ar
- Micaela Paredes
  - Legajo: 176.128-6
  - Mail: miparedes@est.frba.utn.edu.ar
- Martina Roldán Pérez
  - Legajo: 176.230-8
  - Mail: mroldnprez@est.frba.utn.edu.ar

Hipótesis de la solución:

Librerías que decidimos utilizar:

Para guardar, mostrar, desactivar y buscar: #include <stdio.h>, #include <stdlib.h>, #include <string.h>, #include <iostream>, #include <conio.h>
Para ordenar: #include <fstream>
Para guardar el momento en que la persona ingresa un lote de compras: #include <time.h>

Estructuras y algoritmos utilizados:

Estructura guardar:

Al elegir la opción ***guardar*** en el menú, el usuario puede ingresar un nuevo cliente al archivo.

El proceso comienza cuando el documento se abre y se le pide al usuario que ingrese los datos, los cuales están guardados en el struct ***CLIENTES***. Cuando se ingresa el cliente, antes de proseguir con la carga de datos, se verifica que el cliente no haya sido previamente ingresado, para que a la hora de actualizarse el total de cliente no haya errores. Si no es así, se guarda el cliente con éxito.

```
113   bool GUARDAR(void)
114   {
115       FILE *f;
116       CLIENTE c;
117       int existe=false;
118       bool esnum=false;
119       if (f=fopen(NOMBREARCHIVO,"ab"))
120       {
121           cout << "Ingrese usuario ID:" << endl;
122           cin >>  c.USUARIO;
123           existe=EXISTE(c.USUARIO);
124           if (existe==false)
125           {
126               cout << "Ingrese fecha de creacion:" << endl;
127           cin >> c.FECHA;
128           cout << "Ingrese si esta activo (si/no):" << endl;
129           cin >> c.ACTIVO;
130           if (strcmp("si", c.ACTIVO)==0 || strcmp("SI", c.ACTIVO)==0 ){
131               c.activado=true;
132           } else{
133           c.activado=false;
134           }
135          cout << "Ingrese total:" << endl;
136           cin >> c.TOTAL;
137           cout << "Ingrese email:" << endl;
138           cin >> c.EMAIL;
139           fwrite(&c, sizeof(CLIENTE),1,f);
140           fclose(f);
```

```
141           return true;
142           }else
143           {
144               printf("El usuario ya existe en el registro\n");
145               fclose(f);
146           }
147
148       }
149       return false;
150   }
151
```

Estructura Sumar monto total:

Esta estructura es utilizada al momento de guardar un nuevo lote ingresado.

El algoritmo llamado ***guardar lote*** llama al void ***sumar monto total*** para que actualice los montos de los clientes.

Esta función recibe los datos *usuarios* y *monto,* que se guardaron en *guardar lote* y abre el archivo ***clientes.bin.*** Luego, se recorre el archivo para encontrar al cliente solicitado, si se encuentra, retrocede el puntero hasta el inicio del registro que buscó y se escribe sobre este. En caso de no ser encontrado el cliente solicitado, esto será avisado al usuario a través de una leyenda utilizando un IF.

```
54  int SUMARMONTOATOTAL(char USUARIO[],int MONTO)
55  {
56      FILE *f;
57      int encontrado =0;
58      int TOTALACTUA=0;
59      struct CLIENTE c;
60      if (f=fopen(NOMBREARCHIVO,"r+b"))
61      {
62          while(!encontrado && fread(&c,sizeof(struct CLIENTE),1,f))
63          {
64              if (strcmp(USUARIO, c.USUARIO)==0)//porque si son iguales
65              {
66             TOTALACTUA= MONTO+c.TOTAL;
67             c.TOTAL=TOTALACTUA; //es i porque es int
68             int pos=ftell(f)-sizeof(CLIENTE); // ftell retorna posicio
69             fseek(f,pos,SEEK_SET); //retrocede el puntero de archivo
70             fwrite(&c, sizeof(CLIENTE), 1, f);
71             printf("Se modifico el importe para dicho cliente.\n");
72             encontrado=1;
73             break;
74              }
75          }
76          if (!encontrado)
77              printf("No se encontro la persona buscada.\n");
78          fclose(f);
79          return 1;
80      }
81      return 0;
```

Estructura ordenar:

Utilizamos la librería <fstream> para abrir los archivos in y out y buscar toda la información necesaria. Lo que hace el algoritmo es ir al final del fichero y contar los registros que existen en **clientes.bin**. Luego se crea un array con la medida de los clientes anteriormente contados. Se vuelve al puntero del inicio donde se lee de nuevo el archivo clientes.bin y se pasa cada registro al array que se creó antes.

A partir de aquí se llama a la estructura burbuja utilizando el algoritmo del Bubble Sort, este te muestra, usando un for, todos los registros que están activados.

Finalmente, se cierran los dos archivos y el array creado se elimina para que no queden los registros anteriores.

```
298  void  ORDENAR ()
299  {
300      CLIENTE c, *array;
301      int tam = sizeof(CLIENTE), totalclientes, i=0;
302      ifstream in;
303      ofstream out;
304      in.open("CLIENTES.bin", ios::binary);
305      if(in.fail())
306      {
307          cout << "Error al abrir el fichero" << endl;
308          system("pause");
309          exit(1);
310      }
311      out.open("ORDENADOS.bin", ios::binary);
312      if(out.fail())
313      {
314          cout << "Error al crear el fichero" << endl;
315          system("pause");
316          exit(1);
317      }
318      in.seekg(0,ios::end);
319      totalclientes=in.tellg()/tam;
320      array = new CLIENTE[totalclientes];
321      if(array==NULL)
322      {
323          cout << "Error en la asignación de memoria\n";
324          system("pause");
325          exit(1);
326      }
```

```
326  }
327  in.seekg(0);
328
329  in.read((char *) &array[i], tam);
330  while(!in.eof())
331  {
332       i++;
333       in.read((char *) &array[i], tam);
334  }
335
336
337   BURBUJA(array,totalclientes);
338
339    for(i=0;i<totalclientes;i++)
340  {
341      if (array[i].activado==true){
342          cout << "******* Datos de la persona *******" << endl;
343          cout << "USUARIO: " << array[i].USUARIO << endl;
344          cout << "FECHA: " << array[i].FECHA << endl;
345          cout << "ACTIVO: " << array[i].activado << endl;
346          cout << "TOTAL: " << array[i].TOTAL << endl;
347          cout << "EMAIL: " << array[i].EMAIL << endl;
348          out.write((char *) &array[i], tam);
349  }
350  }
351  in.close();
352  out.close();
353  delete [] array; //se elimina el array para  cada vez  que se inicia el programa no guarde basura
354  }
```

Estructura finalizar:

En esta estructura se utilizan dos archivos, el original y otro auxiliar. Del archivo original se buscan todos los clientes que estén activados a través de un IF y se los reescribe en el archivo auxiliar. Luego se borra el archivo original y es reemplazado por el auxiliar.

```
355
356   int FINALIZAR()
357   {
358       FILE *faux;
359               FILE *f;
360               CLIENTE c;
361               faux = fopen("clientes2.bin","wb");
362               if (f=fopen(NOMBREARCHIVO,"rb+"))
363               {
364                   while(fread(&c,sizeof(CLIENTE),1,f))
365                   {
366                       if (c.activado==true)
367                       {
368                           fwrite(&c, sizeof(CLIENTE),1,faux);
369                       }
370                   }
371                   fclose(f);
372               }
373               fclose(faux);
374               if (!remove(NOMBREARCHIVO))
375               {
376                   if(!rename("clientes2.bin",NOMBREARCHIVO))
377                       cout << "Archivo renombrado exitosamente" << endl;
378                   else
379                       cout << "No se pudo renombrar el archivo" << endl;
380               }
381               else
382                   cout << "No se pudo borrar el viejo archivo" << endl;
383       }
```

Ventajas y desventajas del programa:

Ventajas:

-Es de fácil entendimiento tanto para el usuario a la hora de usarlo como para una persona que lee el código al tener estructuras simples de archivos binarios.

Desventajas:

-Al ser muy pesado el archivo por la cantidad de código puede tardar en cargar al iniciarlo o compilarlo.

-Si se ingresa un lote de compras con un usuario id que no existe en CLIENTES.bin este se guarda pero sin modificar ningún cliente del archivo.

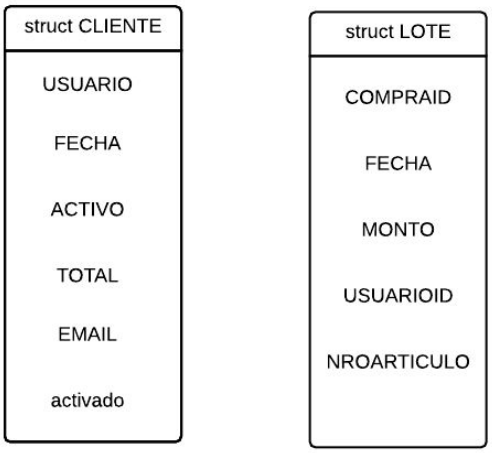
División de tareas:

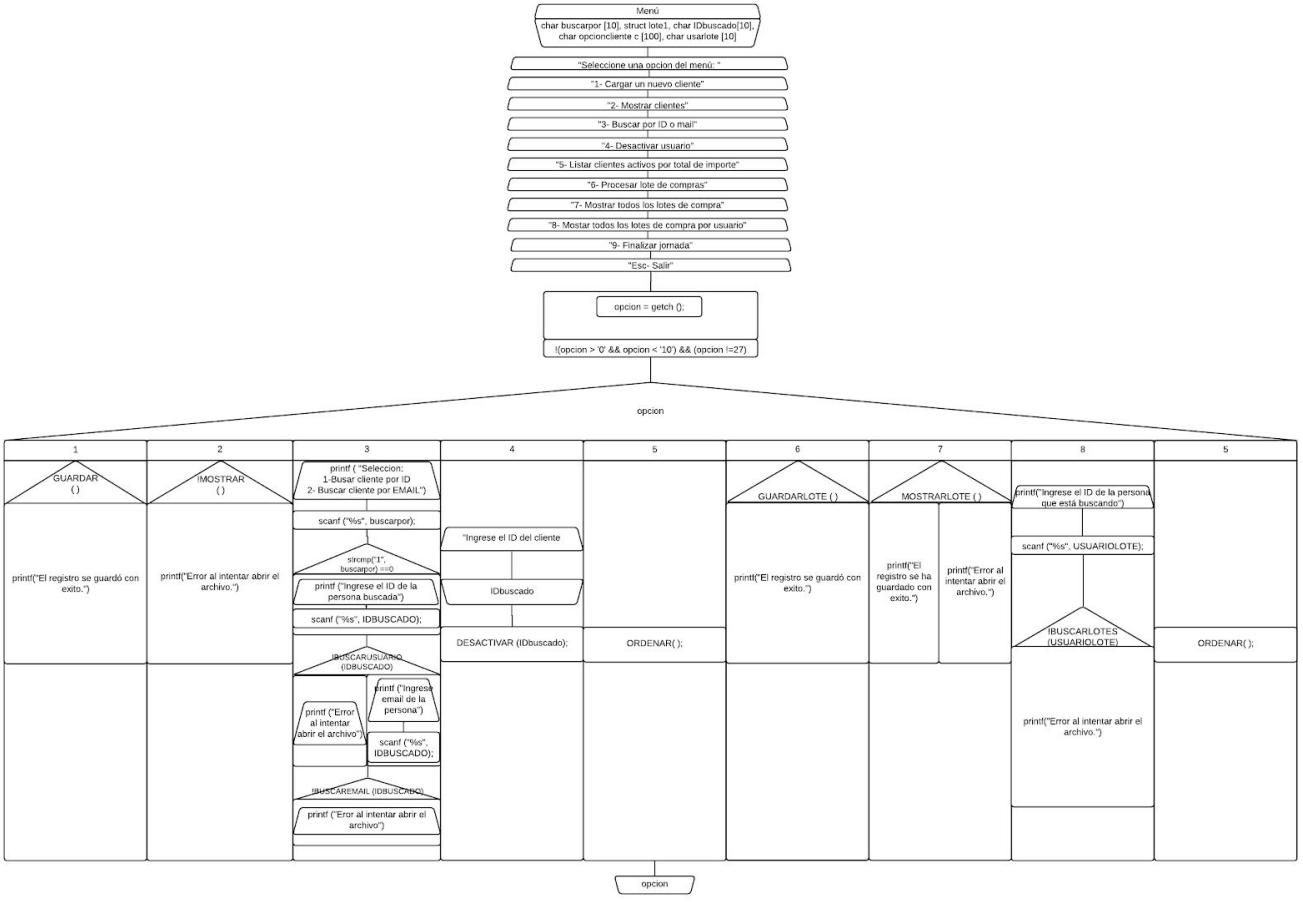
- Código:
  - Hernández Canetti: encargado de las estructuras ***buscar usuario***, ***buscar email*** y ***buscar lotes***.
  - Paredes: encargada de las estructuras ***sumar monto total***, ***total***, ***existe***, ***mostrar lote***, ***burbuja***, ***ordenamiento***, ***finalizar***, ***desactivar*** y ***mostrar***.
  - Roldán Pérez: encargada del ***menú***, unión de estructuras, y las estructuras ***guardar*** y ***guardar lote***.
- Diagramas de bloques:
  - Hernández Canetti: encargado de las estructuras de ***buscar usuario***, ***buscar email***, ***buscar lotes*** y ***ordenamiento***.
  - Paredes: encargada de las estructuras de ***finalizar***, ***sumar monto total***, ***total***, ***mostrar lote*** y ***mostrar***.

- Roldán Pérez: encargada de las estructuras del ***menú***, ***guardar***, ***guardar lote***, ***existe***, ***desactivar*** y ***burbuja***.

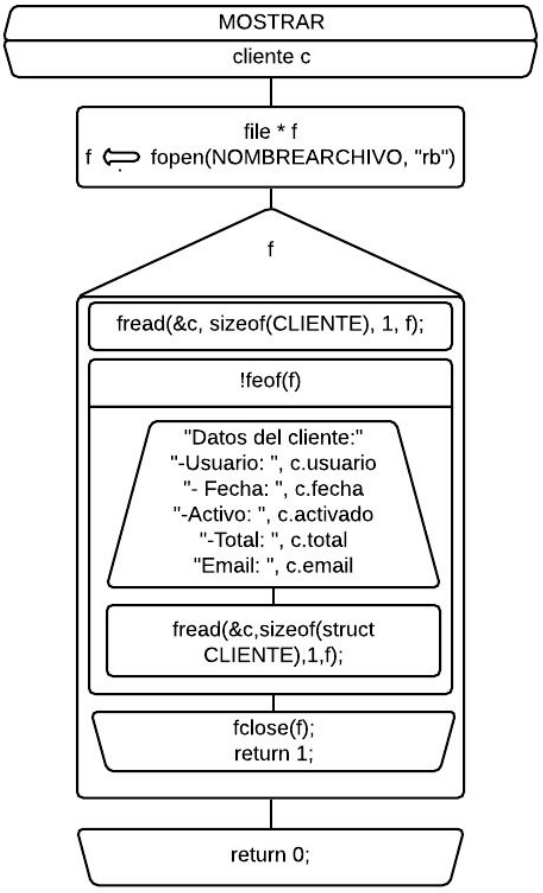
- Informe:
  - El trabajo fue realizado en conjunto por los tres integrantes del grupo.

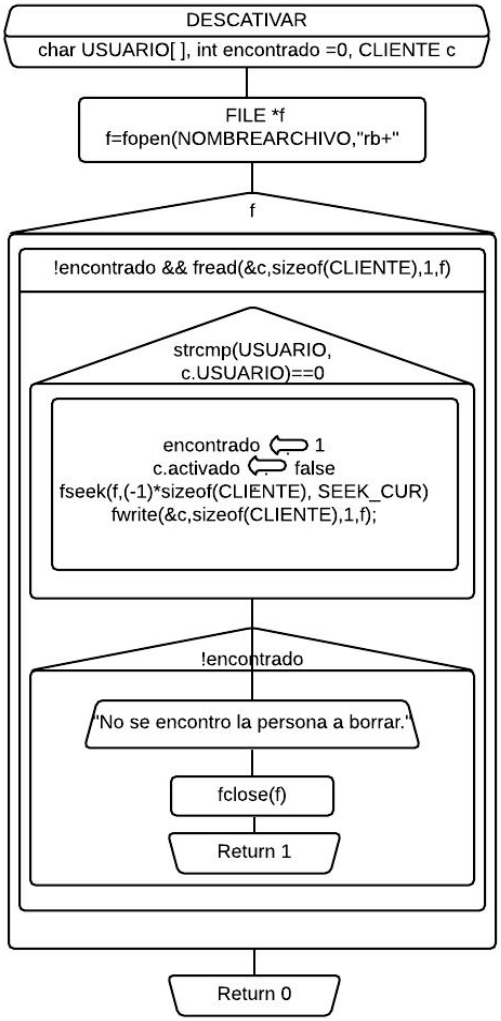
Diagramas de Lindsay:

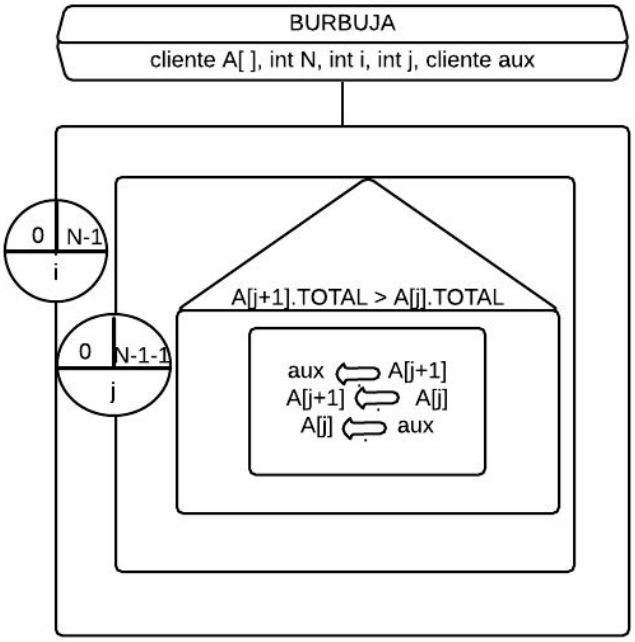


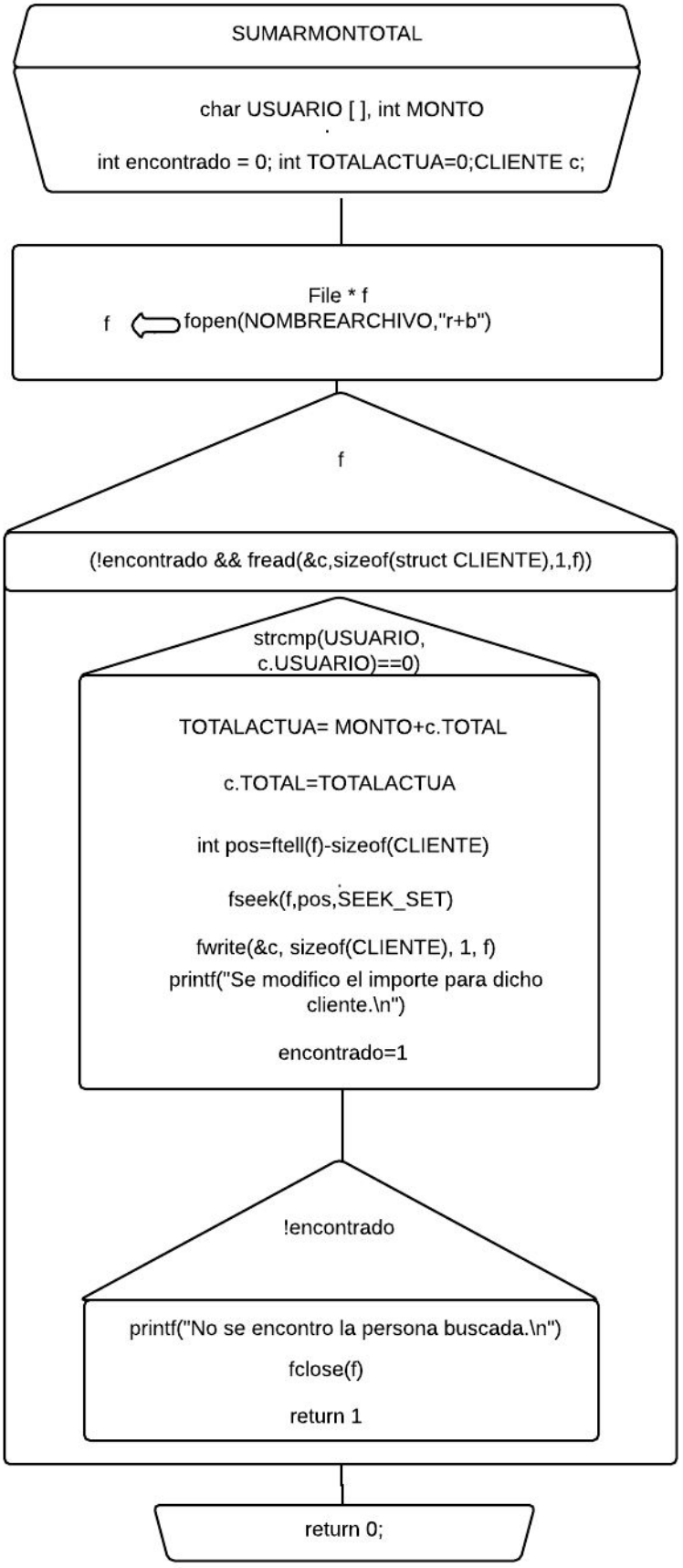


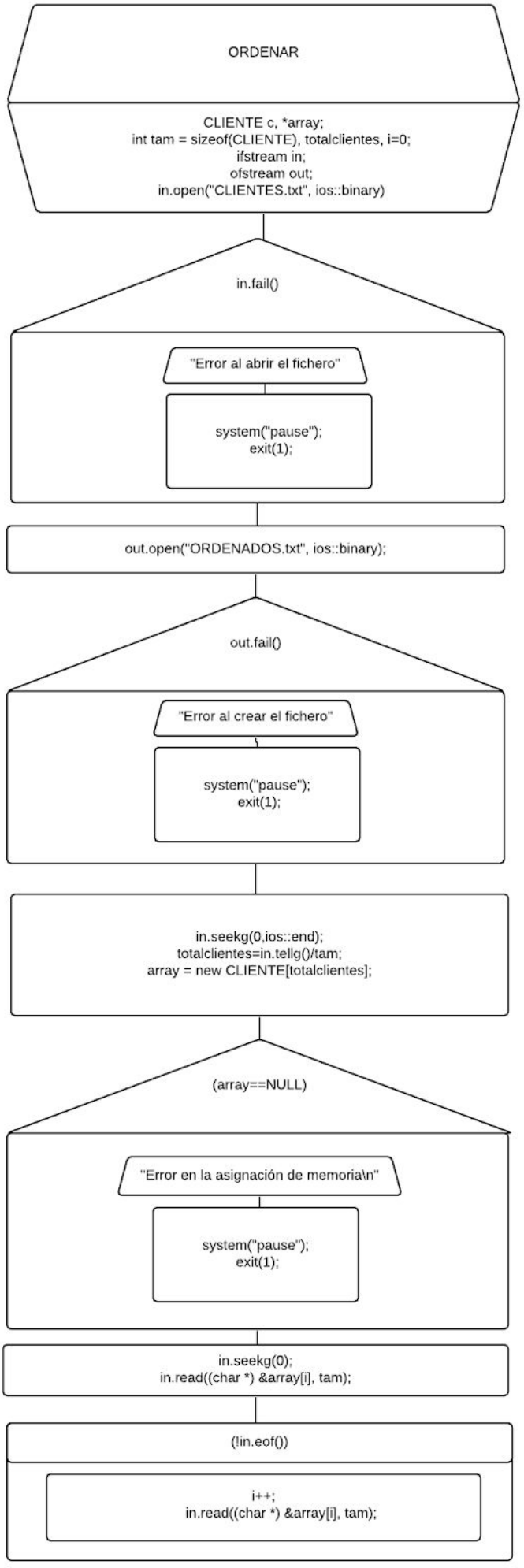
(Diagrama del ***menú*** adjuntado aparte)

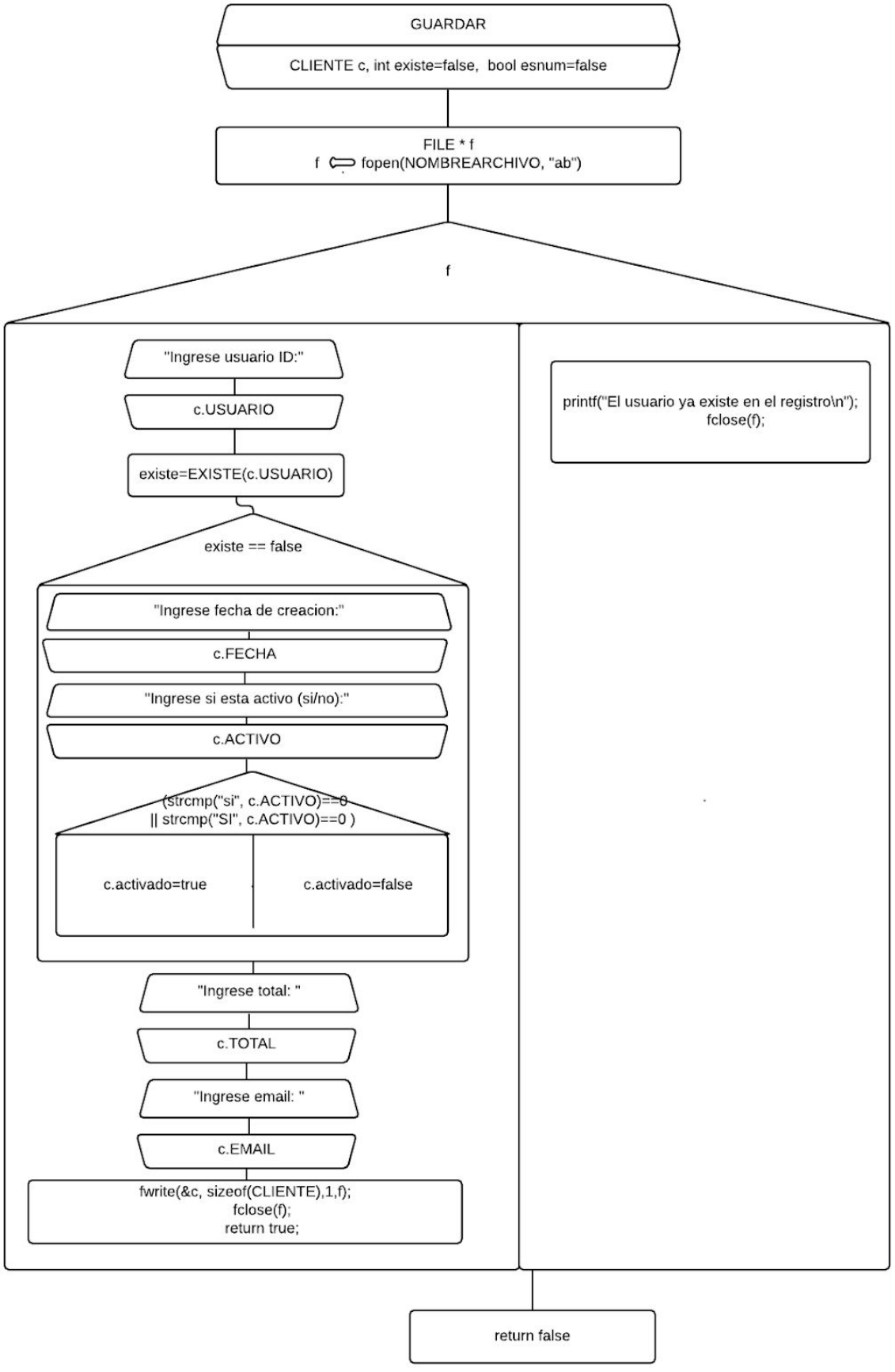

MOSTRAR
cliente c
file * f
f ⇦ fopen(NOMBREARCHIVO, "rb")
f
fread(&c, sizeof(CLIENTE), 1, f);
!feof(f)
"Datos del cliente:"
"-Usuario: ", c.usuario
"- Fecha: ", c.fecha
"-Activo: ", c.activado
"-Total: ", c.total
"Email: ", c.email
fread(&c,sizeof(struct CLIENTE),1,f);
fclose(f);
return 1;
return 0;

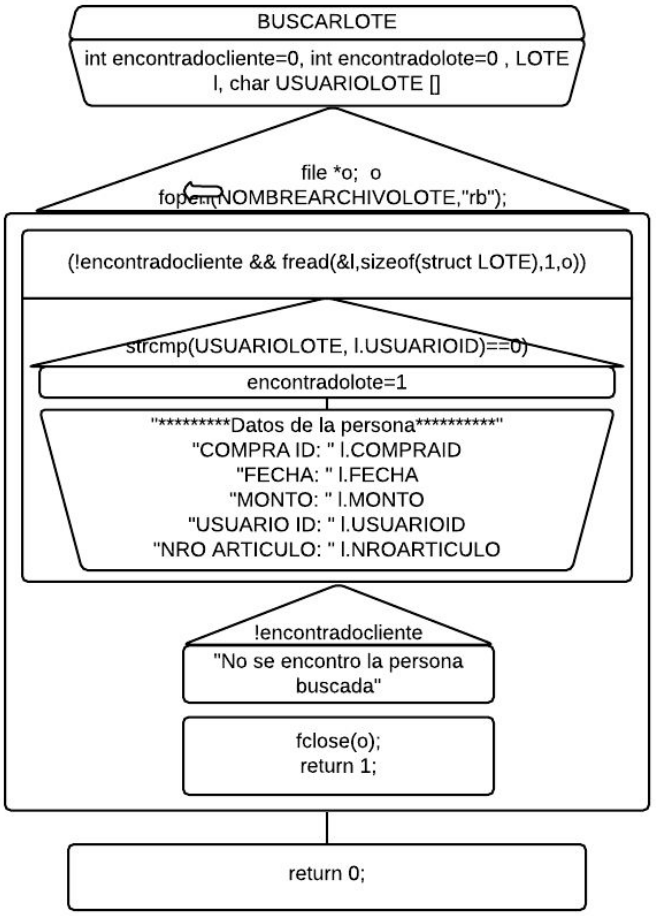

DESCATIVAR
char USUARIO[ ], int encontrado =0, CLIENTE c
FILE *f
f=fopen(NOMBREARCHIVO,"rb+"
f
!encontrado && fread(&c,sizeof(CLIENTE),1,f)
strcmp(USUARIO, c.USUARIO)==0
encontrado ⇦ 1
c.activado ⇦ false
fseek(f,(-1)*sizeof(CLIENTE), SEEK_CUR)
fwrite(&c,sizeof(CLIENTE),1,f);
!encontrado
"No se encontro la persona a borrar."
fclose(f)
Return 1
Return 0

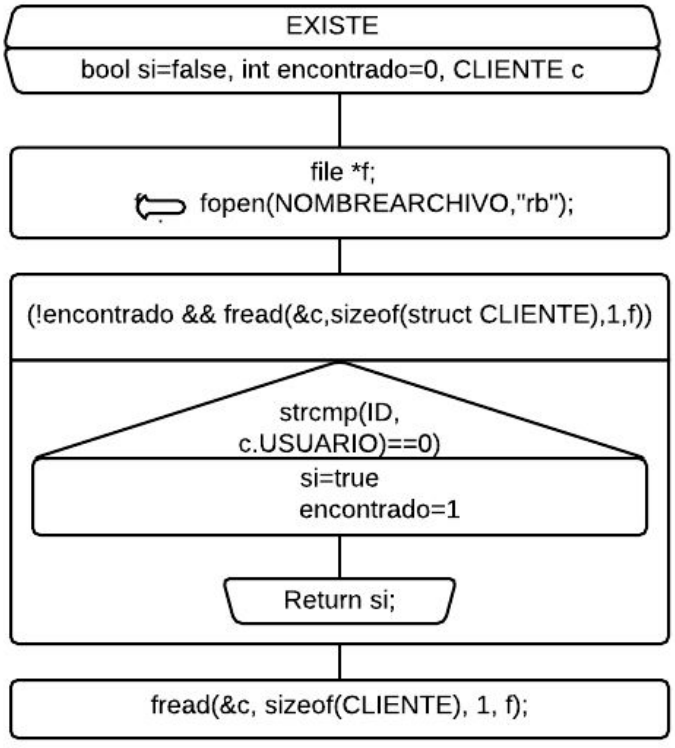

BURBUJA
cliente A[ ], int N, int i, int j, cliente aux
0
N-1
i
0
N-1-1
j
A[j+1].TOTAL > A[j].TOTAL
aux ⇦ A[j+1]
A[j+1] ⇦ A[j]
A[j] ⇦ aux

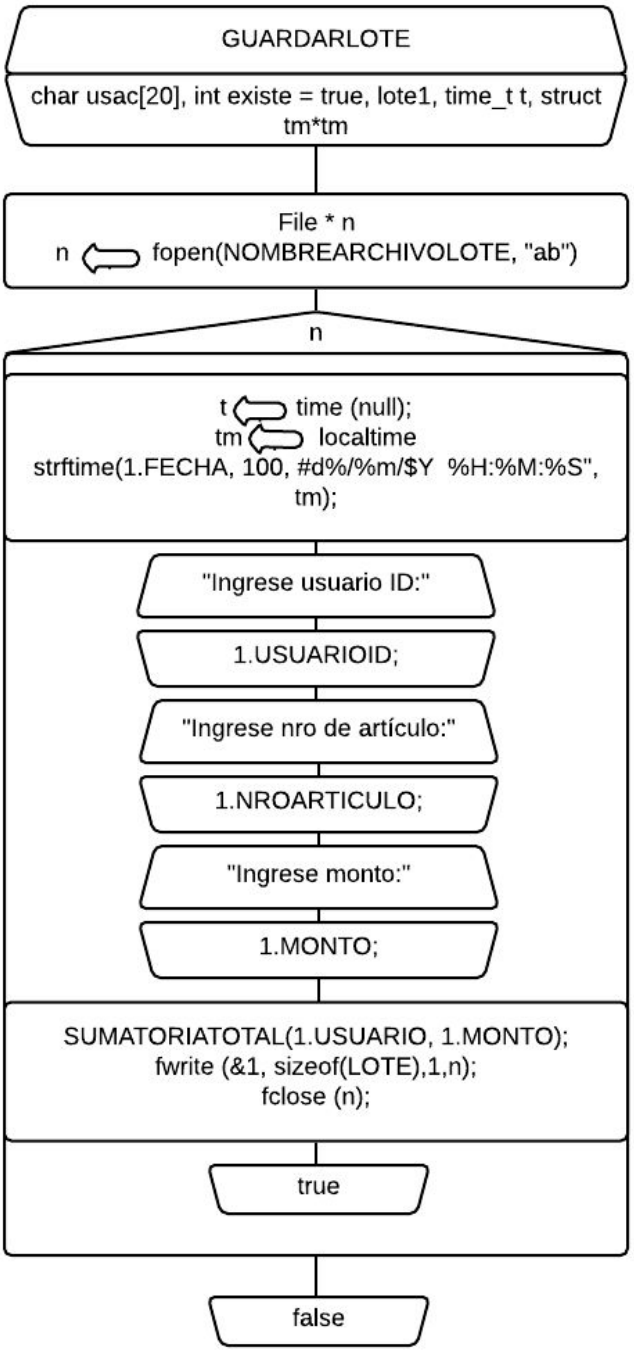

SUMARMONTOTAL
char USUARIO [ ], int MONTO
int encontrado = 0; int TOTALACTUA=0;CLIENTE c;
File * f
f
fopen(NOMBREARCHIVO,"r+b")
f
(!encontrado && fread(&c,sizeof(struct CLIENTE),1,f))
strcmp(USUARIO,
c.USUARIO)==0)
TOTALACTUA= MONTO+c.TOTAL
c.TOTAL=TOTALACTUA
int pos=ftell(f)-sizeof(CLIENTE)
fseek(f,pos,SEEK_SET)
fwrite(&c, sizeof(CLIENTE), 1, f)
printf("Se modifico el importe para dicho
cliente.\n")
encontrado=1
!encontrado
printf("No se encontro la persona buscada.\n")
fclose(f)
return 1
return 0;

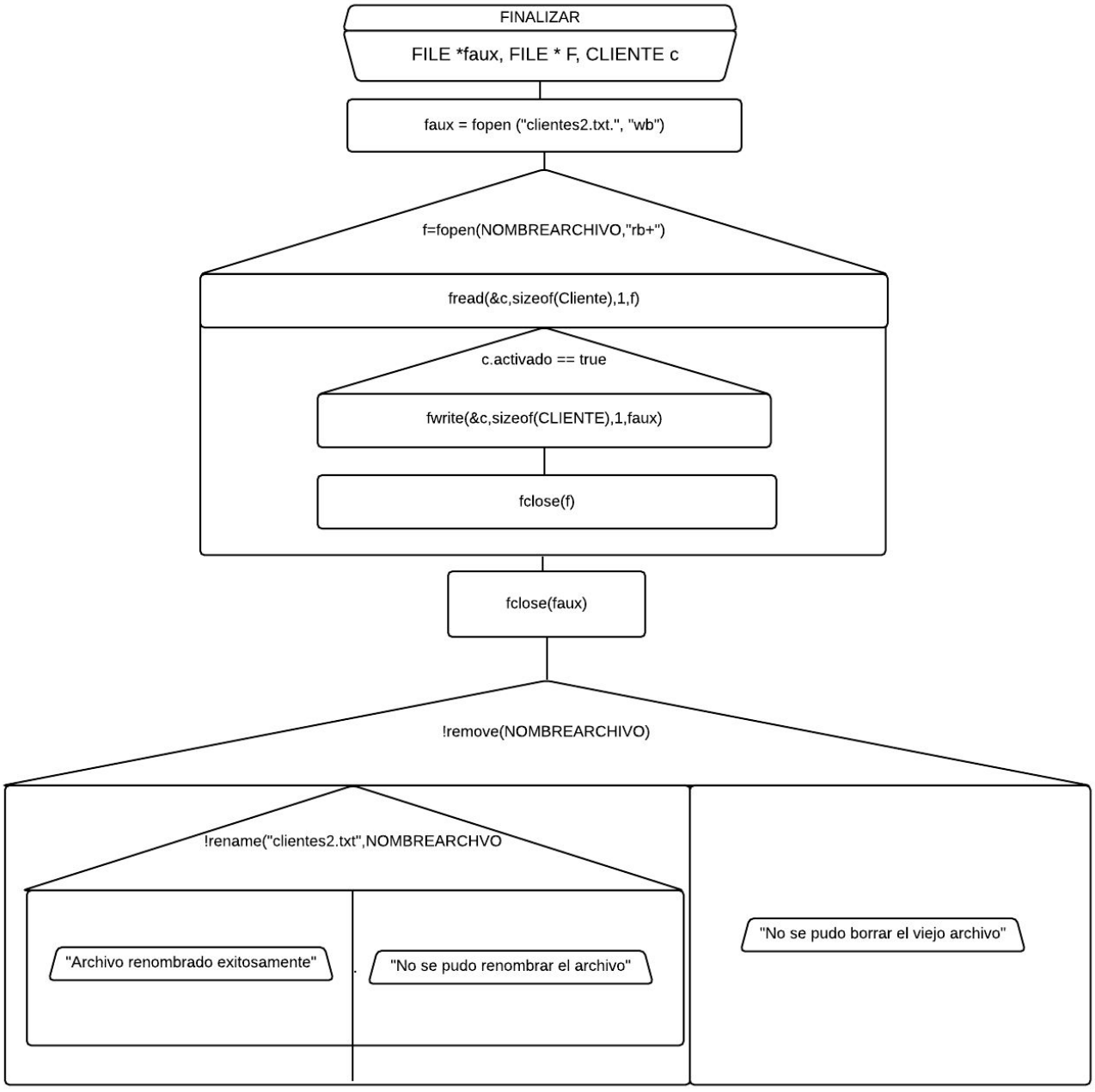

ORDENAR
CLIENTE c, *array;
int tam = sizeof(CLIENTE), totalclientes, i=0;
ifstream in;
ofstream out;
in.open("CLIENTES.txt", ios::binary)
in.fail()
"Error al abrir el fichero"
system("pause");
exit(1);
out.open("ORDENADOS.txt", ios::binary);
out.fail()
"Error al crear el fichero"
system("pause");
exit(1);
in.seekg(0,ios::end);
totalclientes=in.tellg()/tam;
array = new CLIENTE[totalclientes];
(array==NULL)
"Error en la asignación de memoria\n"
system("pause");
exit(1);
in.seekg(0);
in.read((char *) &array[i], tam);
(!in.eof())
i++;
in.read((char *) &array[i], tam);

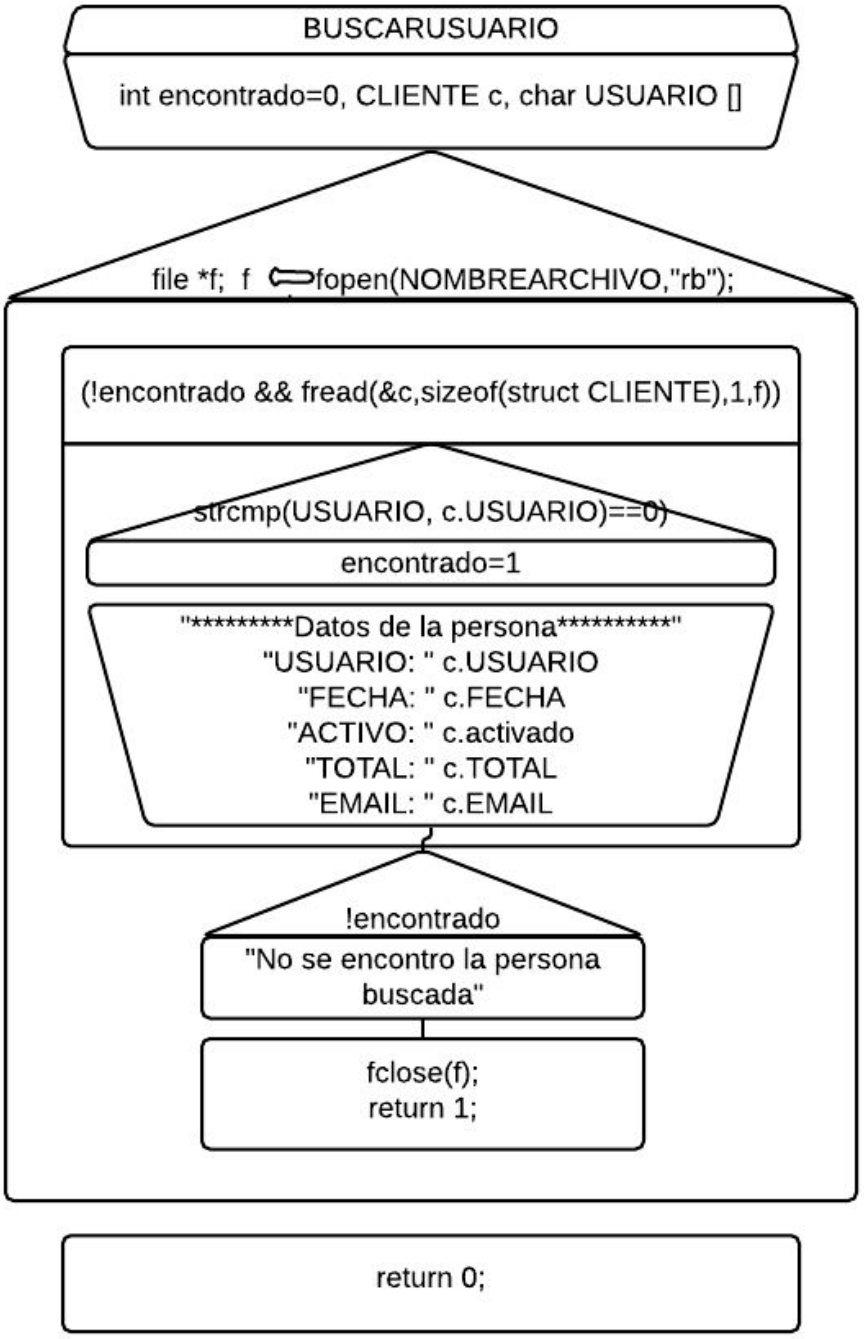

GUARDAR
CLIENTE c, int existe=false, bool esnum=false
FILE * f
f fopen(NOMBREARCHIVO, "ab")
f
"Ingrese usuario ID:"
c.USUARIO
existe=EXISTE(c.USUARIO)
existe == false
"Ingrese fecha de creacion:"
c.FECHA
"Ingrese si esta activo (si/no):"
c.ACTIVO
(strcmp("si", c.ACTIVO)==0
|| strcmp("SI", c.ACTIVO)==0)
c.activado=true
c.activado=false
"Ingrese total: "
c.TOTAL
"Ingrese email: "
c.EMAIL
fwrite(&c, sizeof(CLIENTE),1,f);
fclose(f);
return true;
printf("El usuario ya existe en el registro\n");
fclose(f);
return false

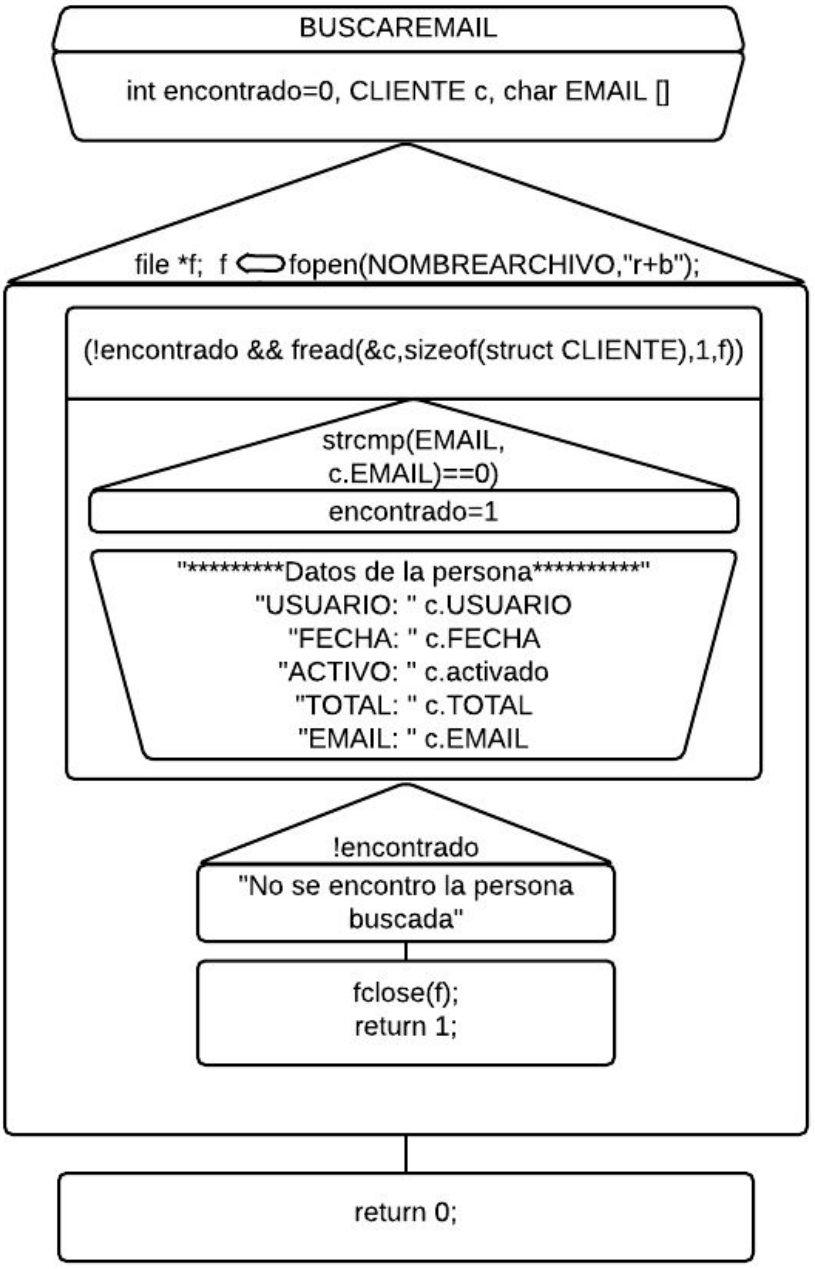

BUSCARLOTE
int encontradocliente=0, int encontradolote=0 , LOTE
l, char USUARIOLOTE []
file *o; o
fopen(NOMBREARCHIVOLOTE,"rb");
(!encontradocliente && fread(&l,sizeof(struct LOTE),1,o))
strcmp(USUARIOLOTE, l.USUARIOID)==0)
encontradolote=1
"*********Datos de la persona**********"
"COMPRA ID: " l.COMPRAID
"FECHA: " l.FECHA
"MONTO: " l.MONTO
"USUARIO ID: " l.USUARIOID
"NRO ARTICULO: " l.NROARTICULO
!encontradocliente
"No se encontro la persona buscada"
fclose(o);
return 1;
return 0;


EXISTE
bool si=false, int encontrado=0, CLIENTE c
file *f;
fopen(NOMBREARCHIVO,"rb");
(!encontrado && fread(&c,sizeof(struct CLIENTE),1,f))
strcmp(ID, c.USUARIO)==0)
si=true
encontrado=1
Return si;
fread(&c, sizeof(CLIENTE), 1, f);


GUARDARLOTE
char usac[20], int existe = true, lote1, time_t t, struct tm*tm
File * n
n fopen(NOMBREARCHIVOLOTE, "ab")
n
t time (null);
tm localtime
strftime(1.FECHA, 100, #d%/%m/$Y %H:%M:%S", tm);
"Ingrese usuario ID:"
1.USUARIOID;
"Ingrese nro de artículo:"
1.NROARTICULO;
"Ingrese monto:"
1.MONTO;
SUMATORIATOTAL(1.USUARIO, 1.MONTO);
fwrite (&1, sizeof(LOTE),1,n);
fclose (n);
true
false


FINALIZAR
FILE *faux, FILE * F, CLIENTE c
faux = fopen ("clientes2.txt.", "wb")
f=fopen(NOMBREARCHIVO,"rb+")
fread(&c,sizeof(Cliente),1,f)
c.activado == true
fwrite(&c,sizeof(CLIENTE),1,faux)
fclose(f)
fclose(faux)
!remove(NOMBREARCHIVO)
!rename("clientes2.txt",NOMBREARCHVO
"Archivo renombrado exitosamente"
"No se pudo renombrar el archivo"
"No se pudo borrar el viejo archivo"


BUSCARUSUARIO
int encontrado=0, CLIENTE c, char USUARIO []
file *f; f fopen(NOMBREARCHIVO,"rb");
(!encontrado && fread(&c,sizeof(struct CLIENTE),1,f))
strcmp(USUARIO, c.USUARIO)==0)
encontrado=1
"*********Datos de la persona**********"
"USUARIO: " c.USUARIO
"FECHA: " c.FECHA
"ACTIVO: " c.activado
"TOTAL: " c.TOTAL
"EMAIL: " c.EMAIL
!encontrado
"No se encontro la persona buscada"
fclose(f);
return 1;
return 0;


BUSCAREMAIL
int encontrado=0, CLIENTE c, char EMAIL []
file *f; f fopen(NOMBREARCHIVO,"r+b");
(!encontrado && fread(&c,sizeof(struct CLIENTE),1,f))
strcmp(EMAIL, c.EMAIL)==0)
encontrado=1
"*********Datos de la persona**********"
"USUARIO: " c.USUARIO
"FECHA: " c.FECHA
"ACTIVO: " c.activado
"TOTAL: " c.TOTAL
"EMAIL: " c.EMAIL
!encontrado
"No se encontro la persona buscada"
fclose(f);
return 1;
return 0;