Elementos de Programación

UNIDAD 7. ARRAYS

Anexo Ejercicios Resueltos

Una empresa tiene 10 vendedores identificados con un código numérico que va del 101 al 110. La empresa comercializa productos que son codificados con números de 4 cifras. No se sabe la cantidad exacta de productos que vende la empresa, pero sí se sabe que no son más de 30.

Se desea realizar un programa que permita:

1. Ingresar los códigos de producto que comercializa la empresa junto con su precio. La carga finaliza con un código igual a 10000.
2. Ingresar las ventas realizadas. Para ello por cada factura se ingresa:
    a. Código de producto
    b. Código de vendedor
    c. Cantidad de unidades vendidas

    Las ventas finalizan con un código de vendedor igual a 0.
3. Al finalizar mostrar el detalle de los importes vendidos por cada vendedor de cada uno de los productos.
4. Mostrar el o los vendedores que realizaron un mayor importe de ventas
5. Indicar aquellos productos que no fueron vendidos por ningún vendedor.

Para encarar un ejercicio de este tipo primero se debe analizar y planificar una estrategia de resolución. El punto 1 nos pide ingresar los códigos de los productos y el precio. Nos dice que no se sabe la cantidad de datos, pero sí que no son más de 30, por lo tanto, al reservar memoria lo haremos por el máximo posible es decir de 30. Como tenemos que almacenar 2 datos, el código y el precio vamos a necesitar dos vectores que estén en paralelo, es decir que en la posición 0 del vector de código vamos a almacenar el código del producto y al mismo tiempo en la posición 0 del vector de precios vamos a almacenar el precio de dicho producto. Como no sabemos la cantidad exacta de datos la función de carga DEBE retornar la cantidad de datos ingresados para luego poder usar ese dato en el resto de las funciones.

En el punto 2 se procesan las ventas, para ello se ingresa el código de producto y el código de vendedor. Como se debe almacenar el detalle de importes de ventas de cada vendedor por cada producto, vamos a necesitar una matriz. Debido a que dicha matriz guarda importes va a ser una matriz de float. Dicha matriz va a tener una columna por cada vendedor (son 10 en total) y una fila por cada producto (máximo 30) entonces se debe declarar como de 30 x 10 por más que luego tengamos filas que no se utilicen. Dicha matriz como va a acumular debe estar inicializada en 0.

La información de las facturas nos permitirá posicionarnos en matriz para acumular el dato:

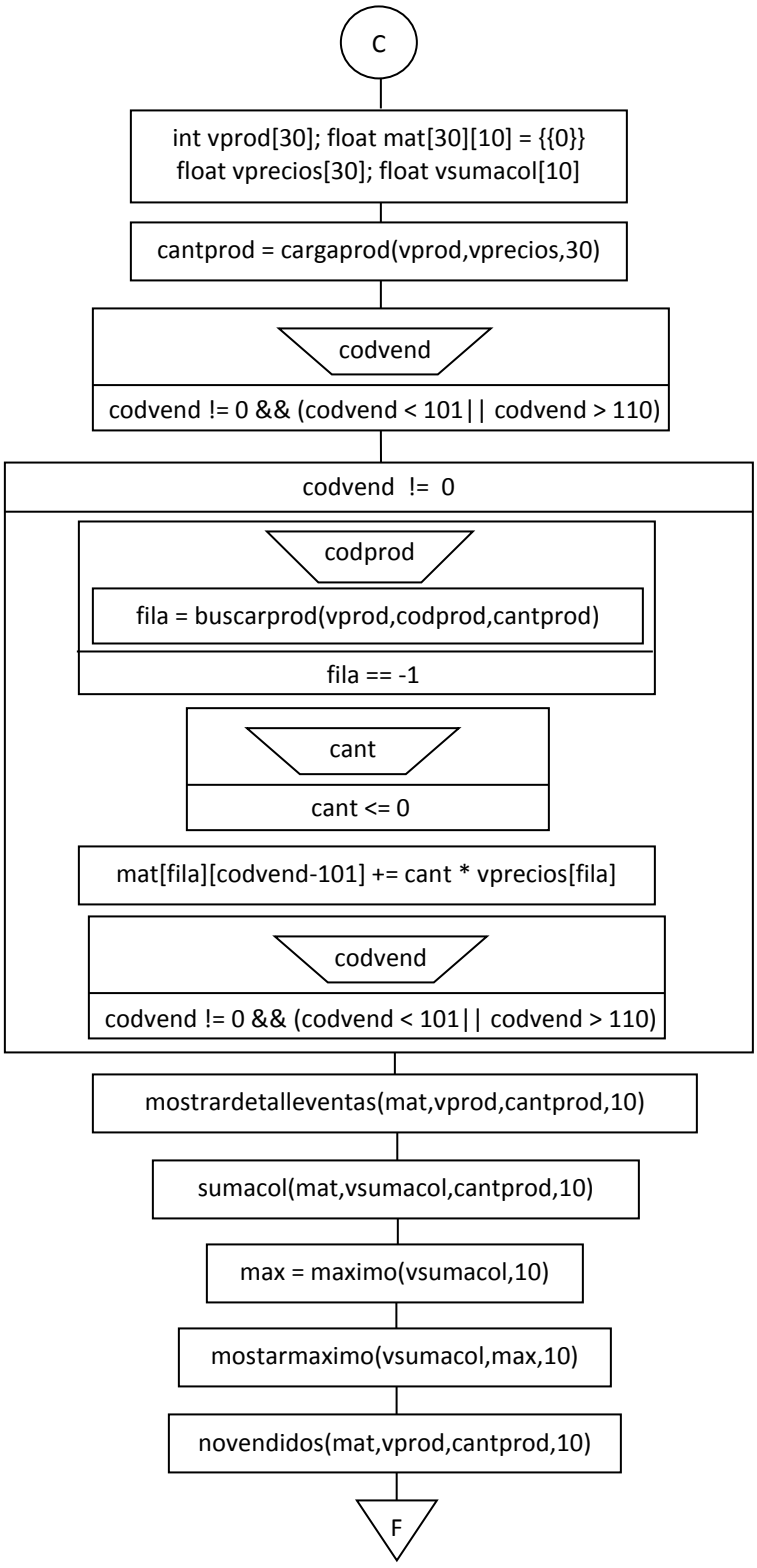
- La columna será el código de vendedor que como son correlativos de 101 a 110 y nuestra matriz comienza en 0 entonces tendremos que restarle 101 al código ingresado para posicionarnos en la columna correspondiente.
- La fila la da el código de producto, pero como los códigos se ingresaron previamente por teclado para saber a qué fila de la matriz corresponde, se debe realizar una búsqueda sobre el vector de códigos y recuperar la posición donde se encuentra dicho código. Esa posición nos indica la fila de la matriz
- El dato para acumular no es la cantidad sino el importe, por lo que se debe recurrir al vector de precios con el subíndice de donde encontramos el producto. Dicho precio se multiplica por la cantidad ingresada y ese importe se acumula en la matriz.

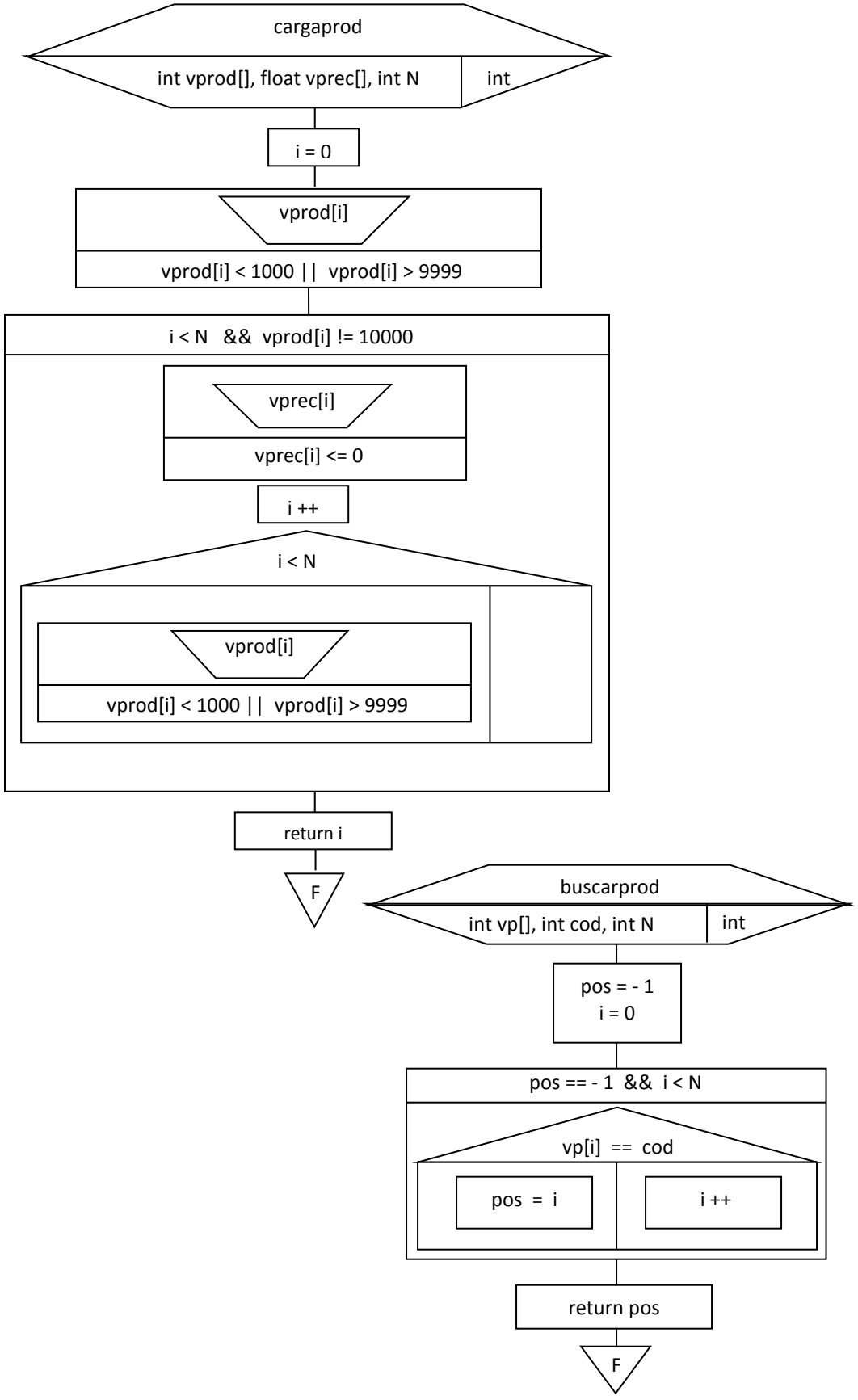
Una vez finalizado el ingreso de facturas (cuando se ingresa un vendedor con código igual a 0) se comienzan a resolver los informes que nos solicitan.

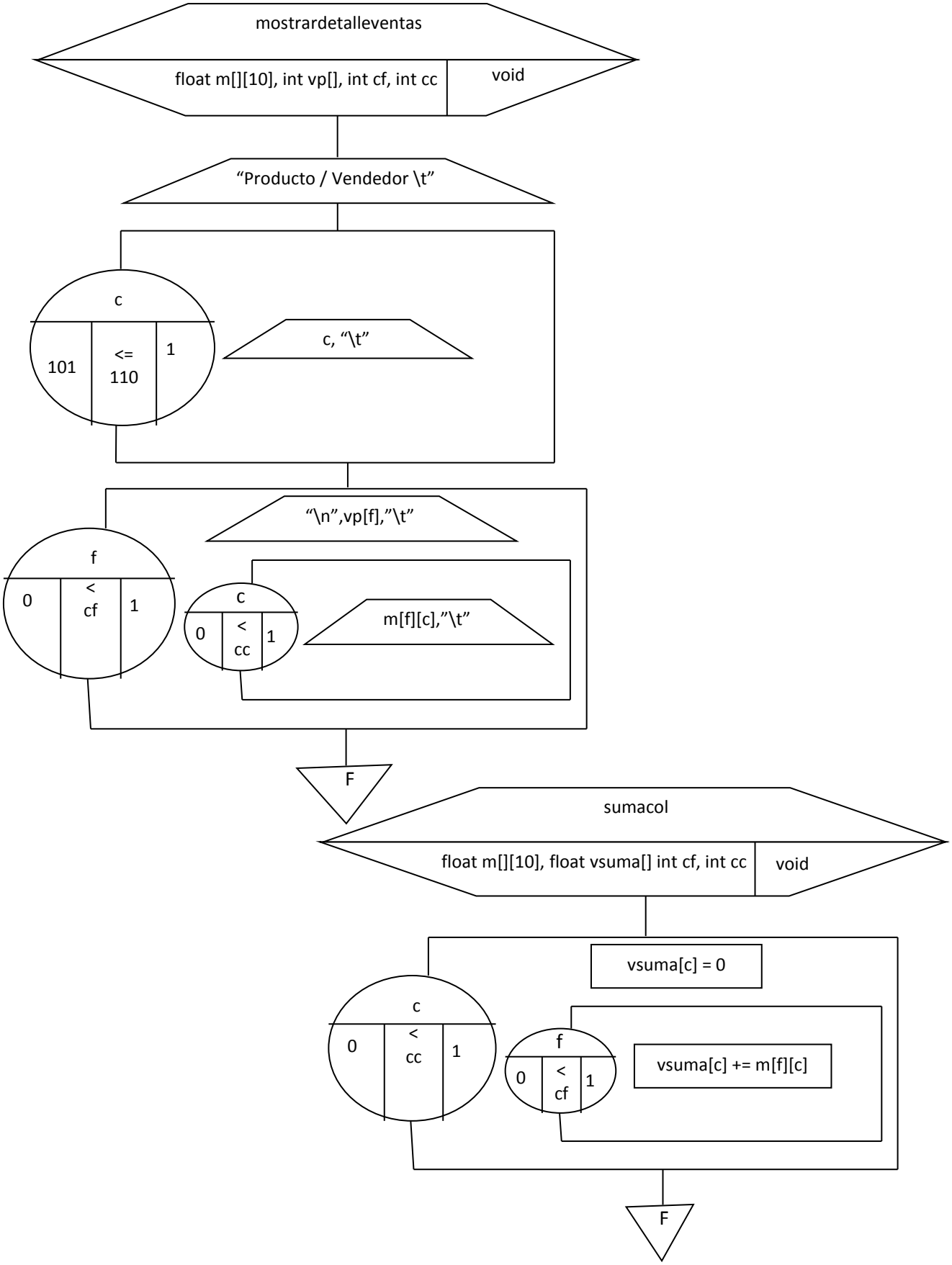
El punto 3 nos pide el detalle ventas, mostrando importe vendido por cada vendedor de cada producto por lo que se debe mostrar toda la matriz encolumnada y con los títulos de filas y columnas correspondientes.

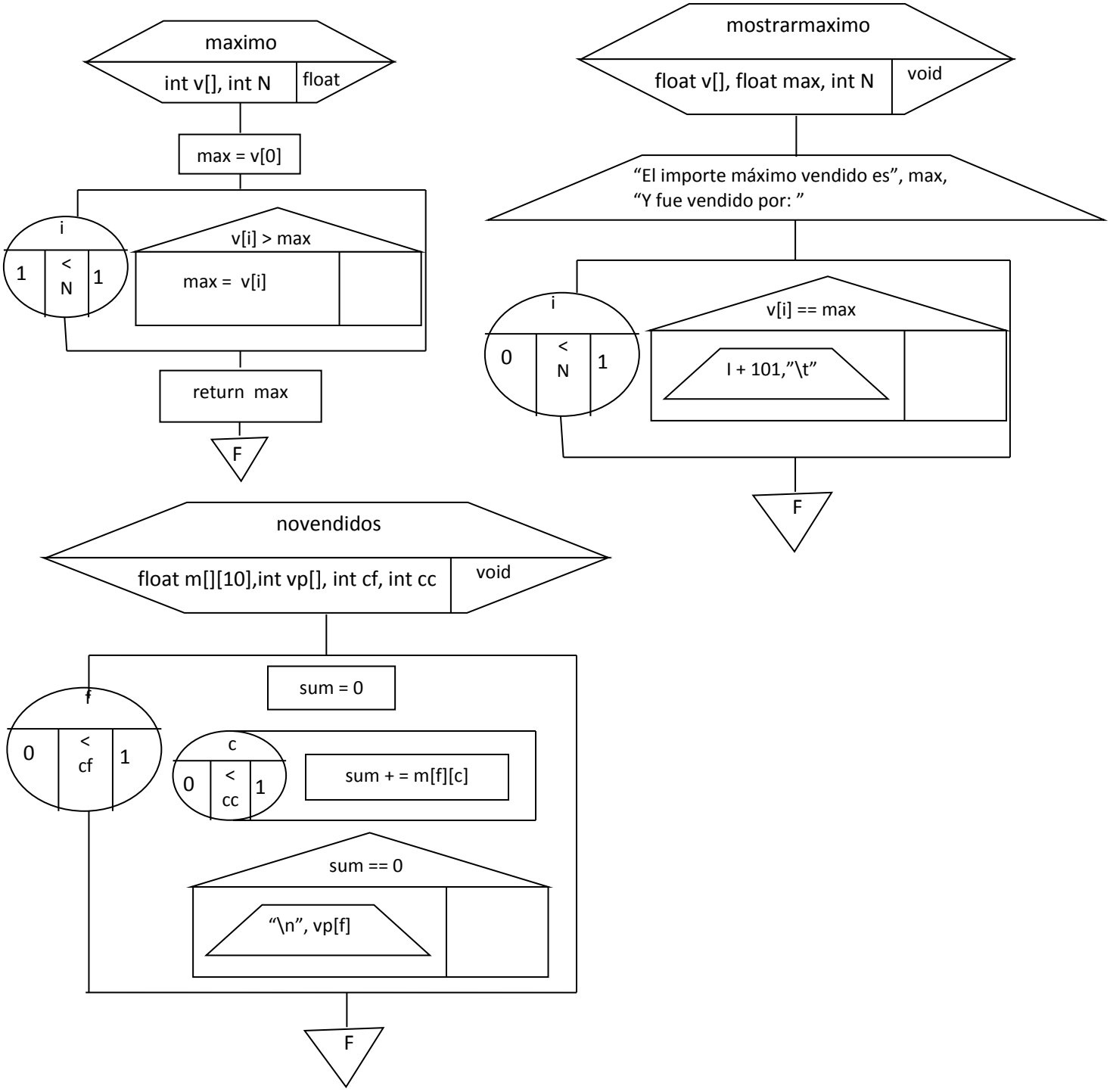
El punto 4 solicita mostrar el o los vendedores que realizaron un mayor importe de ventas. En la matriz la información de los importes de venta de cada vendedor esta detallada por producto, pero en este punto nos pide el total por lo que será necesario sumar todos los importes vendidos para cada vendedor. Debemos realizar una suma por columna almacenando los resultados en un vector. Luego sobre ese vector se debe calcular el importe máximo y por último volver a recorrer ese vector para ver si ese importe máximo se repite o no en varios vendedores.

El punto 5 pide indicar aquellos productos que no fueron vendidos por ningún vendedor. La información de las ventas de cada producto se puede obtener recorriendo por fila la matriz, si sumamos toda una fila y esa suma es igual a 0, significa que no se vendió dicho producto.


C
int vprod[30]; float mat[30][10] = {{0}}
float vprecios[30]; float vsumacol[10]
cantprod = cargaprod(vprod,vprecios,30)
codvend
codvend != 0 && (codvend < 101|| codvend > 110)
codvend != 0
codprod
fila = buscarprod(vprod,codprod,cantprod)
fila == -1
cant
cant <= 0
mat[fila][codvend-101] += cant * vprecios[fila]
codvend
codvend != 0 && (codvend < 101|| codvend > 110)
mostrardetalleventas(mat,vprod,cantprod,10)
sumacol(mat,vsumacol,cantprod,10)
max = maximo(vsumacol,10)
mostarmaximo(vsumacol,max,10)
novendidos(mat,vprod,cantprod,10)
F


cargaprod
int vprod[], float vprec[], int N
int
i = 0
vprod[i]
vprod[i] < 1000 || vprod[i] > 9999
i < N && vprod[i] != 10000
vprec[i]
vprec[i] <= 0
i ++
i < N
vprod[i]
vprod[i] < 1000 || vprod[i] > 9999
return i
F
buscarprod
int vp[], int cod, int N
int
pos = - 1
i = 0
pos == - 1 && i < N
vp[i] == cod
pos = i
i ++
return pos
F


mostrardetalleventas
float m[][10], int vp[], int cf, int cc
void
“Producto / Vendedor \t”
c
101
<=
110
1
c, “\t”
“\n”,vp[f],”\t”
f
0
<
cf
1
c
0
<
cc
1
m[f][c],”\t”
F
sumacol
float m[][10], float vsuma[] int cf, int cc
void
vsuma[c] = 0
c
0
<
cc
1
f
0
<
cf
1
vsuma[c] += m[f][c]
F


maximo
int v[], int N
float
max = v[0]
i
1
<
N
1
v[i] > max
max = v[i]
return max
F
mostrarmaximo
float v[], float max, int N
void
“El importe máximo vendido es”, max,
“Y fue vendido por: ”
i
0
<
N
1
v[i] == max
I + 101,”\t”
F
novendidos
float m[][10],int vp[], int cf, int cc
void
sum = 0
f
0
<
cf
1
c
0
<
cc
1
sum + = m[f][c]
sum == 0
“\n”, vp[f]
F

```
#include <stdio.h>
int CargaProd (int[], float[], int);
int BuscarProd (int[], int, int);
void MostrarDetalleVtas (float[][10], int [], int, int);
void SumaCol (float[][10], float[], int, int);
float Maximo (float[], int);
void MostrarMaximo (float[], float, int);
void NoVendidos(float[][10], int[], int, int);

int main()
{
    int vprod[30], cantprod, codvend, codprod, fila, cant;
    float mat[30][10]={{0}}, vprecios[30], vsumacol[10], max;
    cantprod = CargaProd(vprod, vprecios, 30);
    do
    {
        printf("Ingrese el codigo del vendedor (entre 101 y 110, 0 para terminar):");
        scanf("%d",&codvend);
    }while(codvend != 0 && (codvend<101||codvend>110));

    while(codvend!=0)
    {
        do
        {
            printf("Ingrese el codigo de producto:");
            scanf("%d", &codprod);
            fila = BuscarProd(vprod,codprod, cantprod);
        }while(fila == -1);

        do
        {
            printf("Ingrese la cantidad vendida:");
            scanf("%d", &cant);
        }while (cant<=0);

        mat[fila][codvend-101]+=cant*vprecios[fila];

        do
        {
            printf("Ingrese el codigo del vendedor (entre 101 y 110, 0 para terminar):");
            scanf("%d",&codvend);
        }while(codvend != 0 && (codvend<101||codvend>110));
    }
    MostrarDetalleVtas(mat,vprod,cantprod,10);
    SumaCol(mat,vsumacol, cantprod,10);
    max=Maximo(vsumacol,10);
    MostrarMaximo(vsumacol, max,10);
    NoVendidos(mat,vprod,cantprod,10);
    return 0;
}
int CargaProd (int vprod[], float vprec[], int n)
{
    int i=0;
    do
    {
        printf("Ingrese el codigo de producto (4 cifras) :");
        scanf("%d", &vprod[i]);
    }while(vprod[i]<1000||vprod[i]>9999); //con 9999 obligo a ingresar al menos un producto

    while (i<n && vprod[i]!=10000)
    {
        do
        {
            printf("Ingrese el precio para el producto %d:",vprod[i]);
            scanf("%f", &vprec[i]);
        }while(vprec[i]<=0);

        i++;
        if (i<n)
            do
            {
                printf("Ingrese el codigo de producto (10000 para terminar): ");
```

```
                scanf("%d", &vprod[i]);
            }while(vprod[i]<1000||vprod[i]>10000);
    }
    return i;
}
int BuscarProd (int vp[], int cod, int n)
{
    int pos =-1, i=0;
    while (pos==-1 && i<n)
    {
        if (vp[i]==cod)
            pos=i;
        else
            i++;
    }
    return pos;
}
void MostrarDetalleVtas (float m[][10], int vp[], int cf, int cc)
{
    int c,f;
    printf("producto/vendedor");
    for (c=101;c<=110;c++)
        printf ("%6d", c);
    for (f=0;f<cf;f++)
    {
        printf("\n%17d",vp[f]);
        for (c=0;c<cc;c++)
            printf("%8.2f",m[f][c]);
    }
}
void SumaCol (float m[][10], float vsuma[], int cf, int cc)
{
    int f,c;
    for (c=0;c<cc;c++)
    {
        vsuma[c]=0;
        for(f=0;f<cf;f++)
            vsuma[c]+=m[f][c];
    }
}
float Maximo (float v[] , int n)
{
    float max = v[0];
    int i;
    for (i=1;i<n;i++)
        if (v[i]>max)
            max =v[i];
    return max;
}
void MostrarMaximo (float v[], float max, int n)
{
    int i;
    printf ("\nEl importe maximo vendido es: %5.2f y fue vendido por: ", max);
    for (i=0;i<n;i++)
        if (v[i]==max)
            printf ("%4d", i+101);
}
void NoVendidos(float m[][10], int vp[], int cf, int cc)
{
    int f,c;
    float sum;
    for (f=0;f<cf;f++)
    {
        sum=0;
        for (c=0;c<cc;c++)
            sum+=m[f][c];
        if (sum ==0)
            printf ("\nEl producto %d no fue vendido",vp[f]);
    }
}
```

Elementos de Programación

UNIDAD 10. Archivos - Anexo Ejercicios Resueltos

INDICE

UNIDAD 10 – Archivos: Ejercicios Resueltos

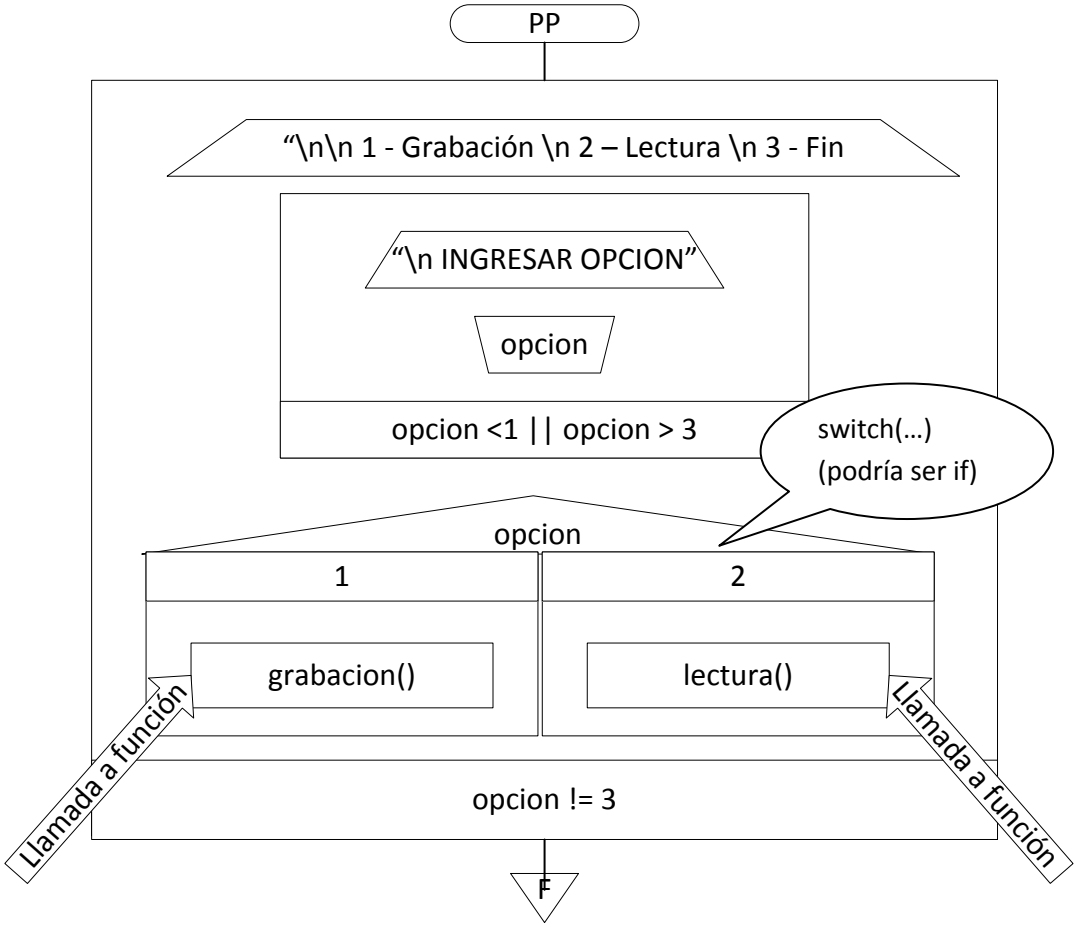
1. Notas de alumnos

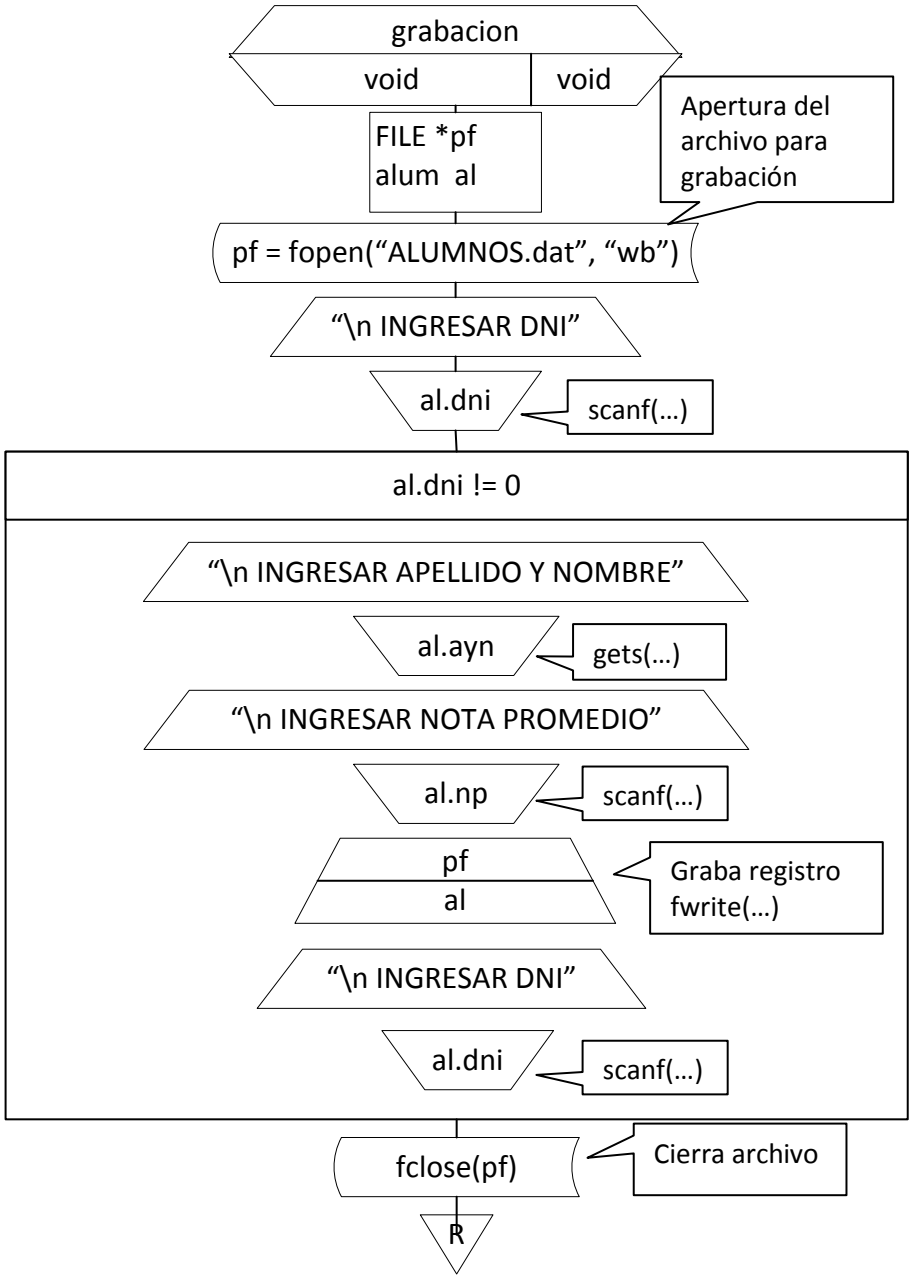
Se ingresa DNI, apellido y nombre y nota promedio de alumnos de un curso. Dicha información termina con DNI igual a cero.

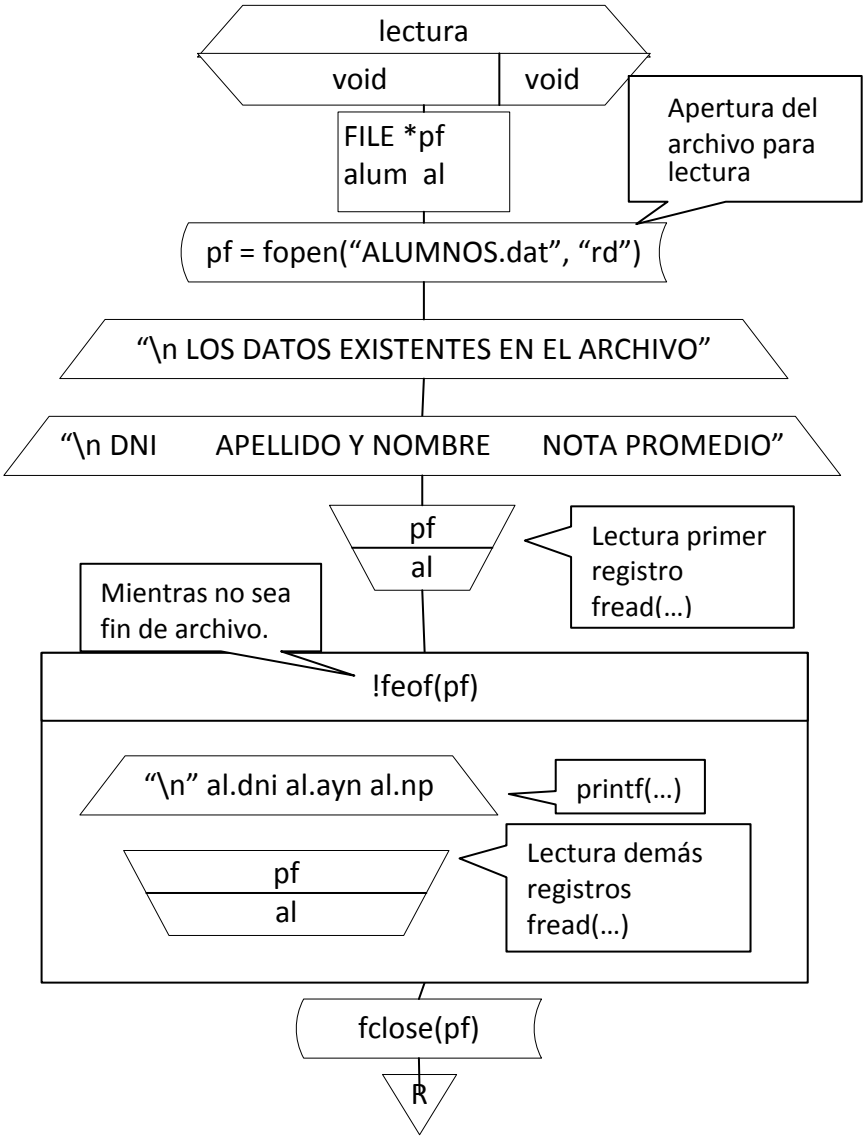
Se pide:

a) Grabar toda la información en el archivo **alumno.dat**

b) Leer la información grabada en el archivo.




grabacion
void
void
FILE *pf
alum al
Apertura del archivo para grabación
pf = fopen("ALUMNOS.dat", "wb")
"\n INGRESAR DNI"
al.dni
scanf(...)
al.dni != 0
"\n INGRESAR APELLIDO Y NOMBRE"
al.ayn
gets(...)
"\n INGRESAR NOTA PROMEDIO"
al.np
scanf(...)
pf
al
Graba registro fwrite(...)
"\n INGRESAR DNI"
al.dni
scanf(...)
fclose(pf)
Cierra archivo
R




```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

////////////////////// DECLARACION DEL TIPO DE DATO ////////////////////
typedef struct
{
   int dni;
   char ayn[51];
   float np;
}alum;

////////////////////// DECLARACION DE FUNCIONES ////////////////////////
void GRABACION (void);
void LECTURA (void);

////////////////////// DEFINICION DEL PROGRAMA PRINCIPAL///////////////
```

```
void main (void)
{
  int opcion;
  do{
      printf ("\n\n\n 1- GRABACION \n 2- LECTURA \n 3- FIN ");
      do{
         printf ("\n INGRESAR UNA OPCION: ");
         scanf ("%d", &opcion);
      }while( opcion < 1 || opcion > 3);

      switch(opcion)
      {
           case 1: GRABACION ();
                    break;
           case 2: LECTURA ();
      }
 }while (opcion !=3);
}

////////////////////////// FUNCION GRABACION ////////////////////
void GRABACION(void)
{
FILE *pf;
alum al;
pf = fopen ("ALUMNO.DAT","wb");  // se abre el archivo binario para grabacion
if (pf ==NULL)    // se verifica si se puede generar
 {
   printf("\n NO SE PUEDE ACCEDER AL ARCHIVO ");
   getch();
   exit(1);
 }
printf ("\n INGRESAR DNI : ");
scanf ("%d",&al.dni);
while (al.dni)
{
  fflush(stdin);
  printf ("\n INGRESAR APELLIDO Y NOMBRE : ");
  gets(al.ayn);
  printf ("\n INGRESAR NOTA PROMEDIO : ");
  scanf ("%f",&al.np);
  fwrite (&al,sizeof(alum),1,pf); //se graba en el archivo la informacion
ingresada en memoria
  printf ("\n INGRESAR DNI : ");
  scanf ("%d",&al.dni);
}
fclose (pf);   // se cierra el archivo
}


////////////////////////// FUNCION LECTURA ////////////////////
void LECTURA(void)
{
FILE *pf;
alum al;
pf = fopen ("ALUMNO.DAT","rb");         // se abre el archivo binario para lectura
if (pf ==NULL)                          // se verifica si existe el archivo
 {
   printf("\n NO SE PUEDE ACCEDER AL ARCHIVO ");
   getch();
   exit(1);
 }
printf("\n LOS DATOS EXISTENTES EN EL ARCHIVO SON ");
printf (" \n\n    DNI                APELLIDO Y NOMBRE                    NOTA
PROMEDIO ");
fread (&al,sizeof(alum),1,pf);  // se lee del archivo la informacion grabada
while (! feof(pf)) // se lee mientras exista informacion (registros) en el archivo
{
  printf ("\n %8d          %-40s %5.2f", al.dni, al.ayn, al.np); // muestra
  fread (&al,sizeof(alum),1,pf);  // se lee del archivo la informacion grabada
}
```

```
fclose (pf);   // se cierra el archivo
}
```

2. Saldos

La empresa Disco Mundo dispone del archivo secuencial **SALDOS.dat** conteniendo los siguientes datos de casi 1200 clientes.

- Nro. de Cliente (entero - entre 100 y 11900)
- Razón Social (25 caracteres)
- Importe Total Facturado (real)
- Total de Pagos Efectuados (real)
- Saldo Deudor (real)
- Código de Estado ('A' – es Activo, 'B' – es Baja)

Nos suministran, además, todas las operaciones de ventas y cobranzas del día que efectuó la empresa que no están ordenadas. Para cada operación se informa:

- Nro. de Cliente (entero - entre 100 y 11900)
- Zona (1 a 100)
- Código de Operación ('V' – Venta, 'P' - Pago)
- Importe (> 0 y < 20.000)

Las operaciones finalizan con una operación ficticia que tiene Nro. de Cliente CERO:

Confeccionar el diagrama de lógica y la respectiva codificación para:

a) Generar el array de estructuras llamado **Clie_Activos**, con solamente los datos de los clientes activos (Código de Estado 'A') del archivo SALDOS. Utilizar función **LECTURA_CLIENTE**.

b) Ingresar y controlar las operaciones del día. Para controlar la validez se debe confeccionar una función **CONTROL** que pueda validar los 4 datos que se ingresaron para cada operación respondiendo si son o no correctos. Utilizarla para aceptar o rechazar cada operación. Cuando la operación se rechaza, grabar todos los datos de la operación en un archivo **ERROR.dat** y pasar al ingreso de otra operación.
Utilizar función **ENTRADA** para verificar la existencia del Cliente, si no existe indicar mediante un mensaje "CLIENTE INEXISTENTE ", pasando a ingresar otra operación.

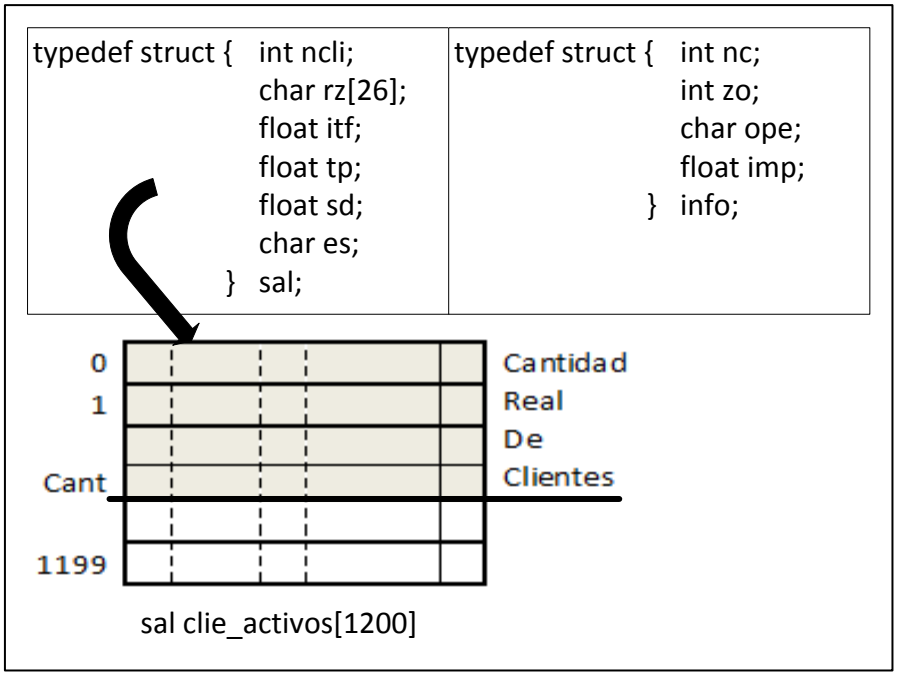
c) Actualizar el array **Clie_Activos** con las operaciones exitosas.
Con las operaciones de Ventas actualizar los campos Importe Total Facturado (sumando) y Saldo Deudor (Importe Total Facturado actualizado - Total de Pagos Efectuados)
Con las operaciones de Pagos actualizar Total de Pagos Efectuados (sumando) y calcular el nuevo Saldo Deudor.

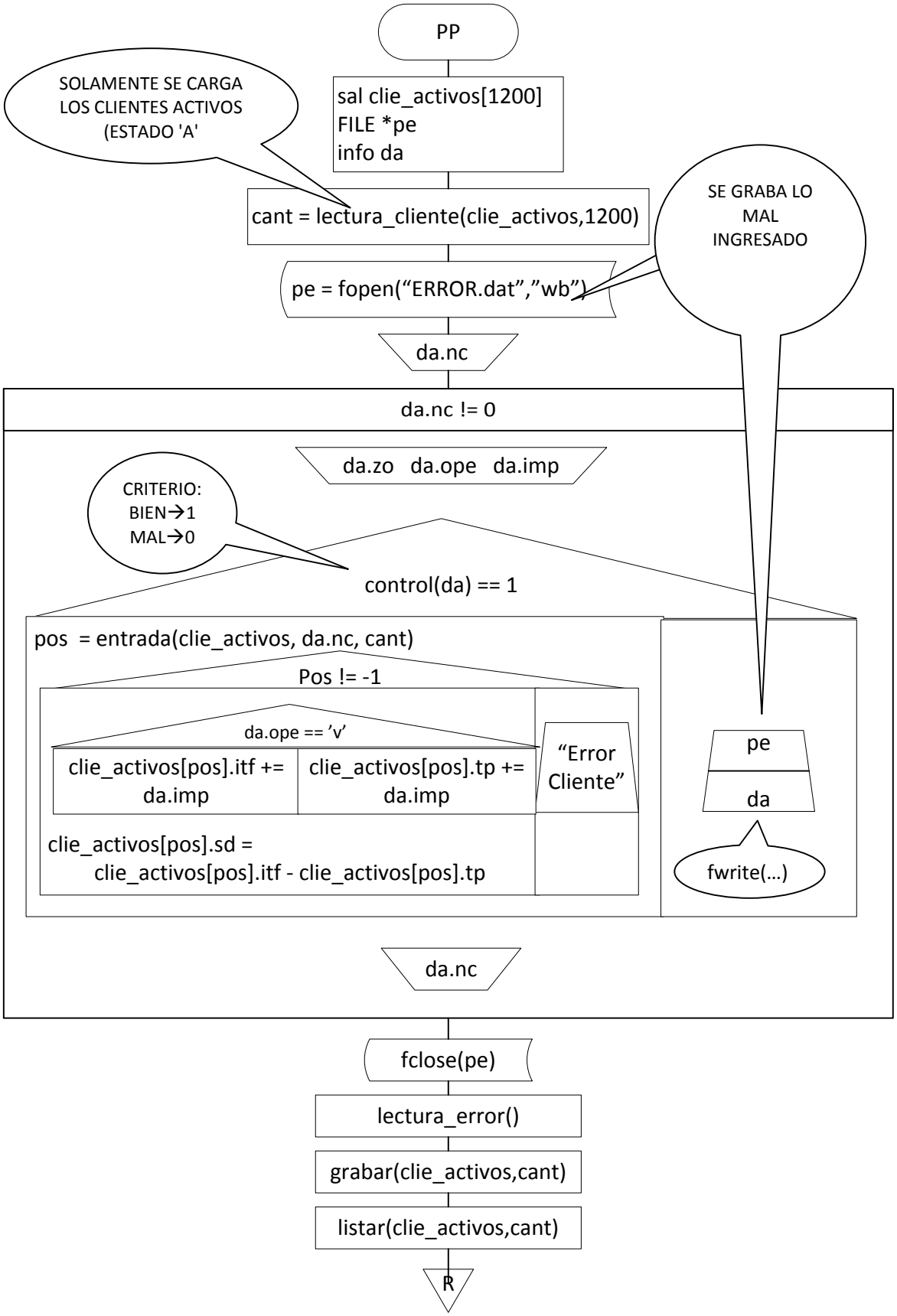
d) Informar los datos grabados en el archivo **ERROR.dat.** Mediante la función **LECTURA_ERROR**.

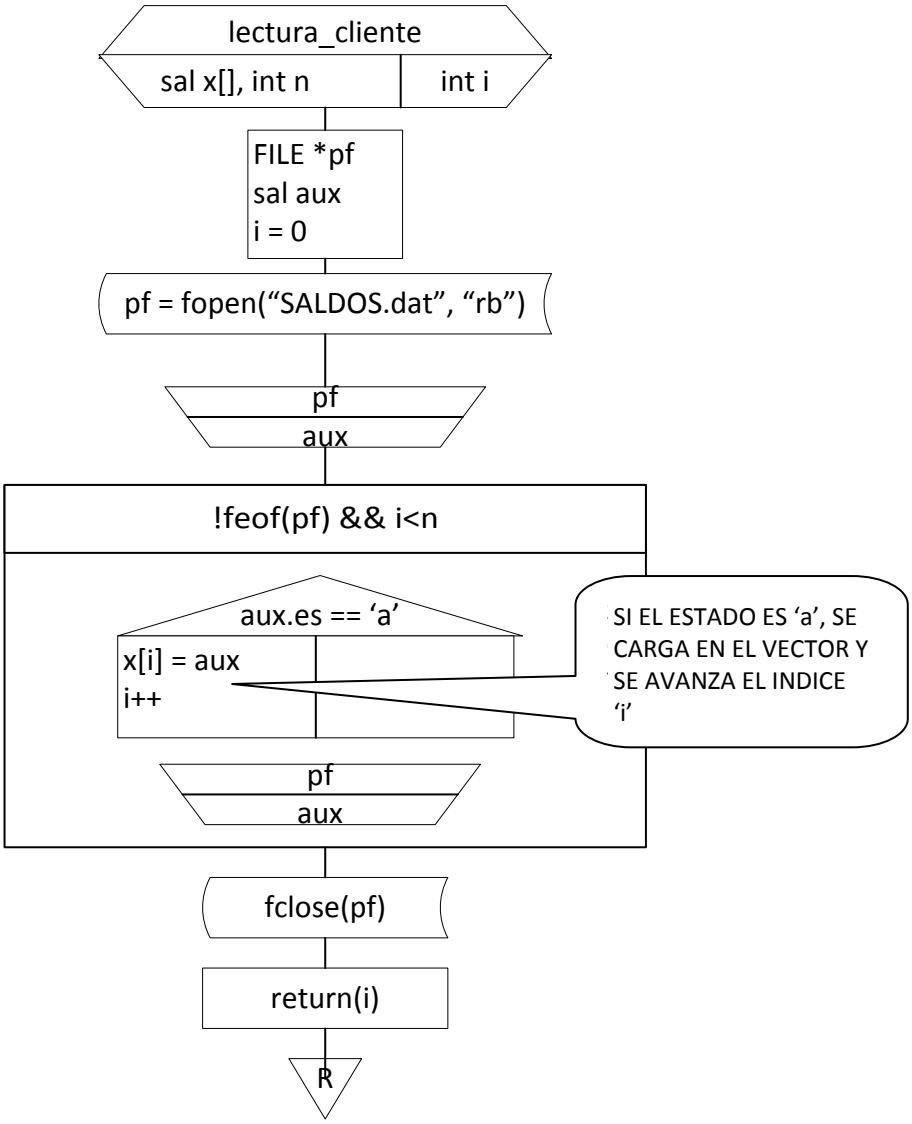
e) Actualizar el archivo SALDOS. Mediante la función **GRABAR**.

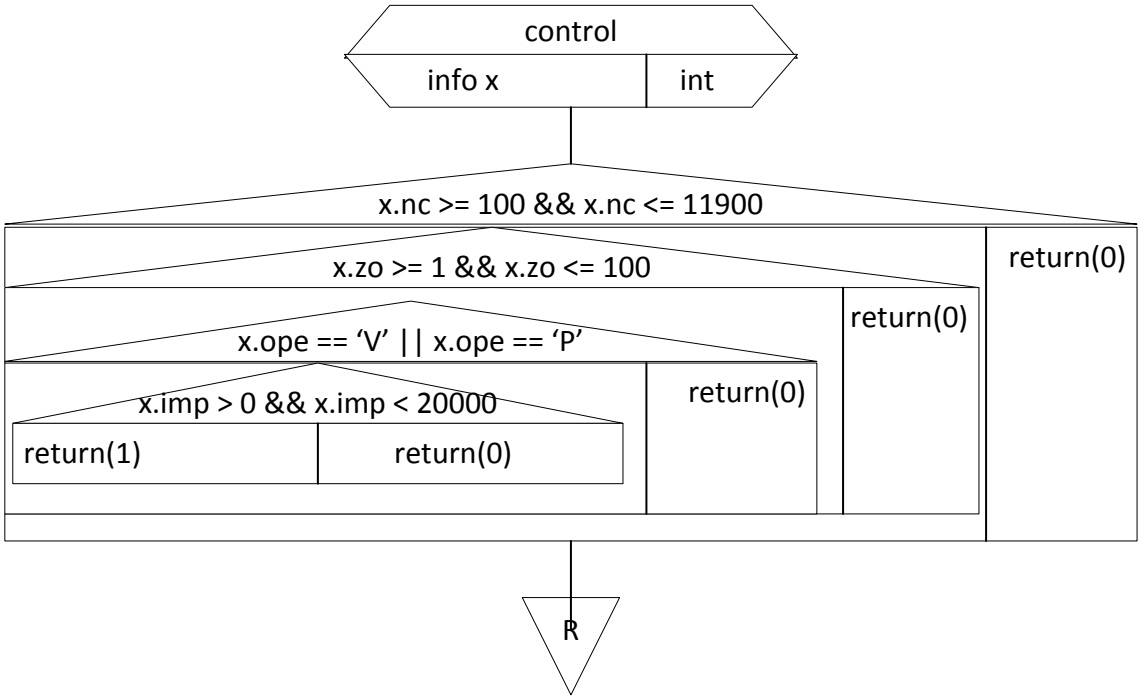
f) Al finalizar el proceso, confeccionar una lista con aquellos clientes, que han quedado con saldo deudor mayor de 1500 pesos. Mediante la función **LISTAR.**

```
                SALDOS DEUDORES MAYORES DE $1500
  Nro. Cliente          Razón Social          Saldo Deudor
  xxxx                  aaaaaaaaaaaa               xxxx.xx
  xxxx                  aaaaaaaaaaaa               xxxx.xx
```

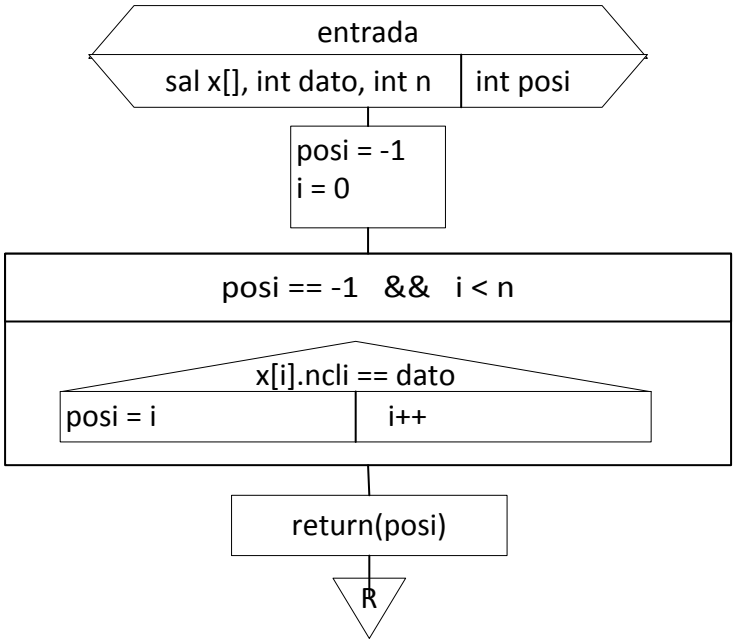


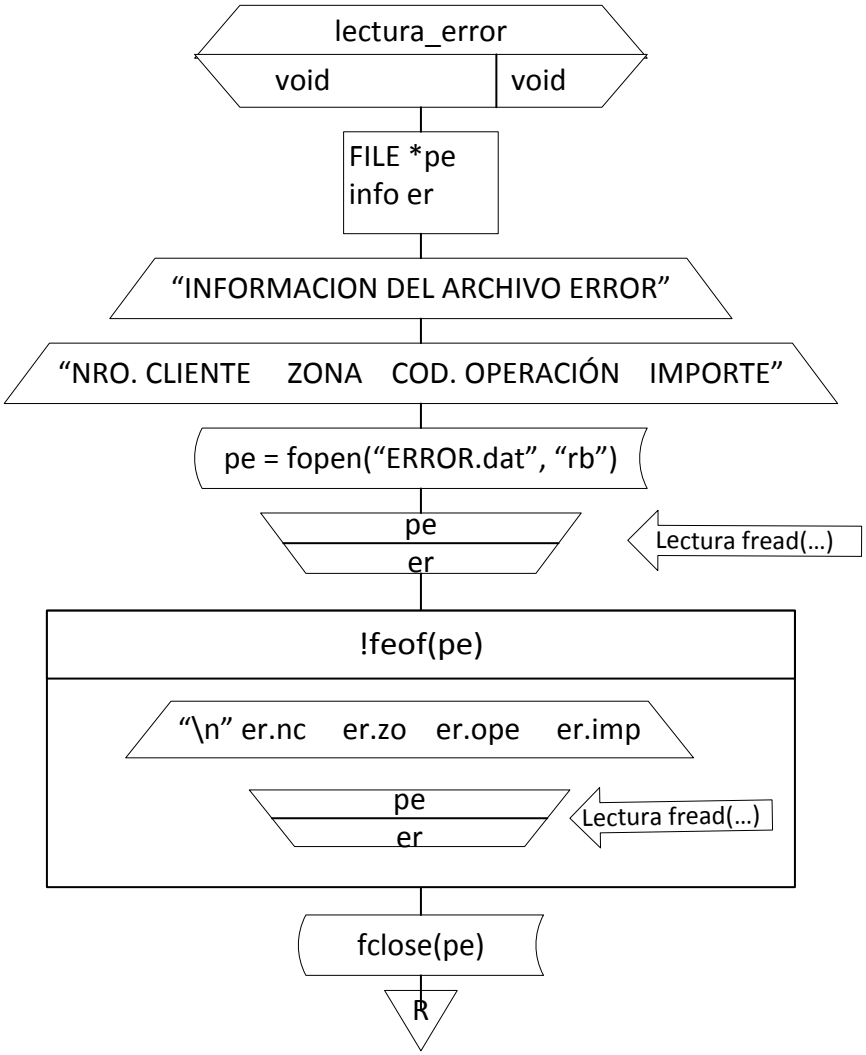

PP
SOLAMENTE SE CARGA LOS CLIENTES ACTIVOS (ESTADO 'A'
sal clie_activos[1200]
FILE *pe
info da
cant = lectura_cliente(clie_activos,1200)
SE GRABA LO MAL INGRESADO
pe = fopen("ERROR.dat","wb")
da.nc
da.nc != 0
da.zo da.ope da.imp
CRITERIO: BIEN→1 MAL→0
control(da) == 1
pos = entrada(clie_activos, da.nc, cant)
Pos != -1
da.ope == 'v'
clie_activos[pos].itf += da.imp
clie_activos[pos].tp += da.imp
"Error Cliente"
clie_activos[pos].sd = clie_activos[pos].itf - clie_activos[pos].tp
pe
da
fwrite(...)
da.nc
fclose(pe)
lectura_error()
grabar(clie_activos,cant)
listar(clie_activos,cant)
R

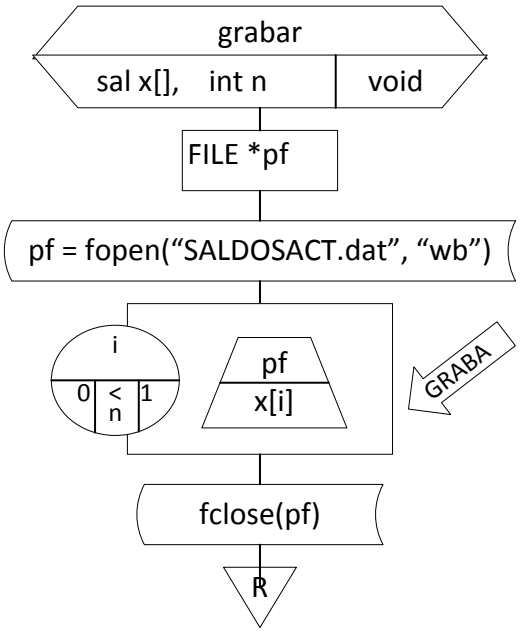

lectura_cliente
sal x[], int n
int i
FILE *pf
sal aux
i = 0
pf = fopen("SALDOS.dat", "rb")
pf
aux
!feof(pf) && i<n
aux.es == 'a'
x[i] = aux
i++
SI EL ESTADO ES 'a', SE CARGA EN EL VECTOR Y SE AVANZA EL INDICE 'i'
pf
aux
fclose(pf)
return(i)
R

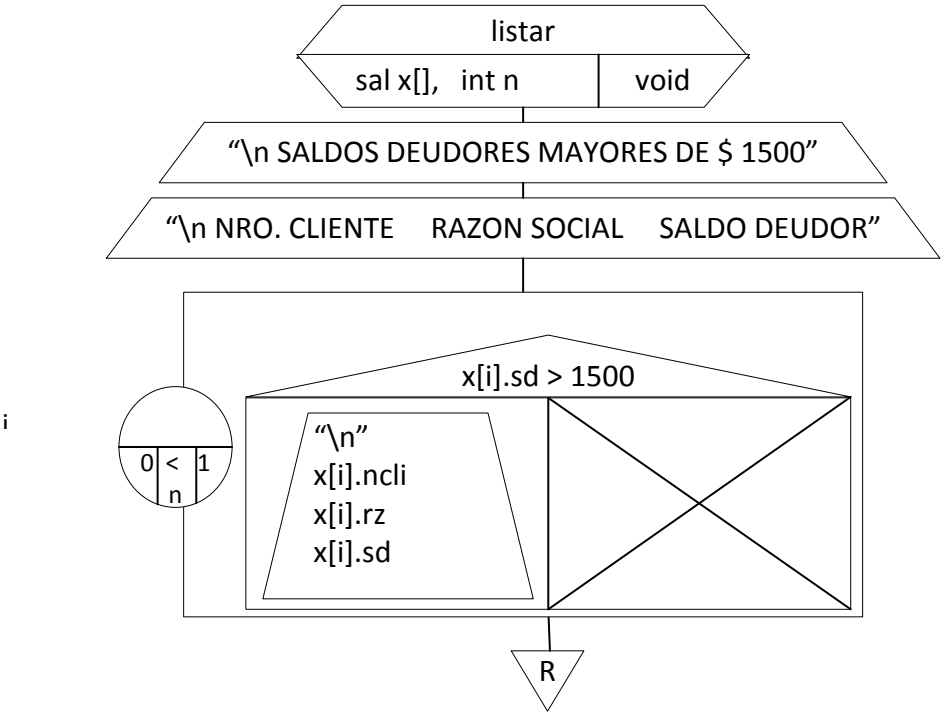


SI TODOS LOS DATOS SE INGRESARON CORRECTAMENTE, LA FUNCIÓN RETORNA 1, SI ALGUNO SE INGRESO MAL, LA FUNCION RETORNA 0.




lectura_error
void
void
FILE *pe
info er
“INFORMACION DEL ARCHIVO ERROR”
“NRO. CLIENTE ZONA COD. OPERACIÓN IMPORTE”
pe = fopen(“ERROR.dat”, “rb”)
pe
er
Lectura fread(…)
!feof(pe)
“\n” er.nc er.zo er.ope er.imp
pe
er
Lectura fread(…)
fclose(pe)
R


grabar
sal x[], int n
void
FILE *pf
pf = fopen(“SALDOSACT.dat”, “wb”)
i
0
<
n
1
pf
x[i]
GRABA
fclose(pf)
R




```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

typedef struct {
       int ncli;
       char rz[26];
       float itf;
       float tp;
       float sd;
       char es;
}sal;

typedef struct {
       int nc;
       int zo;
       float imp;
       char ope;
}info;

int lectura_cliente(sal[],int);
int control(info);
int entrada(sal[],int,int);
void lectura_error(void);
void grabar(sal[],int);
void listar(sal[],int);

void main (void)
{
       sal clie_activos[1200];
       FILE *pe;
       info da;
       int cant=lectura_cliente(clie_activos,1200);
       pe=fopen("ERROR.dat","wb");
       int pos;

       if(pe==NULL)
```

```
        {
                printf("\n NO SE PUEDE ACCEDER");
                getch();
                exit(1);
        }

        printf("\n INGRESE NUMERO CLIENTE (FIN CON CERO)");
        scanf("%d",&da.nc);

        while (da.nc!=0)
        {
                printf("\n INGRESE ZONA:  ");
                scanf("%d",&da.zo);
                fflush(stdin);
                printf("\n INGRESE CODIGO DE OPERACION");
                scanf("%C",&da.ope);
                printf("\n INGRESE IMPORTE");
                scanf("%f",&da.imp);
                if(control(da)==1)
                {
                        pos=entrada(clie_activos,da.nc,cant);
                        if(pos!=-1)
                        {
                                if(da.ope=='v')
                                        clie_activos[pos].itf+=da.imp;
                                else
                                        clie_activos[pos].tp=da.imp;

                                clie_activos[pos].sd=clie_activos[pos].itf-
                        clie_activos[pos].tp;
                        }
                        else
           {
                                printf("\n ERROR CLIENTE");
           }
                }
                else
                        fwrite(&da,sizeof(info),1,pe);

                printf("\n INGRESE NUMERO CLIENTE (FIN CON CERO)");
                scanf("%d",&da.nc);

        }
        fclose(pe);
        lectura_error();
        grabar(clie_activos,cant);
        listar(clie_activos,cant);
        getch();
}

int lectura_cliente(sal x[],int n)
{
    FILE *pf;
    sal aux;
    int i=0;
    pf=fopen("SALDOS.dat","rb");
    if(pf==NULL)
    {
        printf("\n NO SE PUEDE ACCEDER");
        getch();
        exit(1);
```

```
    }
    fread(&aux,sizeof(sal),1,pf);
    while(!feof(pf) && i<n)
    {
        if(aux.es=='a')
        {
            x[i]=aux;
            i++;
        }
        fread(&aux,sizeof(sal),1,pf);
    }
    fclose(pf);
    return(i);
}

int control(info x)
{
    if(x.nc>=0 && x.nc<=11900)
        if(x.zo>=1 && x.nc<=100)
            if(x.ope=='V' || x.ope=='P')
                if(x.imp>0 && x.imp<20000)
                    return(1);
                else
                    return(0);
            else
                return(0);
        else
            return(0);
    else
        return(0);
}

int entrada(sal x[], int dato, int n)
{
    int posi, i;
    posi=-1;
    i=0;
    while(posi==-1 && i<n)
        if(x[i].ncli==dato)
            posi=i;
        else
            i++;
    return(posi);
}

void lectura_error(void)
{
    FILE *pe;
    info er;

    printf("\n INFORMACION DEL ARCHIVO DE ERROR");
    printf("\n NRO. CLIENTE - ZONA - COD. OPERACION - IMPORTE");
    pe=fopen("ERROR.dat","rb");
    if(pe==NULL)
    {
        printf("\n NO SE PUEDE ACCEDER");
        getch();
        exit(1);
    }
    fread(&er,sizeof(info),1,pe);
    while(!feof(pe))
```

```
    {
        printf("\n %d   %d   %c%.2f",er.nc,er.zo,er.ope,er.imp);
        fread(&er,sizeof(info),1,pe);
    }
    fclose(pe);
}

void grabar(sal x[], int n)
{
    FILE *pf;
    int i;
    pf=fopen("SALDOSACT.dat","wb");
    if(pf==NULL)
    {
       getch();
       exit(1);
    }
    for(i=0;i<n;i++)
        fwrite(&x[i],sizeof(sal),1,pf);
    fclose(pf);
}
void listar(sal x[], int n)
{
    int i;
    printf("\n SALDOS DEUDORES MADORES DE $ 1500");
    printf("\n NRO. CLIENTE   RAZÓN SOCIAL   SALDO DEUDOR");
    for(i=0; i<n; i++)
        printf("\n %d   %d   &s   %.2f", x[i].ncli, x[i].rz, x[i].sd);
}
```

3. Expensas

Confeccionar un programa, diagrama y codificación para controlar el pago de las expensas en un edificio de propiedad horizontal con 200 departamentos.
Se pide:

a) Se dispone de un archivo, solamente con la información de los departamentos vendidos llamado **EDIFICIO.dat**, cuyo diseño es:
   - Nro. de departamento (numérico NO correlativo de 3 cifras)
   - Nombre del propietario (máximo 30 caracteres)
   - Superficie en metros cuadrados (xxx.xx)
   - Valor expensas (xxxx. xx)
   - Fecha del Ultimo pago efectuado

b) Ingresar desde el teclado, los pagos realizados. Por cada uno, se ingresa el Nro. del departamento y fecha de pago. Para finalizar, como todos no pagaron, se debe ingresar un nro. de departamento igual a cero.

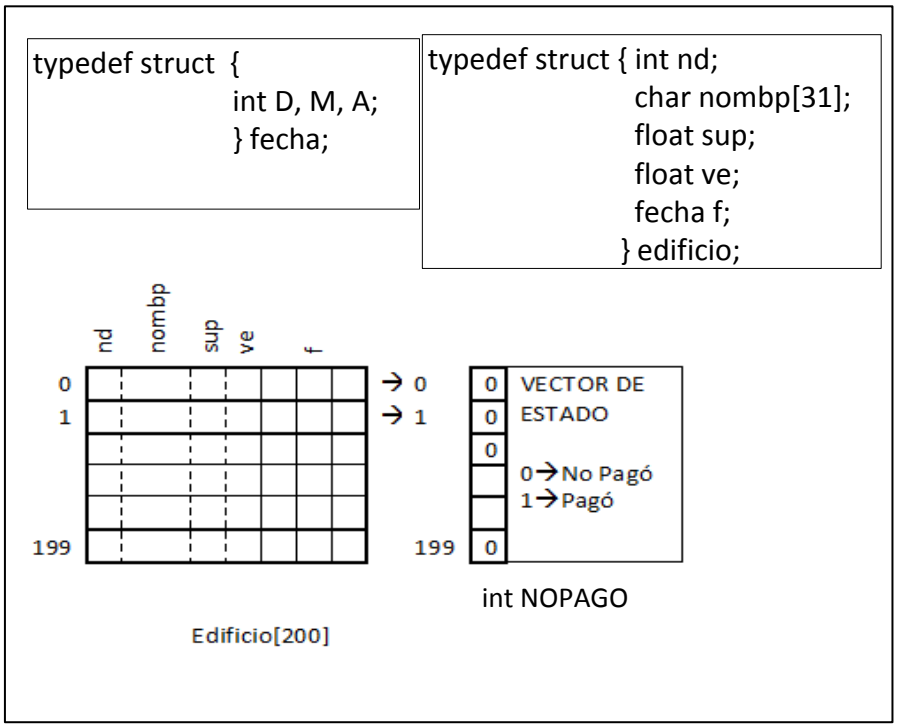
c) Actualizar la Fecha de Ultimo Pago efectuado por cada propietario.

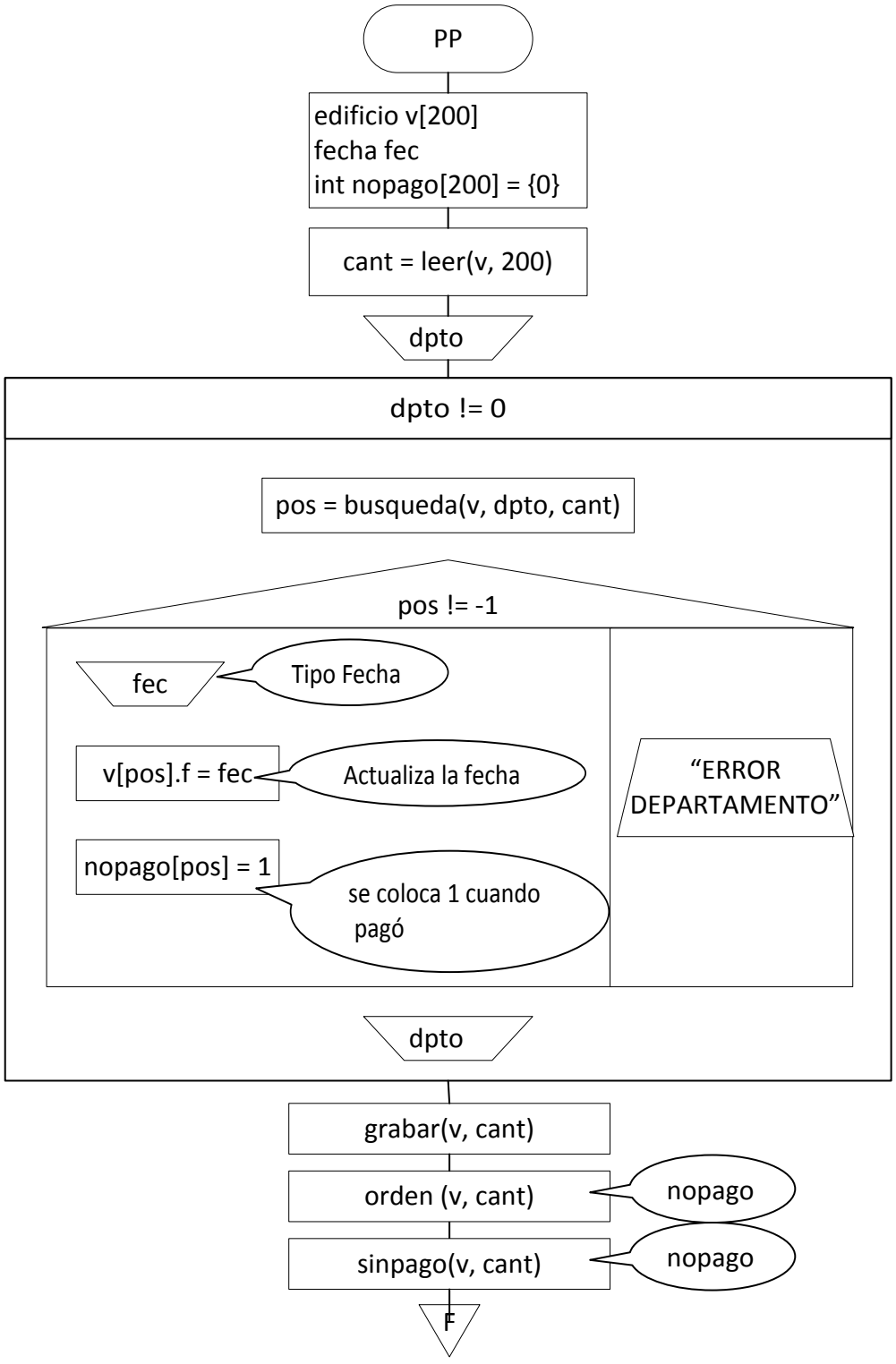
d) Grabar el archivo **EDIFICIO.dat** actualizado.

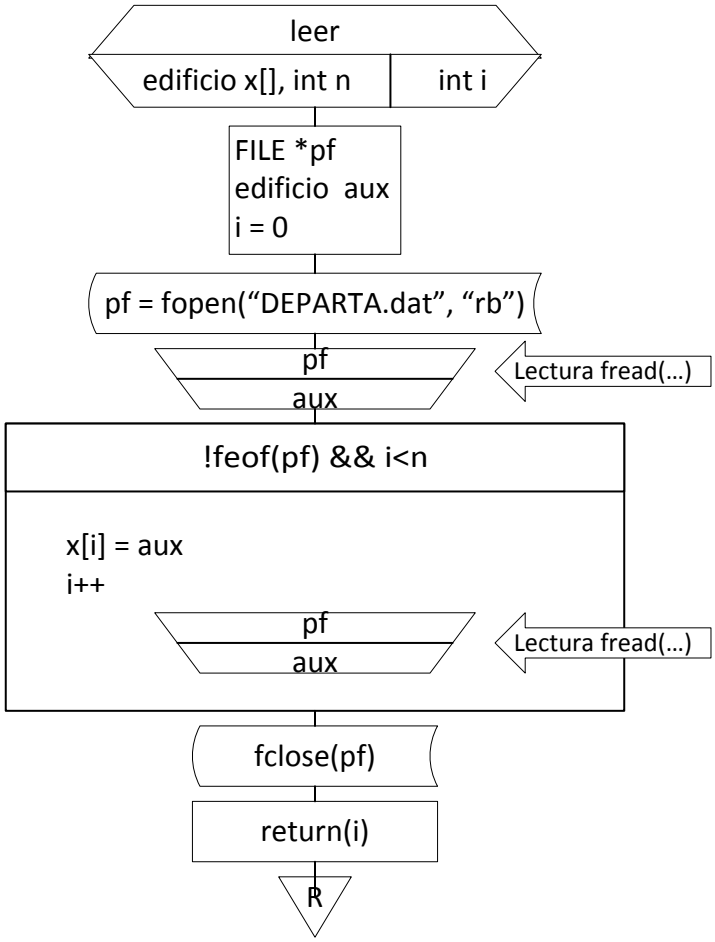
e) Confeccionar una función llamada **SINPAGO**, con parámetros que reciba la información necesaria para determinar el siguiente listado con los morosos (no han pagado), ordenado en forma descendente por valor expensa.

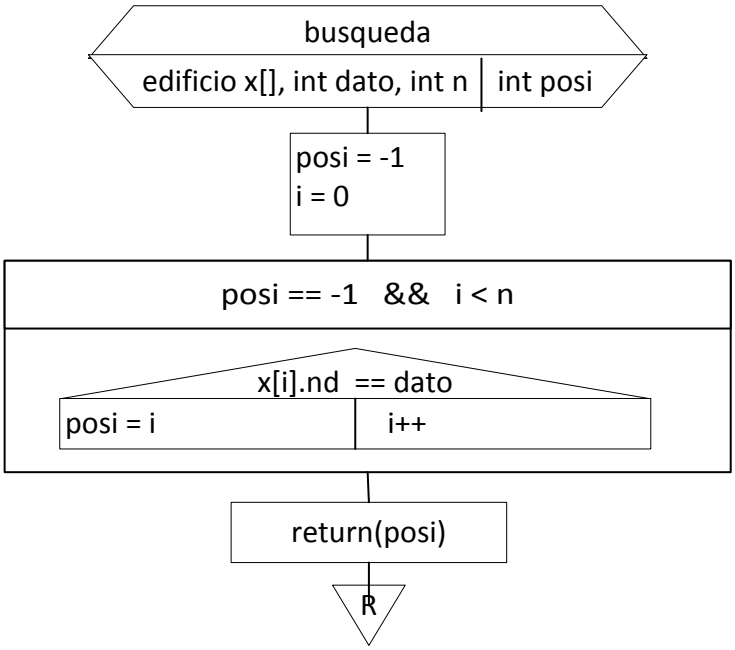
```
                    EXPENSAS IMPAGAS
DEPARTAMENTO        PROPIETARIO        VALOR EXPENSA
XXX                 XXXXXXXXXX               XXXX.XX
```

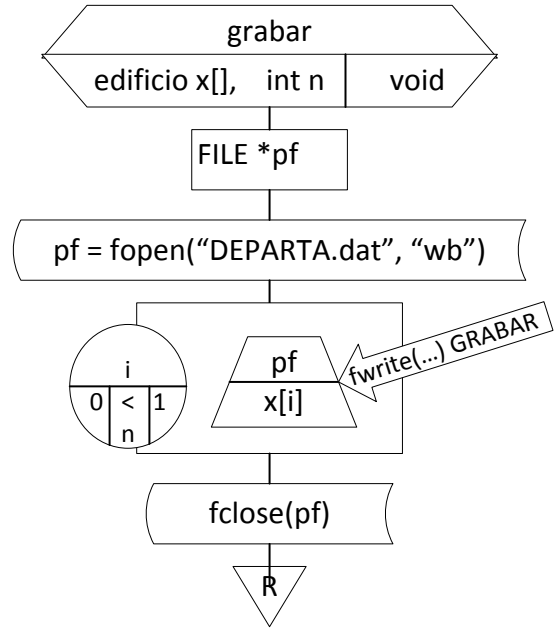
f) Determinar el porcentaje de departamentos que hayan pagado la expensa sobre el total de departamentos vendidos utilizando la función **PORCEN**.

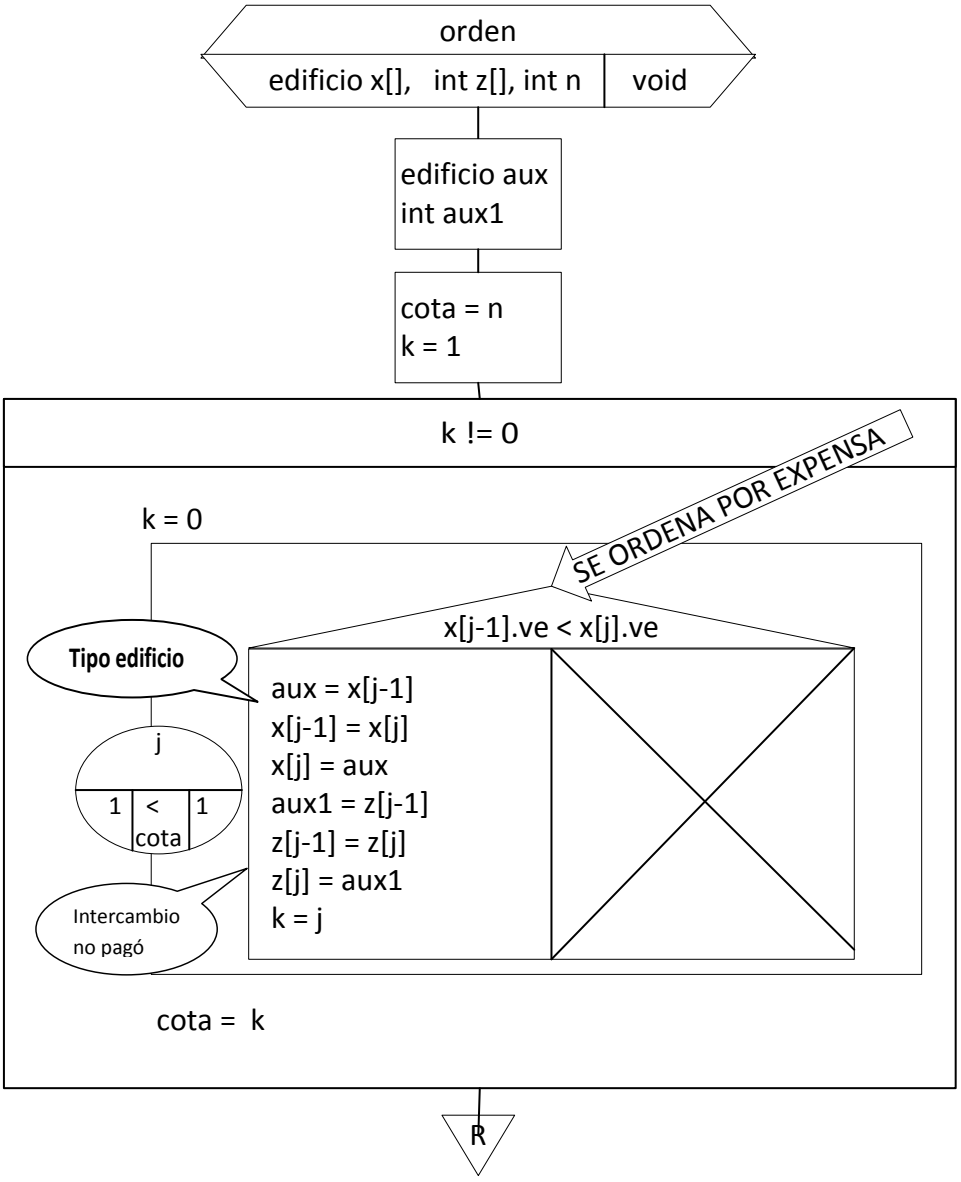


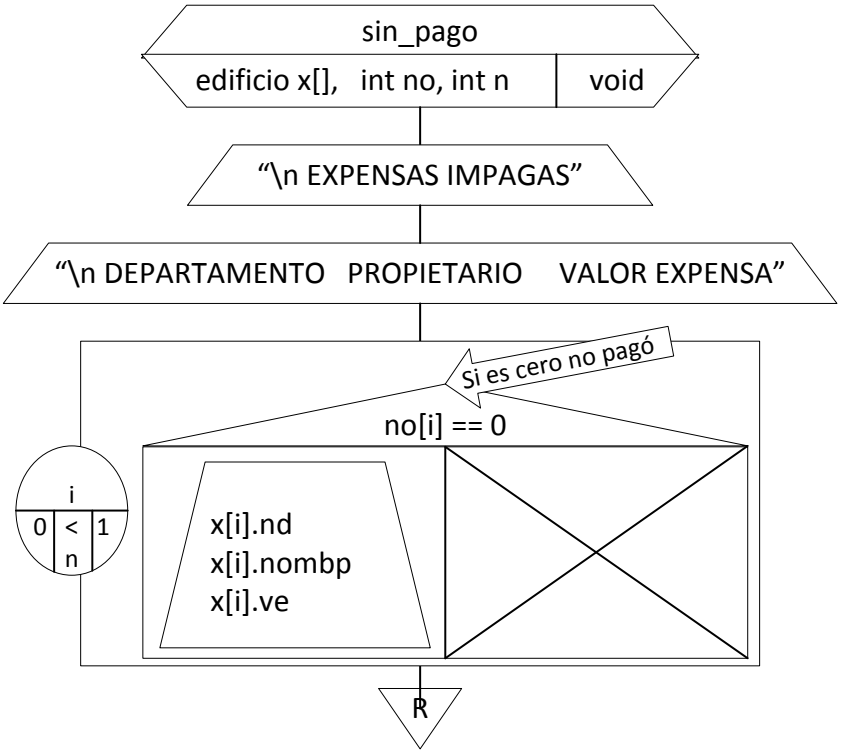

PP
edificio v[200]
fecha fec
int nopago[200] = {0}
cant = leer(v, 200)
dpto
dpto != 0
pos = busqueda(v, dpto, cant)
pos != -1
fec
Tipo Fecha
v[pos].f = fec
Actualiza la fecha
nopago[pos] = 1
se coloca 1 cuando pagó
"ERROR DEPARTAMENTO"
dpto
grabar(v, cant)
orden (v, cant)
nopago
sinpago(v, cant)
nopago
F


leer
edificio x[], int n
int i
FILE *pf
edificio aux
i = 0
pf = fopen("DEPARTA.dat", "rb")
pf
aux
Lectura fread(...)
!feof(pf) && i<n
x[i] = aux
i++
pf
aux
Lectura fread(...)
fclose(pf)
return(i)
R


busqueda
edificio x[], int dato, int n
int posi
posi = -1
i = 0
posi == -1 && i < n
x[i].nd == dato
posi = i
i++
return(posi)
R


grabar
edificio x[], int n
void
FILE *pf
pf = fopen("DEPARTA.dat", "wb")
i
0 < n 1
pf
x[i]
fwrite(...) GRABAR
fclose(pf)
R


orden
edificio x[], int z[], int n
void
edificio aux
int aux1
cota = n
k = 1
k != 0
k = 0
SE ORDENA POR EXPENSA
x[j-1].ve < x[j].ve
Tipo edificio
aux = x[j-1]
x[j-1] = x[j]
x[j] = aux
aux1 = z[j-1]
z[j-1] = z[j]
z[j] = aux1
k = j
j
1 < cota 1
Intercambio
no pagó
cota = k
R




```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

typedef struct {
                int d, m ,a;
                }fecha;

typedef struct {
                int nd;
                char nombp[31];
                float sup;
                float ve;
                fecha f;
                }edificio;

int leer(edificio[],int);
int busqueda(edificio[],int,int);
void grabar(edificio[],int);
void orden(edificio[],int[], int);
void sinpago(edificio[], int[], int);

void main(void)
{
    edificio v[200];
    fecha fec;
    int nopago[200];
    int cant, dpto, pos;

    cant=leer(v,200);
    printf("\n Ing. Departamento (Fin cero)");
    scanf("%d%, &dpto");
    while(dpto!=0)
    {
```

```
        pos=busqueda(v,dpto,cant);
        if(pos!=-1)
        {
            printf("\n Ingrese Fecha(Día, Mes, Año)");
            scanf("%d%d%d",&fec.d, &fec.m, &fec.a );
            v[pos].f=fec;
            nopago[pos];
        }
        else
            printf("\n ERROR NRO. DEPARTAMENTO");
    }
    grabar(v,cant);
    orden(v,nopago,cant);
    sinpago(v,nopago,cant);
    getch();
}

int leer(edificio x[],int n)
{
    FILE *pf;
    edificio aux;
    int i=0;
    pf=fopen("DEPARTA.dat","rb");
    if(pf==NULL)
    {
        printf("\n NO SE PUEDE ACCEDER");
        getch();
        exit(1);
    }
    fread(&aux,sizeof(edificio),1,pf);
    while(!feof(pf) && i<n)
    {
        x[i]=aux;
        i++;
        fread(&aux,sizeof(edificio),1,pf);
    }
    fclose(pf);
    return(i);
}

int busqueda(edificio x[], int dato, int n)
{
    int posi, i;
    posi=-1;
    i=0;
    while(posi==-1 && i<n)
        if(x[i].nd==dato)
            posi=1;
    else
        i++;
    return(posi);
}

void grabar(edificio x[],int n)
{
    FILE *pf;
    int i;
    pf=fopen("DEPARTA.dat","wb");
    if(pf==NULL)
    {
        printf("\n NO SE PUEDE ACCEDER");
```

```
        getch();
        exit(1);
    }
    for(i=0; i<n; i++)
        fwrite(&x[i], sizeof(edificio), 1, pf);
    fclose(pf);
}

void orden(edificio x[],int z[], int n)
{
    edificio aux;
    int cota, k, i, j, aux1;
    cota=n;
    k=1;
    while(j!=0)
    {
        k=0;
        for(i=1; j<cota; j++)
            if(x[j-1].ve<x[j].ve)
        {
            aux=x[j-1];
            x[j-1]=x[j];
            x[j]=aux;
            aux1=z[j-1];
            z[j-1]=z[j];
            z[j]=aux1;
            k=j;
        }
        cota=k;
    }
}

void sinpago(edificio x[], int no[], int n)
{
    int i;
    printf("\n EXPENSAS IMPAGAS");
    printf("\n DEPARTAMENTO   PROPIETARIO   VALOR EXPENSA");
    for(i=0; i<n; i++)
        if(no[i]==i++)
            printf("\n %s %.2f", x[i].nd, x[i].nombp, x[i].ve);
}
```

4. Alquiler de autos

Una empresa de alquiler de autos dispone como máximo de 50 coches, de cada uno de ellos se conoce.

- Nro. Patente
- Apellido y nombre del propietario

Esta información está grabada en un archivo **AUTO.dat**, que ya existe. Utilizar función **Carga**, para ingresarlo a memoria.

Además, cada vez que se alquila un auto, se tiene la siguiente información producida durante un mes, esta información se encuentra grabada en el archivo "ALQUILER.DAT"**,** secuencial, no ordenado.

- Nro. de Patente
- Día de alquiler (1 a 30)
- Importe del alquiler

Utilizar función **Búsqueda** para el Nro. de Patente

Se pide determinar:

a) Un listado ordenado en forma descendente por recaudación total de cada auto alquilado durante el mes, indicando:

```
        RECAUDACIÓN POR AUTOMÓVIL
PATENTE DEL AUTO        RECAUDACIÓN
xxxxxxx                     xxxx.xx
```

Utilizar la función **Orden** para ordenar los datos y la función **Listado1** para mostrarlos

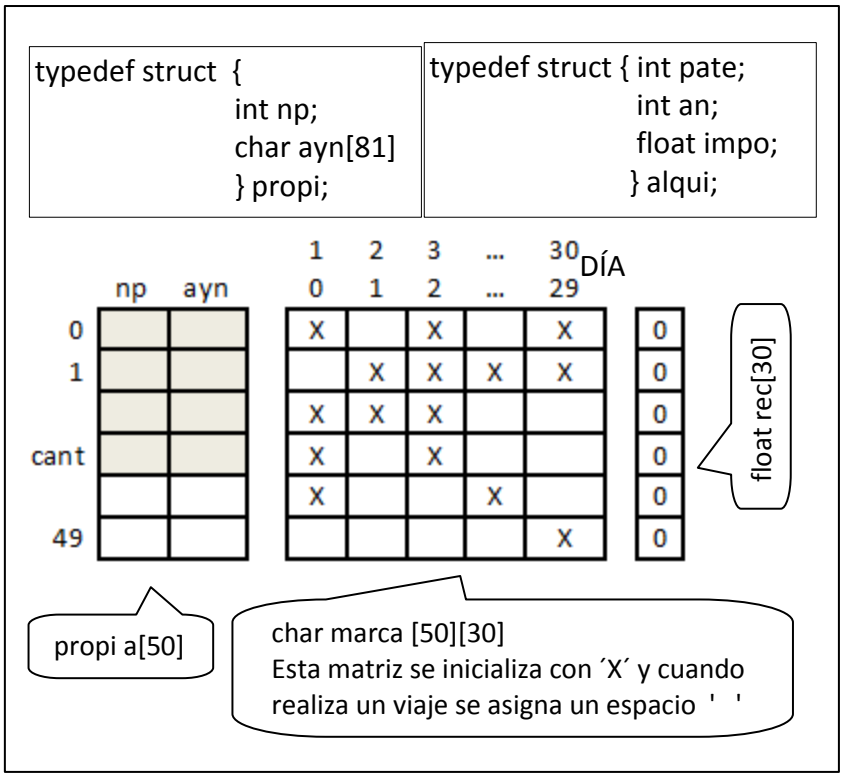
b) Un listado donde figure cada automóvil y los días en el mes que NO se alquiló. Ingresar la fecha con la función **Fecha**, con el formato día, mes, año, validando día (1 a 30) , mes (1 a 12) , año (2011 o 2012)

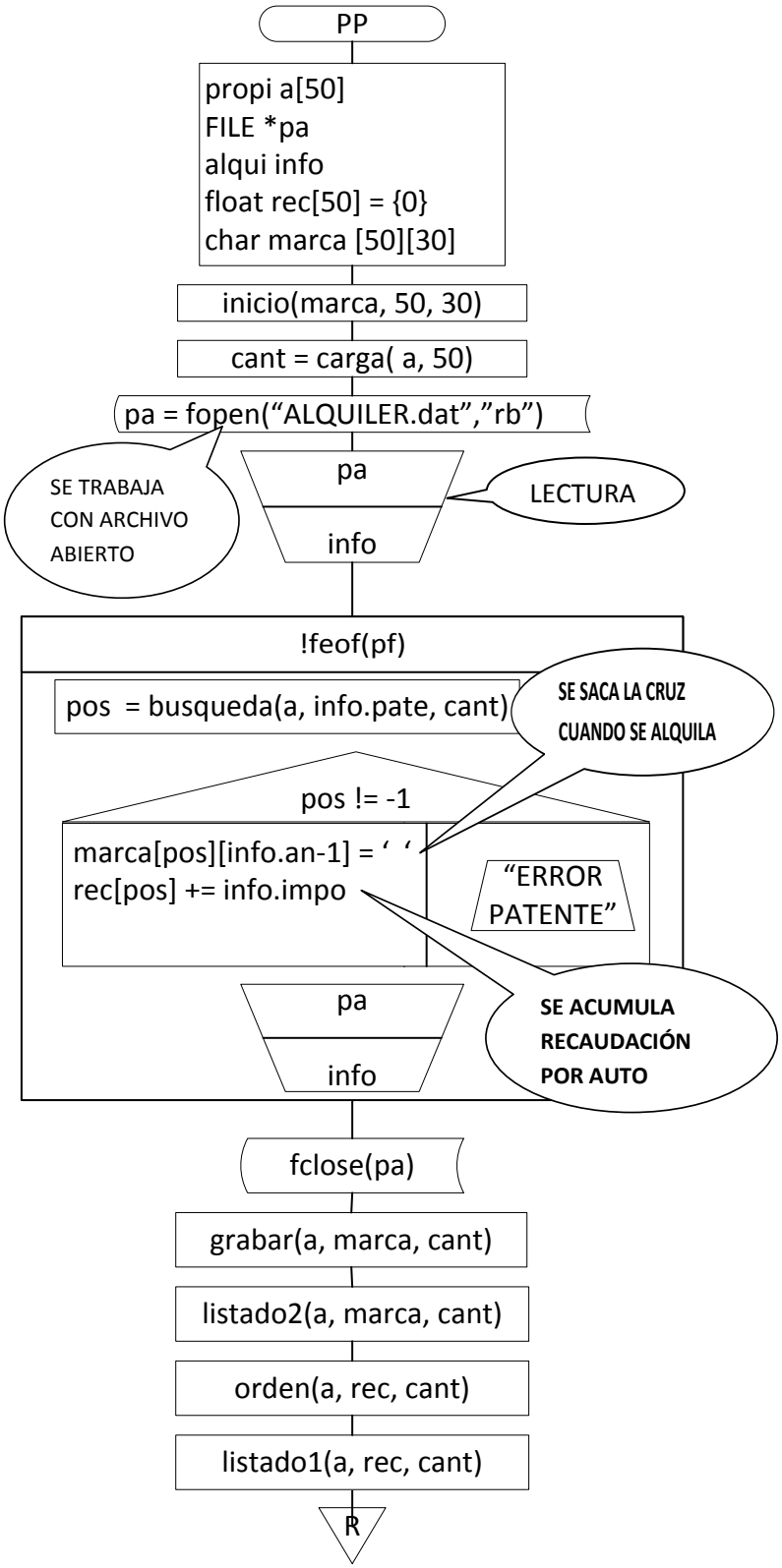
```
                              LISTADO AL dd/mm/aa
Patente Auto/DIA    1     2     3     4    ..................28    29    30
      BSU123        x     x                                  x
      CJU236                    x     x                      x
      AHP888        x           x     x                      x     x
```

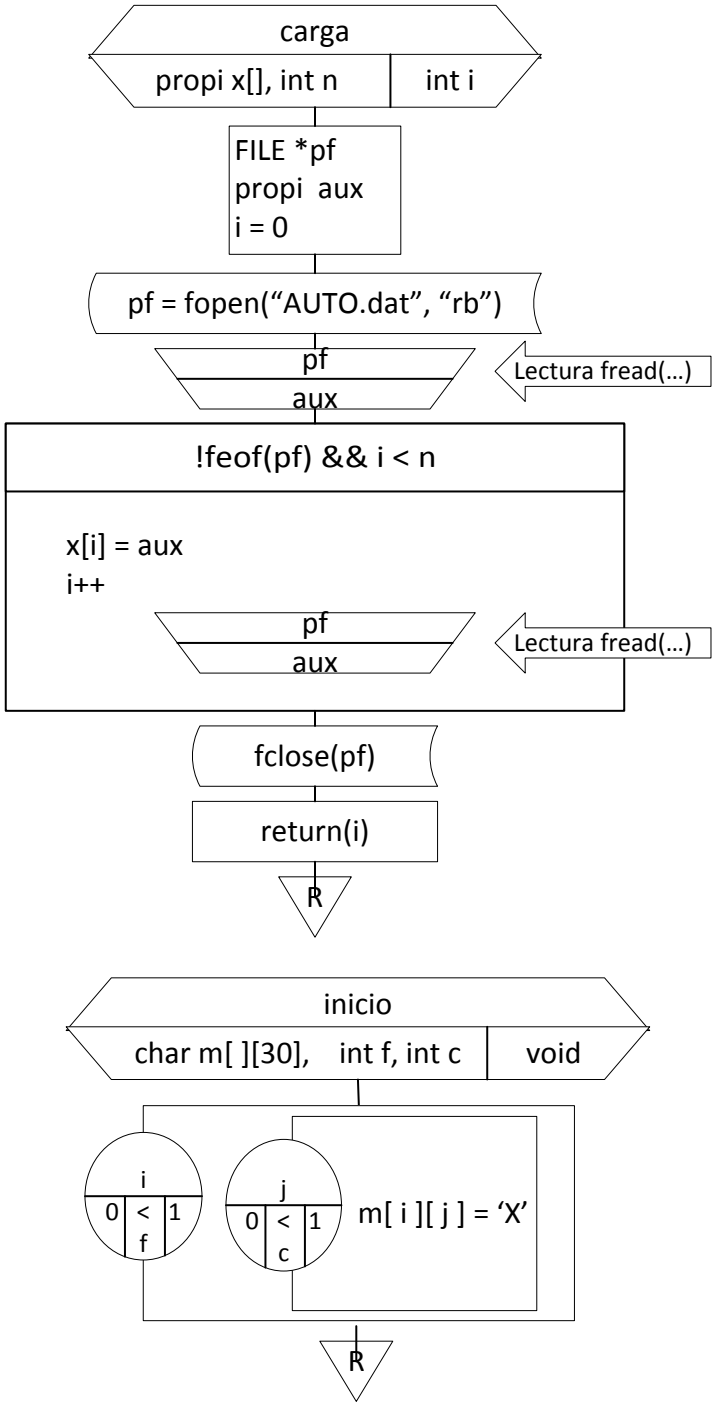
Utilizar la función **Listado2**

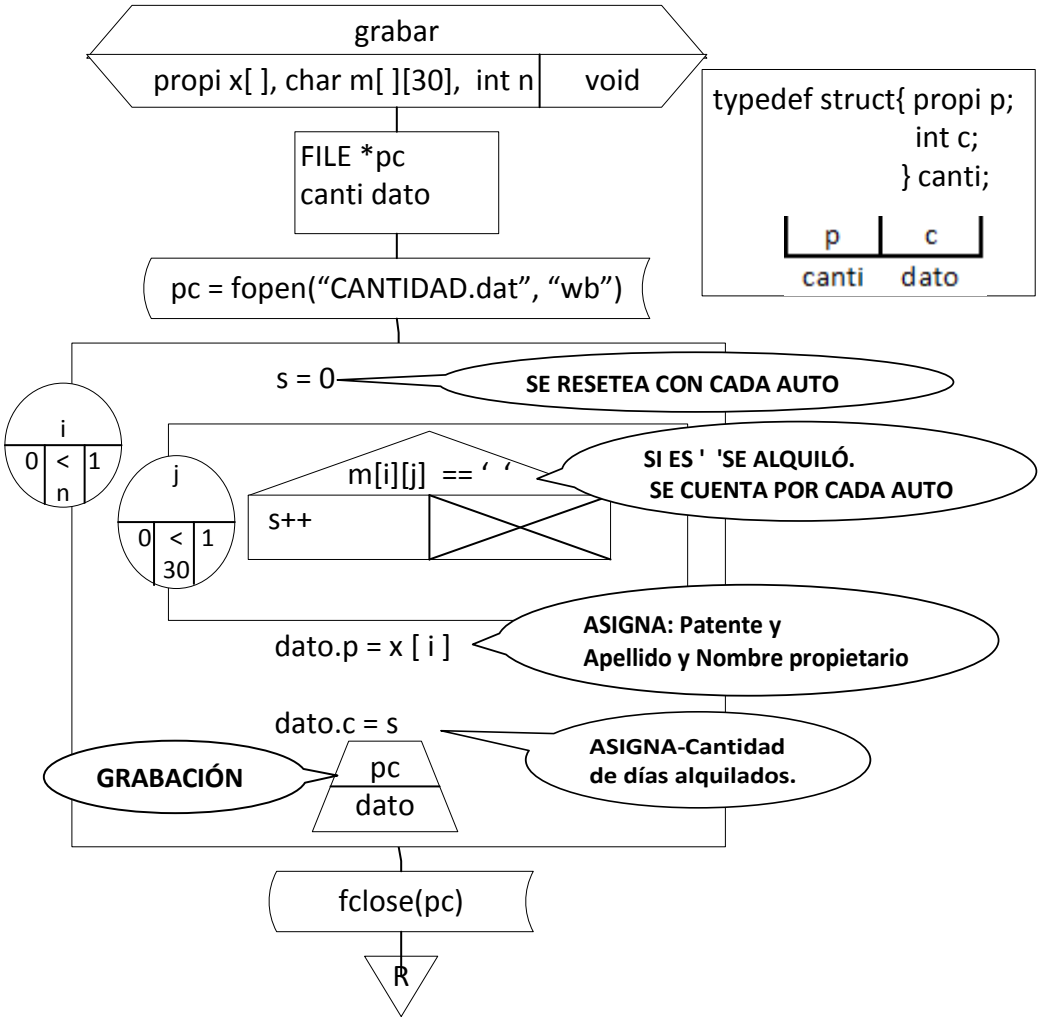
c) Grabar el archivo “CANTIDAD.DAT” con los siguientes datos:
   - Nro. de Patente
   - Apellido y nombre del propietario
   - Cantidad de días que se alquiló cada auto en el mes.

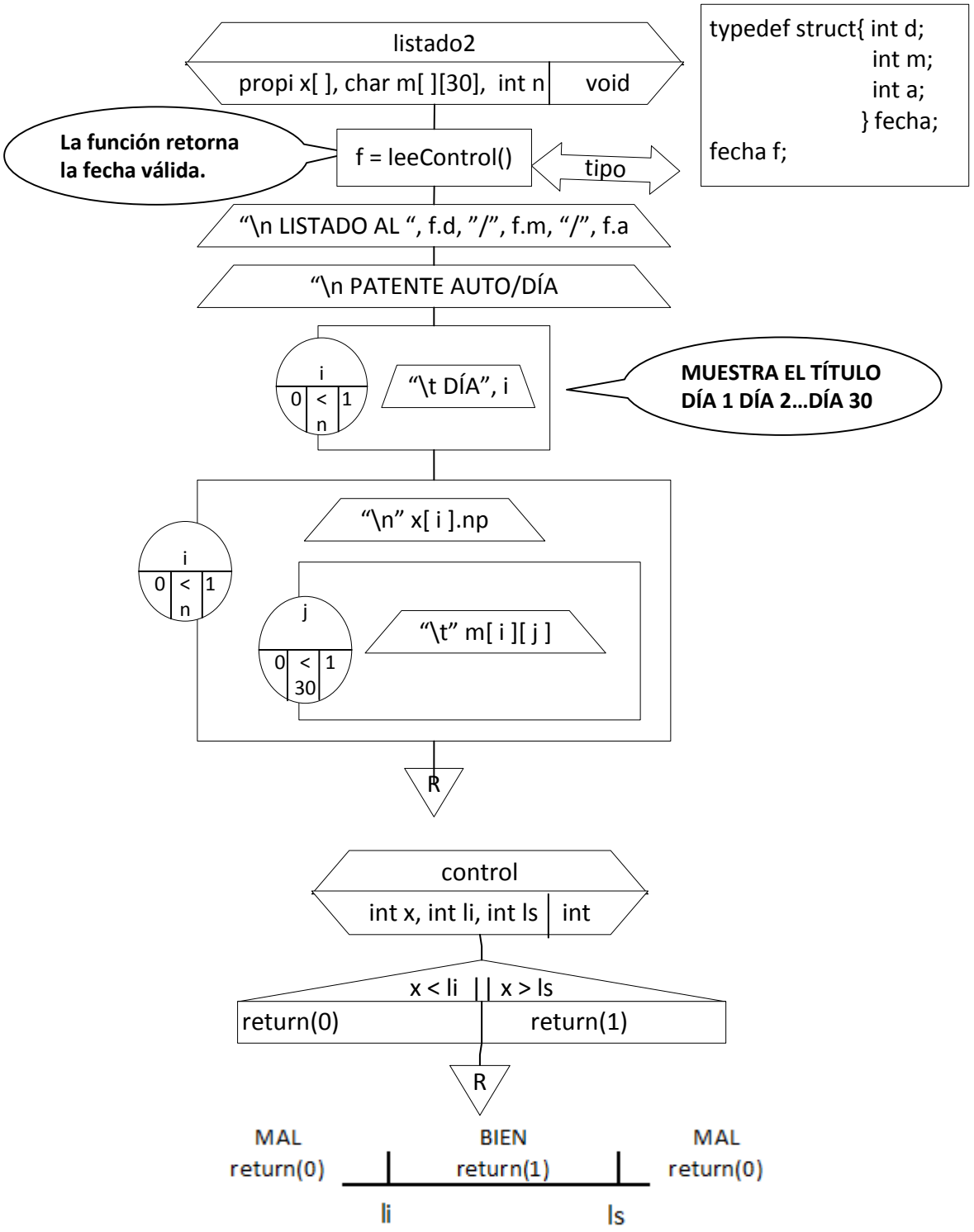
   Utilizar la función **Grabar**

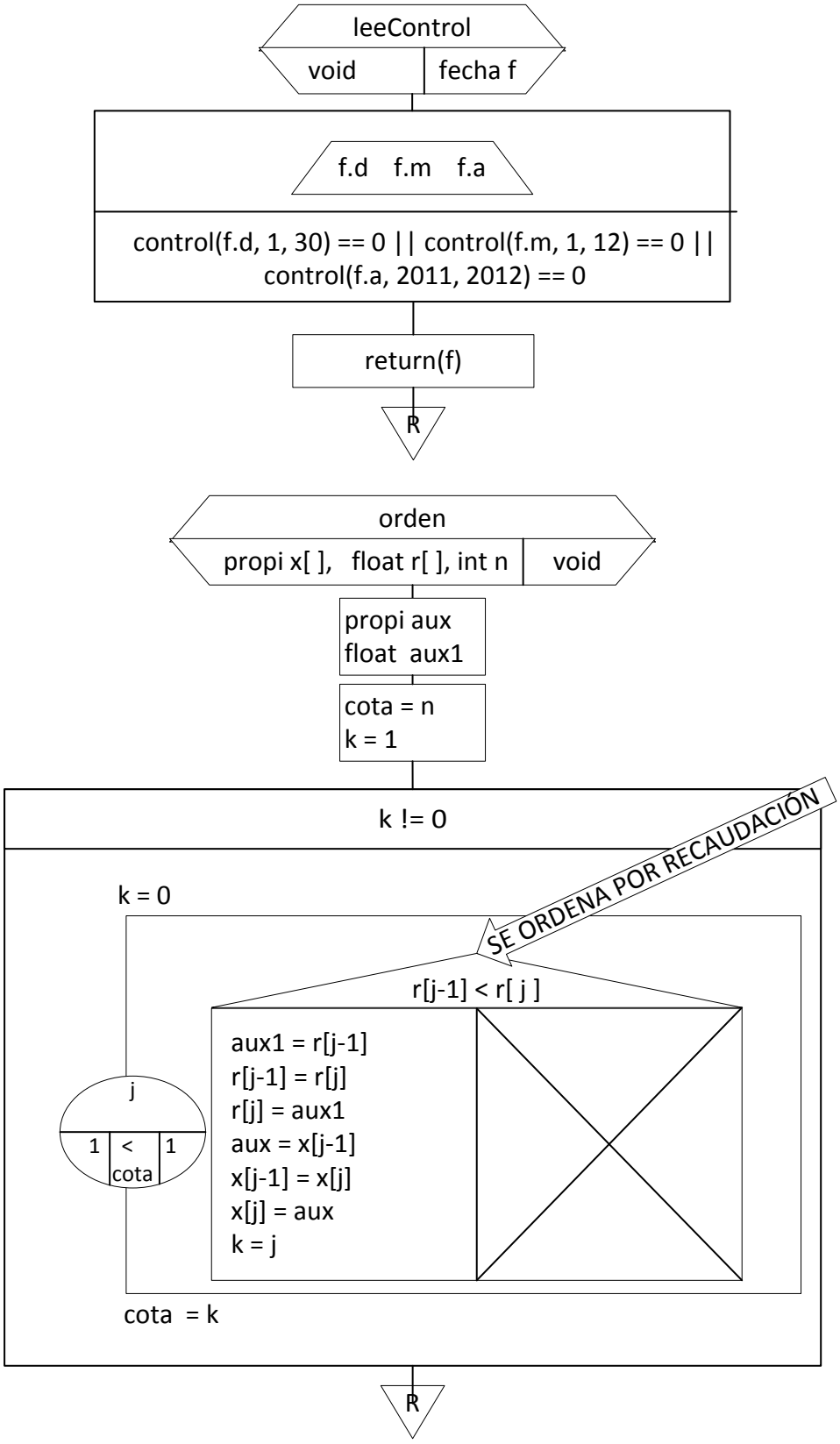

typedef struct {
int np;
char ayn[81]
} propi;
typedef struct { int pate;
int an;
float impo;
} alqui;
1 2 3 ... 30 DÍA
0 1 2 ... 29
np ayn
0
1
cant
49
float rec[30]
propi a[50]
char marca [50][30]
Esta matriz se inicializa con ´X´ y cuando realiza un viaje se asigna un espacio ' '

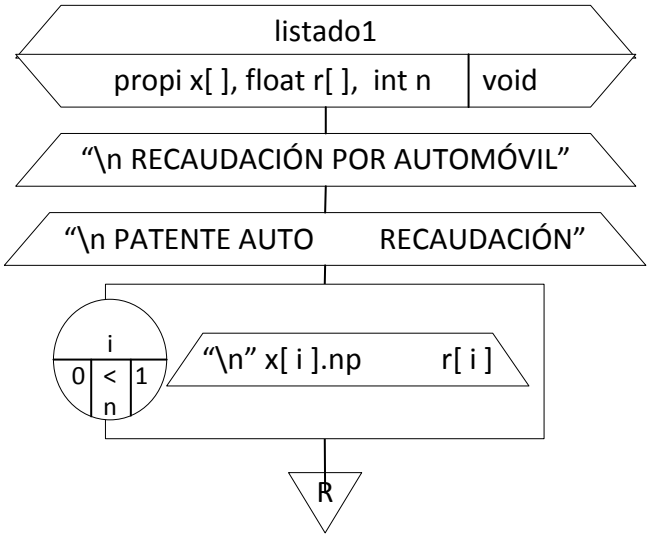

PP
propi a[50]
FILE *pa
alqui info
float rec[50] = {0}
char marca [50][30]
inicio(marca, 50, 30)
cant = carga(a, 50)
pa = fopen("ALQUILER.dat","rb")
SE TRABAJA CON ARCHIVO ABIERTO
pa
info
LECTURA
!feof(pf)
pos = busqueda(a, info.pate, cant)
SE SACA LA CRUZ CUANDO SE ALQUILA
pos != -1
marca[pos][info.an-1] = ' '
rec[pos] += info.impo
"ERROR PATENTE"
pa
info
SE ACUMULA RECAUDACIÓN POR AUTO
fclose(pa)
grabar(a, marca, cant)
listado2(a, marca, cant)
orden(a, rec, cant)
listado1(a, rec, cant)
R


carga
propi x[], int n
int i
FILE *pf
propi aux
i = 0
pf = fopen("AUTO.dat", "rb")
pf
aux
Lectura fread(...)
!feof(pf) && i < n
x[i] = aux
i++
pf
aux
Lectura fread(...)
fclose(pf)
return(i)
R
inicio
char m[ ][30], int f, int c
void
i
0 < 1
f
j
0 < 1
c
m[i][j] = 'X'
R


grabar
propi x[ ], char m[ ][30], int n
void
FILE *pc
canti dato
pc = fopen("CANTIDAD.dat", "wb")
typedef struct{ propi p;
int c;
} canti;
p
c
canti
dato
s = 0
SE RESETEA CON CADA AUTO
i
0
<
n
1
j
0
<
30
1
m[i][j] == ' '
s++
SI ES ' 'SE ALQUILÓ.
SE CUENTA POR CADA AUTO
dato.p = x [i]
ASIGNA: Patente y
Apellido y Nombre propietario
dato.c = s
ASIGNA-Cantidad
de días alquilados.
GRABACIÓN
pc
dato
fclose(pc)
R


listado2
propi x[ ], char m[ ][30], int n
void
typedef struct{ int d;
int m;
int a;
} fecha;
fecha f;
La función retorna la fecha válida.
f = leeControl()
tipo
"\n LISTADO AL ", f.d, "/", f.m, "/", f.a
"\n PATENTE AUTO/DÍA
i
0 < n 1
"\t DÍA", i
MUESTRA EL TÍTULO
DÍA 1 DÍA 2...DÍA 30
"\n" x[i].np
i
0 < n 1
j
0 < 30 1
"\t" m[i][j]
R
control
int x, int li, int ls
int
x < li || x > ls
return(0)
return(1)
R
MAL
return(0)
BIEN
return(1)
MAL
return(0)
li
ls


leeControl
void
fecha f
f.d f.m f.a
control(f.d, 1, 30) == 0 || control(f.m, 1, 12) == 0 ||
control(f.a, 2011, 2012) == 0
return(f)
R
orden
propi x[ ], float r[ ], int n
void
propi aux
float aux1
cota = n
k = 1
k != 0
k = 0
SE ORDENA POR RECAUDACIÓN
r[j-1] < r[j]
aux1 = r[j-1]
r[j-1] = r[j]
r[j] = aux1
aux = x[j-1]
x[j-1] = x[j]
x[j] = aux
k = j
j
1 < cota 1
cota = k
R


listado1
propi x[ ], float r[ ], int n
void
"\n RECAUDACIÓN POR AUTOMÓVIL"
"\n PATENTE AUTO RECAUDACIÓN"
i
0
<
1
n
"\n" x[i].np r[i]
R

Elementos de Programación

UNIDAD 11. Corte de Control
Anexo Ejercicios Resueltos

INDICE

UNIDAD 11
CORTE DE CONTROL – Ejercicios Resueltos

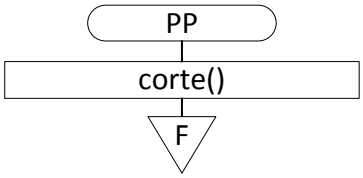
1. Ventas

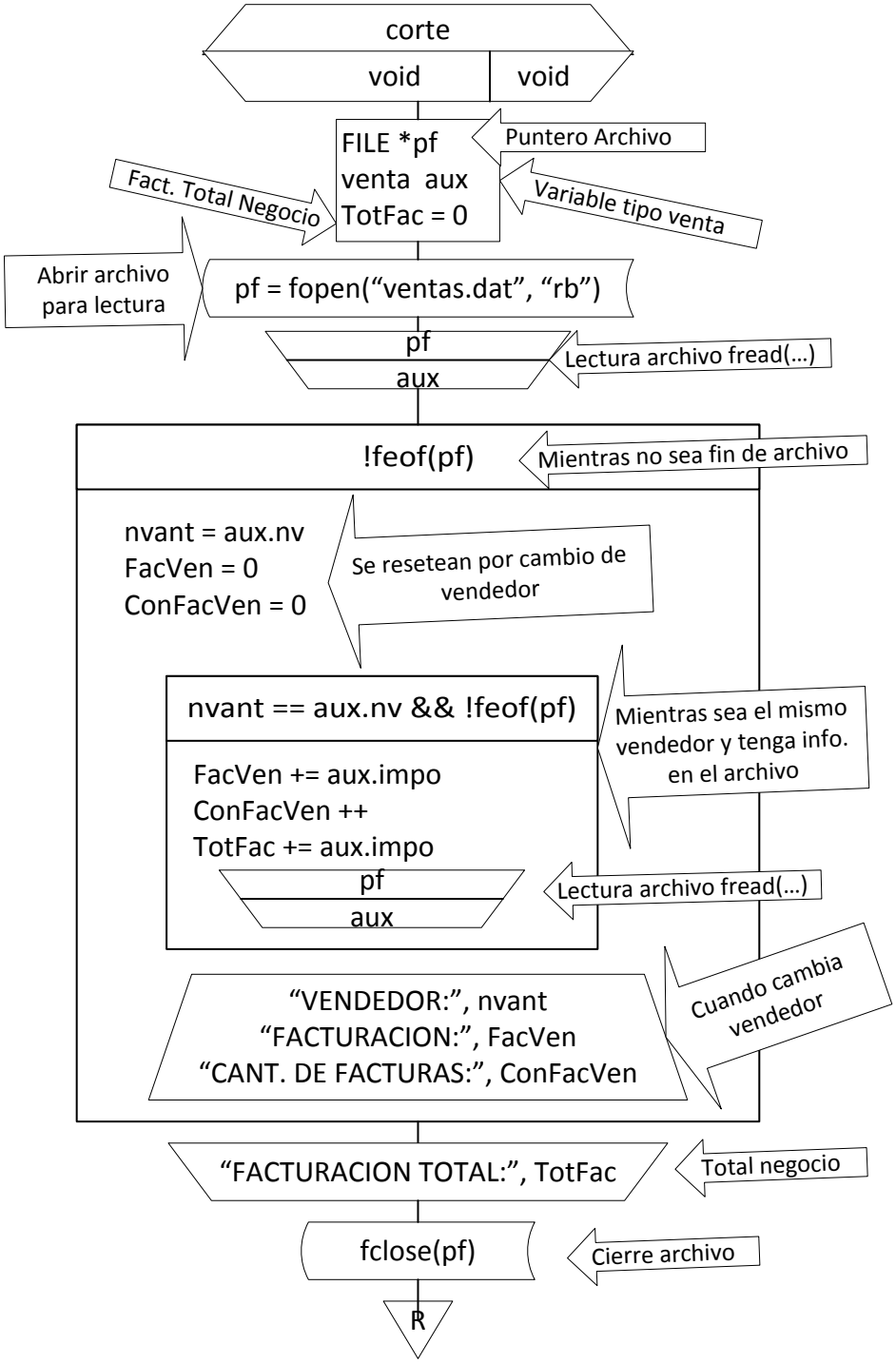
Se dispone de un archivo llamado ventas.dat que contiene la información de las ventas realizadas por la empresa a lo largo del mes. El archivo se encuentra ordenado por número de vendedor y tiene la siguiente estructura:

- Número de factura (entero)
- Número de Vendedor (entero)
- Importe de la factura (real)

Se desea realizar un programa que leyendo el archivo calcule la cantidad de ventas y el importe total facturado por cada vendedor. Al finalizar mostrar además el importe total facturado por la empresa.

```
typedef struct  {
                int nf,nv;
                float impo;
                } venta;
```




corte
void
void
FILE *pf
venta aux
TotFac = 0
Puntero Archivo
Variable tipo venta
Fact. Total Negocio
Abrir archivo para lectura
pf = fopen("ventas.dat", "rb")
pf
aux
Lectura archivo fread(...)
!feof(pf)
Mientras no sea fin de archivo
nvant = aux.nv
FacVen = 0
ConFacVen = 0
Se resetean por cambio de vendedor
nvant == aux.nv && !feof(pf)
Mientras sea el mismo vendedor y tenga info. en el archivo
FacVen += aux.impo
ConFacVen ++
TotFac += aux.impo
pf
aux
Lectura archivo fread(...)
"VENDEDOR:", nvant
"FACTURACION:", FacVen
"CANT. DE FACTURAS:", ConFacVen
Cuando cambia vendedor
"FACTURACION TOTAL:", TotFac
Total negocio
fclose(pf)
Cierre archivo
R

2. Control de Personal

La empresa GoogleMar dispone del archivo secuencial PERSONAL.dat con los siguientes datos de cada uno de sus 175 operarios activos.

- Nro. de Legajo (Nro. entero de 3 cifras NO correlativo)
- Apellido y Nombres (80 caracteres)
- Valor Hora (xx.xx)
- Total de horas trabajadas (*actualizar sumando)*
- Total de sueldos a cobrar a la Fecha (*actualizar sumando)*
- Fecha del último proceso (dia-mes-año) (*actualizar cambiando)*

Confeccionar el diagrama de lógica y la respectiva codificación para:

a) Generar un vector llamado **ArrayPerso** con los datos existentes en el archivo **PERSONAL.dat**, utilizando la función **LECTURA_PERSONAL**.

b) Ingresar desde el teclado la fecha del día de proceso: día (1 a 30), mes (1 a 12) y año (2011 a 2013 inclusive). Solo permitir el ingreso de una fecha correcta, por error volver a solicitar toda la fecha. Confeccionar la función **LEECONTROL** para tal fin.

c) También se dispone del archivo **HORAS.dat** donde por cada día trabajado se registró.
   - Nro. de Legajo (Nro. entero de 3 cifras NO correlativo)
   - Cantidad de horas trabajadas

   Estos datos se encuentran en el archivo ordenados por legajo.
   Para el control de la existencia del Nro. de Legajo Confeccionar la función **BUSQUEDA** con los argumentos necesarios. Si no existe, grabar la información ingresada (Nro. de legajo – cantidad de horas trabajadas) en el archivo **ERROR.dat**.

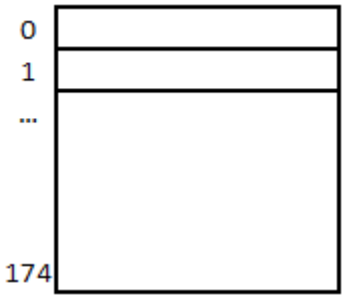
d) Actualizar **ArrayPerso** con las novedades ingresadas
   El importe a cobrar se calcula como el producto entre las horas trabajadas y el valor que cobra por hora el operario. La fecha de proceso es la ingresada por teclado.

e) Actualizar el archivo **PERSONAL.dat** Mediante función **GRABAR**.

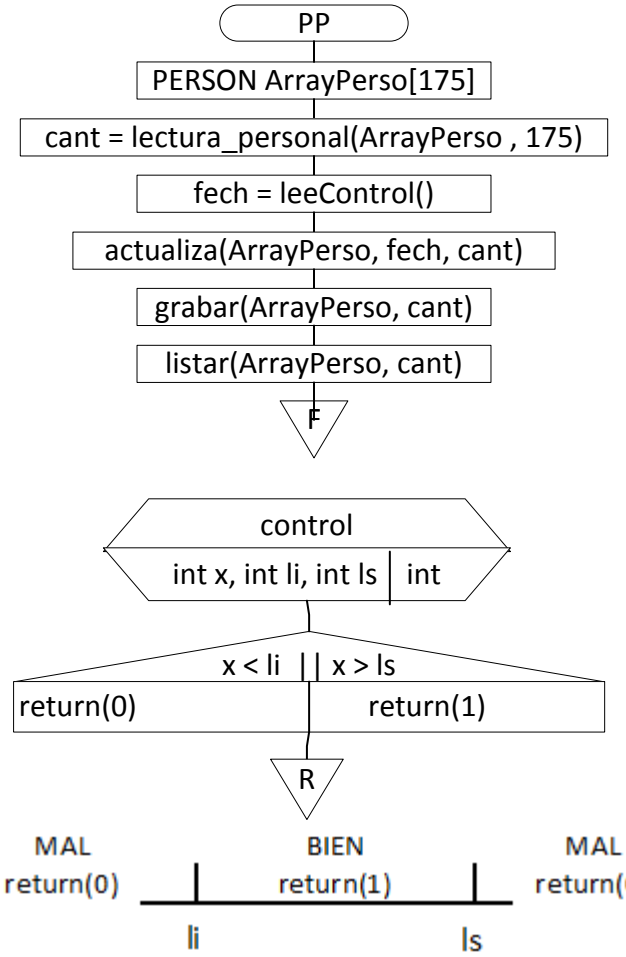
f) Informar los datos de aquellos legajos que tienen un acumulado de sueldo mayor a $ 1.500 y las horas NO exceden a 140. Función **LISTAR**

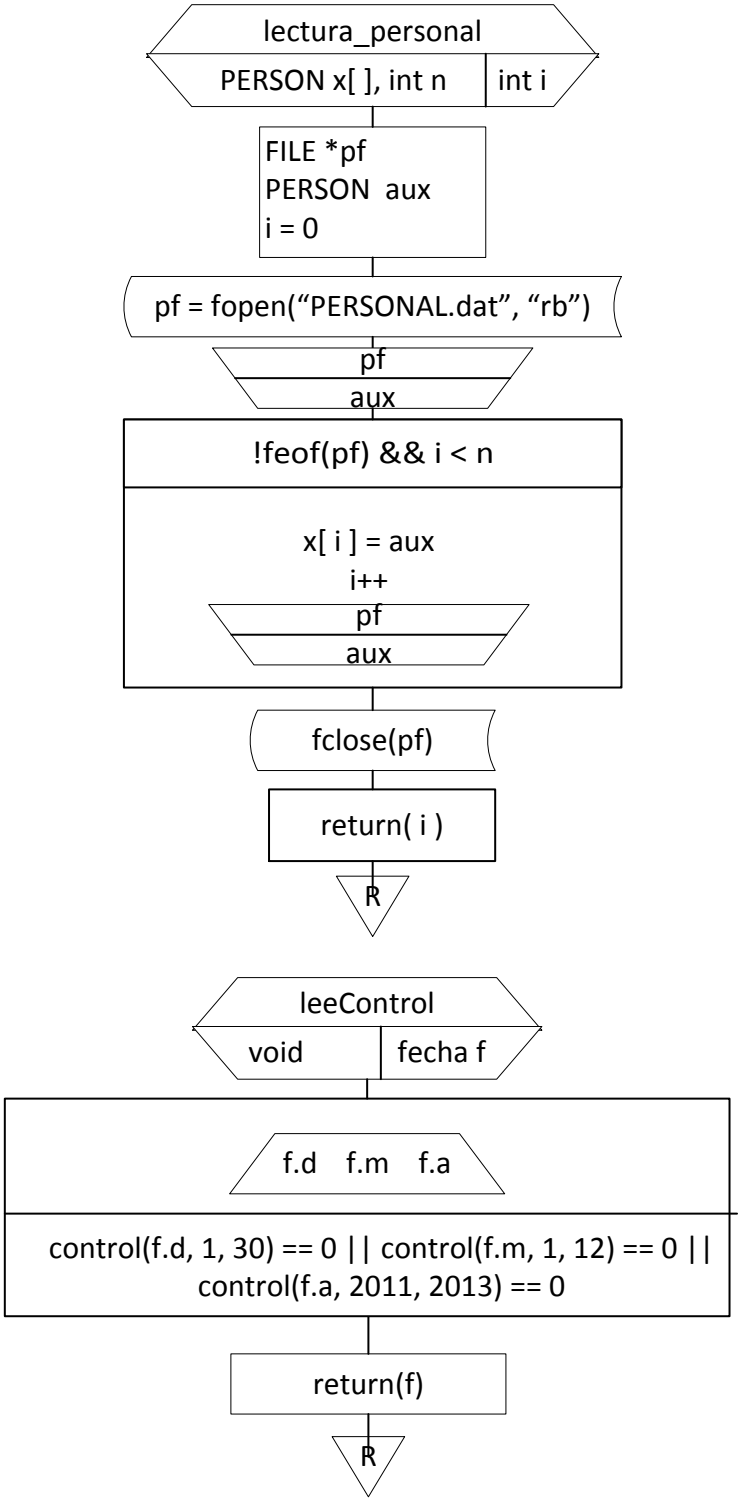
```
typedef struct  {
                int NL, THT;
                char AyN[81];
                float VH, TS;
                FECHA FE
                } PERSON;
```

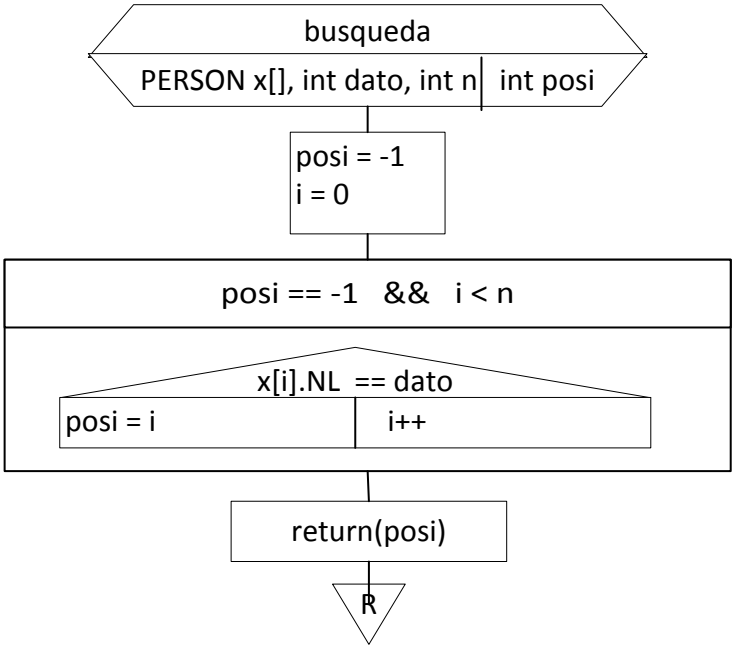
```
typedef struct  {
                int D, M, A;
                } FECHA;
```

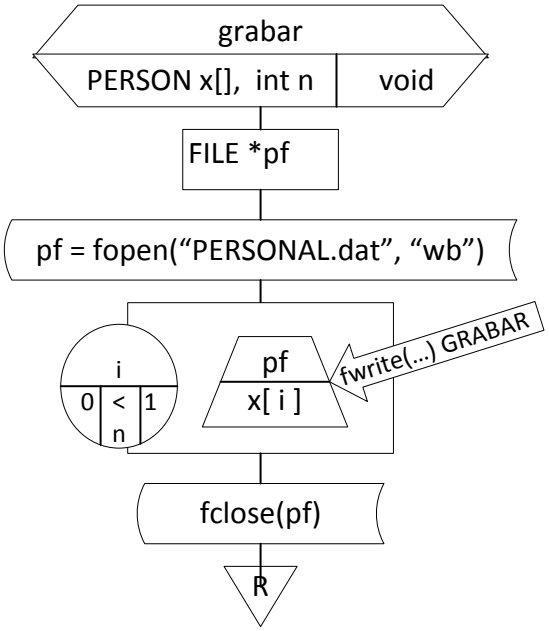


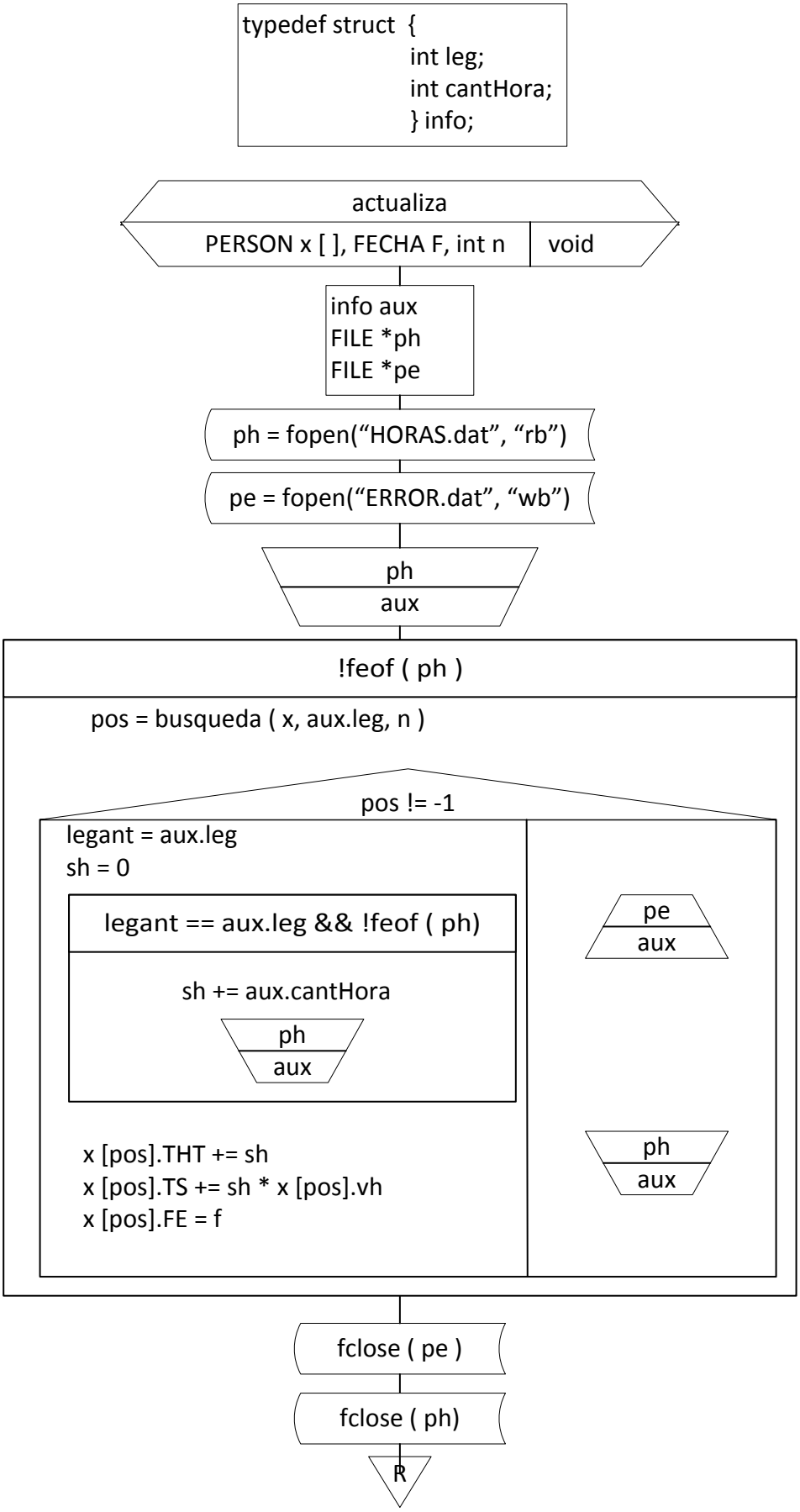
PERSON ArrayPerso[175]

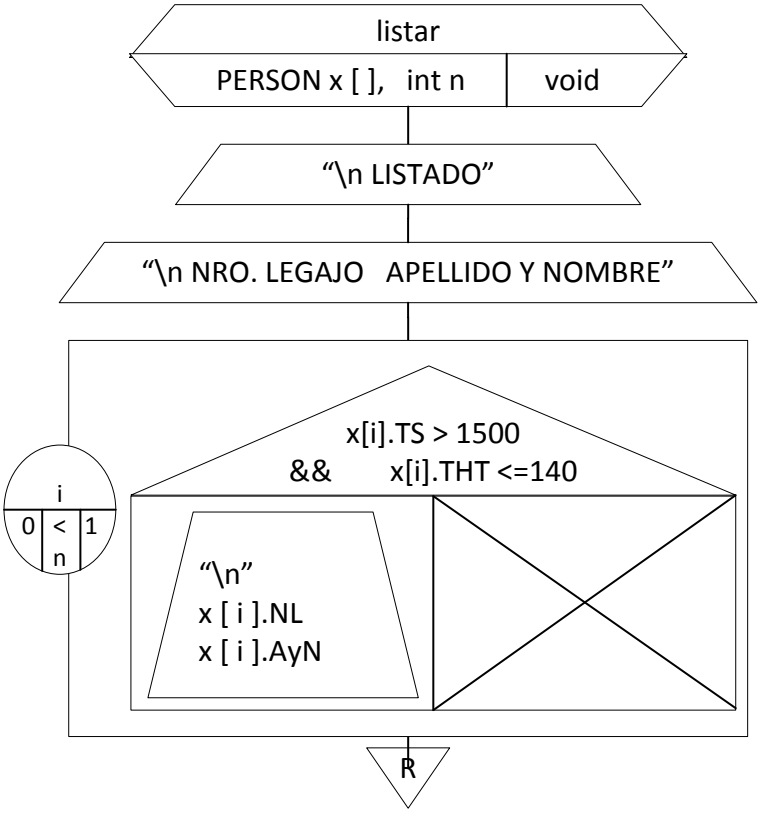



lectura_personal
PERSON x[ ], int n
int i
FILE *pf
PERSON aux
i = 0
pf = fopen("PERSONAL.dat", "rb")
pf
aux
!feof(pf) && i < n
x[i] = aux
i++
pf
aux
fclose(pf)
return(i)
R
leeControl
void
fecha f
f.d f.m f.a
control(f.d, 1, 30) == 0 || control(f.m, 1, 12) == 0 ||
control(f.a, 2011, 2013) == 0
return(f)
R


busqueda
PERSON x[], int dato, int n
int posi
posi = -1
i = 0
posi == -1 && i < n
x[i].NL == dato
posi = i
i++
return(posi)
R


grabar
PERSON x[], int n
void
FILE *pf
pf = fopen("PERSONAL.dat", "wb")
i
0
<
n
1
pf
x[i]
fwrite(...) GRABAR
fclose(pf)
R


typedef struct {
int leg;
int cantHora;
} info;
actualiza
PERSON x [ ], FECHA F, int n
void
info aux
FILE *ph
FILE *pe
ph = fopen("HORAS.dat", "rb")
pe = fopen("ERROR.dat", "wb")
ph
aux
!feof (ph)
pos = busqueda (x, aux.leg, n)
pos != -1
legant = aux.leg
sh = 0
legant == aux.leg && !feof (ph)
sh += aux.cantHora
ph
aux
pe
aux
ph
aux
x [pos].THT += sh
x [pos].TS += sh * x [pos].vh
x [pos].FE = f
fclose (pe)
fclose (ph)
R




```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
typedef struct{int d, m, a;} fecha;

typedef struct {int nl, tht;
                char ayn;
                float vh, ts;
                fecha fe;}person;

typedef struct {int leg, canthora;}info;

int lectura_personal(person[],int);
int control(int, int, int);
fecha lee_control(void);
int busqueda(person[], int, int);
void actualiza(person[], fecha, int);
void grabar(person[], int);
void listar(person[], int);

void main(void)
{
person arrayperso[175];
fecha fech;
int cant;

cant=lectura_personal(arrayperso,175);
fech=lee_control();
actualiza(arrayperso, fech, cant);
grabar(arrayperso,cant);
listar(arrayperso, cant);
getch();
}
```

```
int lectura_personal(person x[], int n)
{
FILE *pf;
person aux;
int i=0;
pf=fopen("PERSONAL.dat","rb");
if(pf==NULL)
    {
    printf("\n No se puede acceder al archivo");
    getch();
    exit(1);
    }
fread(&aux, sizeof(person),1,pf);
while(!feof(pf) && i<n)
    {
    x[i]=aux;
    i++;
    fread(&aux, sizeof(person),1,pf);
    }
fclose(pf);
return(i);
}

fecha lee_control(void)
{
fecha f;
do
{
    printf("\n ingresa fecha (dia - mes - anio)");
    scanf("%d-%d-%d", &f.d, &f.m, &f.a);
}while(control(f.d, 1, 30)==0 || control(f.m, 1, 12)==0 || control(f.a, 2011,
2016)==0);
return(f);
}

int control(int x, int li, int ls)
{
if(x<li || x>ls)
    return(0);
return(1);
}

void actualiza(person x[], fecha f, int n)
{
info aux;
FILE *ph, *pe;
int pos, legant, sh;
ph=fopen("HORAS.dat", "rb");
if(ph==NULL)
{
    printf("\n No se puede acceder al archivo");
    getch();
    exit(1);
}
pe=fopen("ERROR.dat", "wb");
if(pe==NULL)
{
    printf("\n No se puede acceder al archivo");
    getch();
    exit(1);
```

```
}
fread(&aux, sizeof(info),1,ph);
while(!feof(ph))
{
    pos=busqueda(x, aux.leg, n);
    if(pos!=-1)
    {
        legant=aux.leg;
        sh=0;
        while(legant==aux.leg && !feof(ph))
        {
            sh+=aux.canthora;
            fread(&aux, sizeof(info),1,ph);
        }
        x[pos].tht+=sh;
        x[pos].ts+=sh*x[pos].vh;
        x[pos].fe=f;
    }
    else
        {
        fwrite(&aux, sizeof(info),1,pe);
        fread(&aux, sizeof(info),1,pe);
        }
    }
fclose(pe);
fclose(ph);
}

int busqueda(person x[], int dato, int n)
{
int posi, i;
posi=-1;
i=0;
while(posi==-1 && i<n)
    {
    if(x[i].nl==dato)
        posi=i;
    else
        i++;
    }
return(posi);
}

void grabar(person x[], int n)
{
FILE *pf;
int i;
pf=fopen("PERSONAL.dat","wb");
if(pf==NULL)
    {
    printf("\n No se puede acceder al archivo");
    getch();
    exit(1);
    }
for(i=0; i<n; i++)
    fwrite(&x[i], sizeof(person), 1, pf);
fclose(pf);
}

void listar(person x[], int n)
```

```
{
int i;
printf("\n LISTADO");
printf("\n NRO. Legajo    APELLIDO Y NOMBRE");
for(i=0; i<n; i++)
   if(x[i].ts>1500 && x[i].tht<=140)
       printf("\n %d %s", x[i].nl, x[i].ayn);
}
```

3. Impuesto Municipal

La municipalidad de la ROTONDA desea efectuar un control de la cobranza del impuesto Municipal en cada uno de los 6 bimestres de cada año. Existen como máximo 15200 contribuyentes.

Se dispone del archivo **CONTRIBUYENTE.dat**, conteniendo:

- **Nro. de Contribuyente** (Nro. No correlativo de 4 cifras)
- **Apellido Y Nombre** (60 caracteres)
- **Domicilio** (40 caracteres)

Se dispone también del archivo, secuencial, llamado **PAGOS.dat**, con un registro por cada cobro efectuado, con los siguientes datos:

- **Nro. contribuyente (Nro. No correlativo de 4 cifras)**
- **Año del impuesto pagado**
- **Bimestre pagado**
- **Importe pagado**

Este archivo se encuentra **ordenado por Año**

Si el contribuyente NO existe en el archivo CONTRIBUYENTE, grabar en el archivo **ERROR.dat** el registro leído en PAGOS:

Confeccionar un programa para determinar e informar:

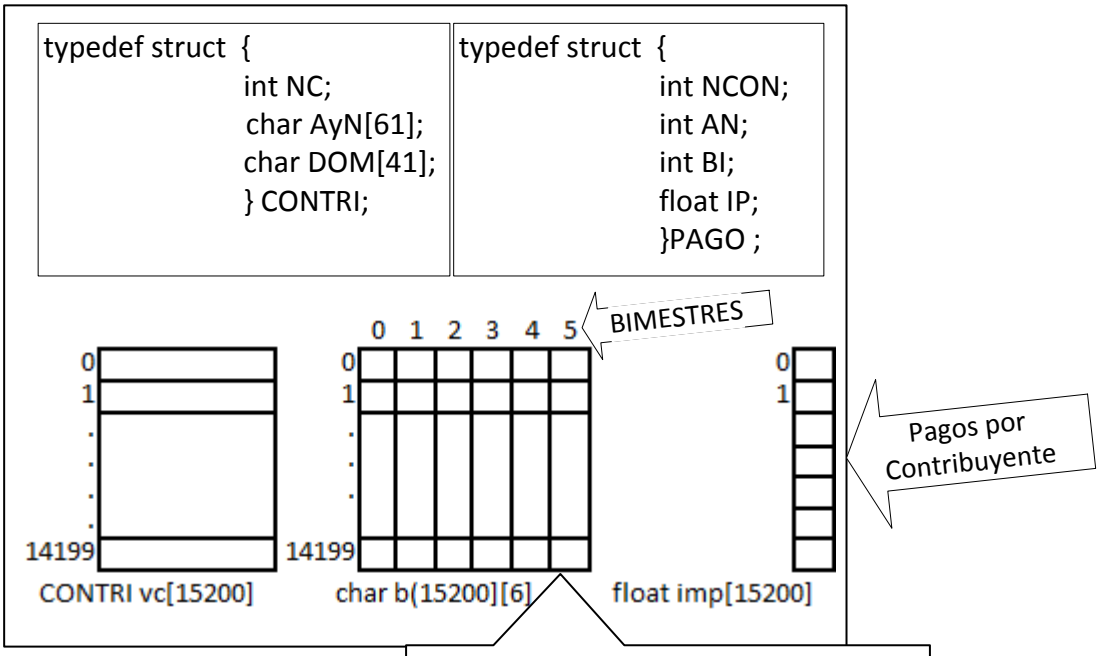
a) El importe total recaudado en cada año.

b) Las deudas de cada contribuyente, según el siguiente formato de impresión:

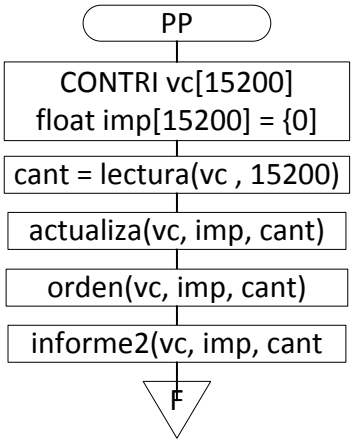
```
          DEUDAS REGISTRADAS DEL AÑO: xxxx
CONTRIBUYENTE     BIMESTRES 1-2-3-4-5-6
    xxxx                    x     x
```

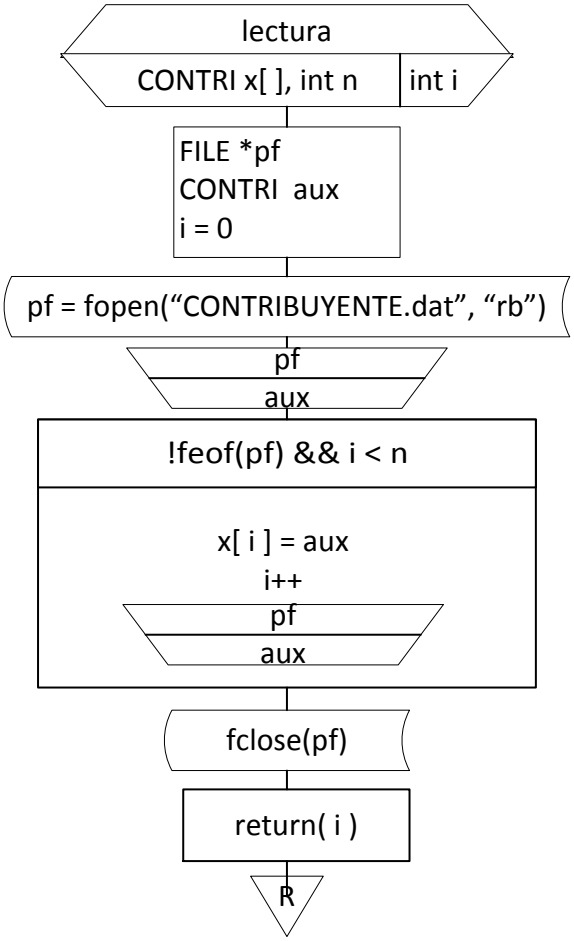
c) Importe total pagado por contribuyente, ordenado por importe total pagado, indicando:

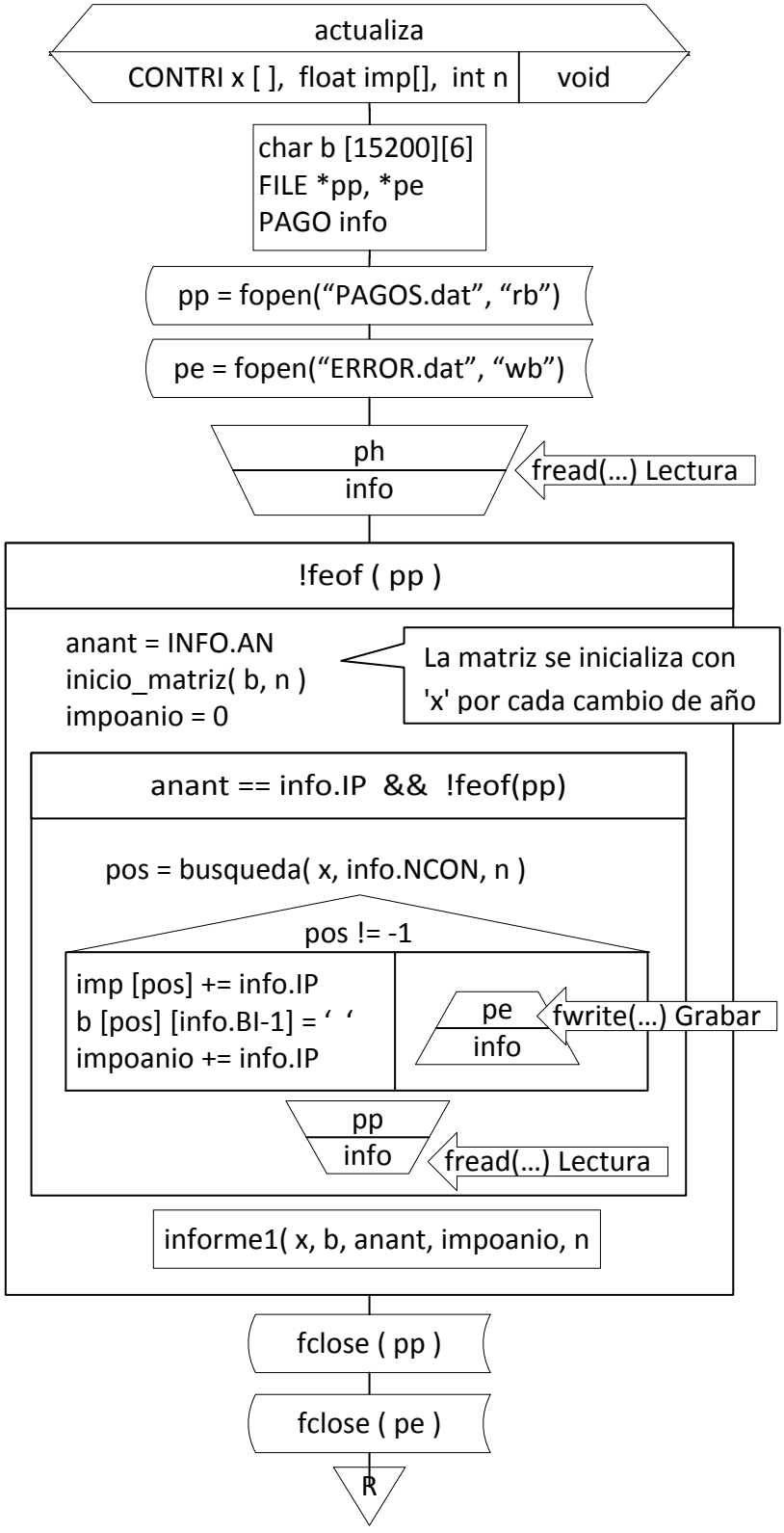
```
Nro. de Contribuyente        Apellido Y Nombre       Importe Total Pagado
```

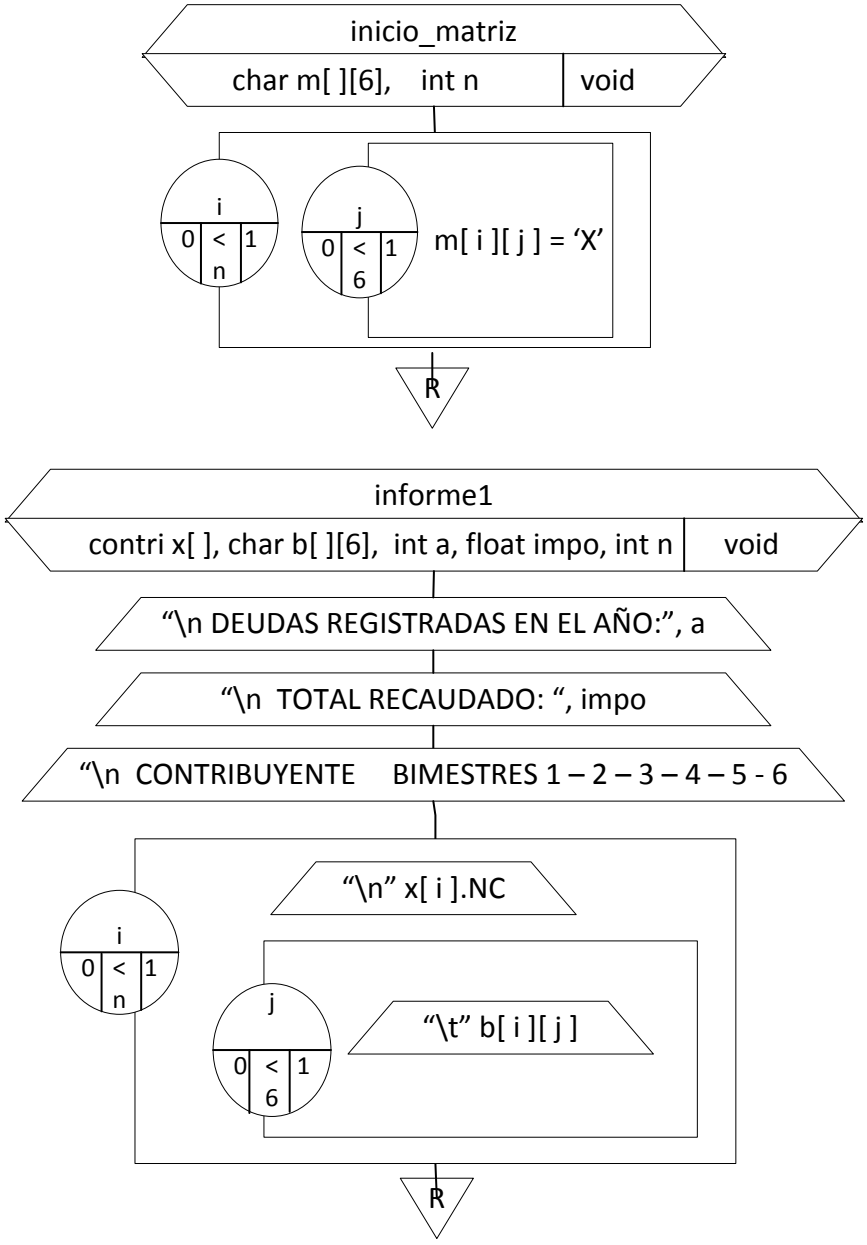


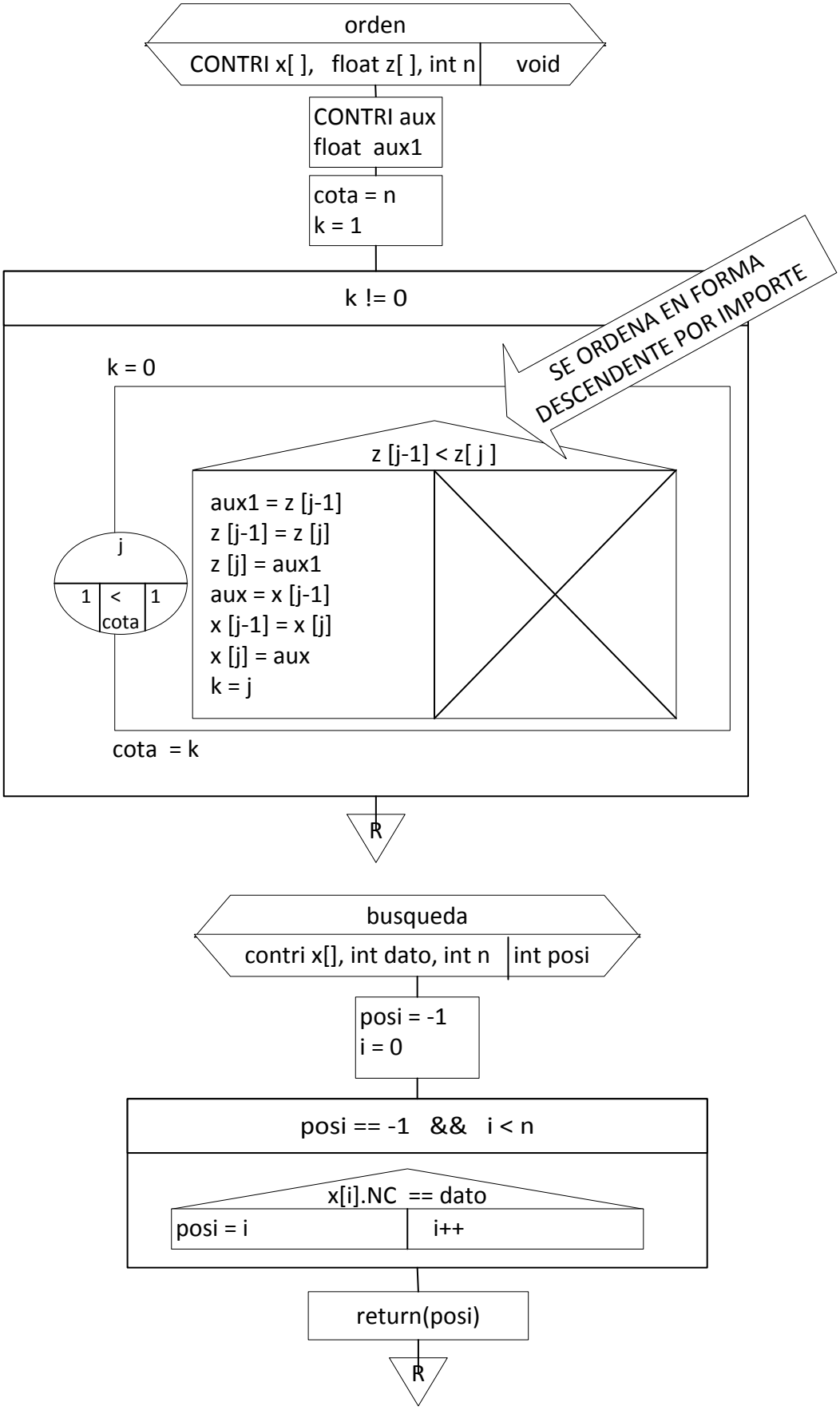
Por cada inicio de año se resetea con 'x' (no pagó). Cuando paga se borra la 'x' y se reemplaza por ' '.
Se usa en función actualiza.

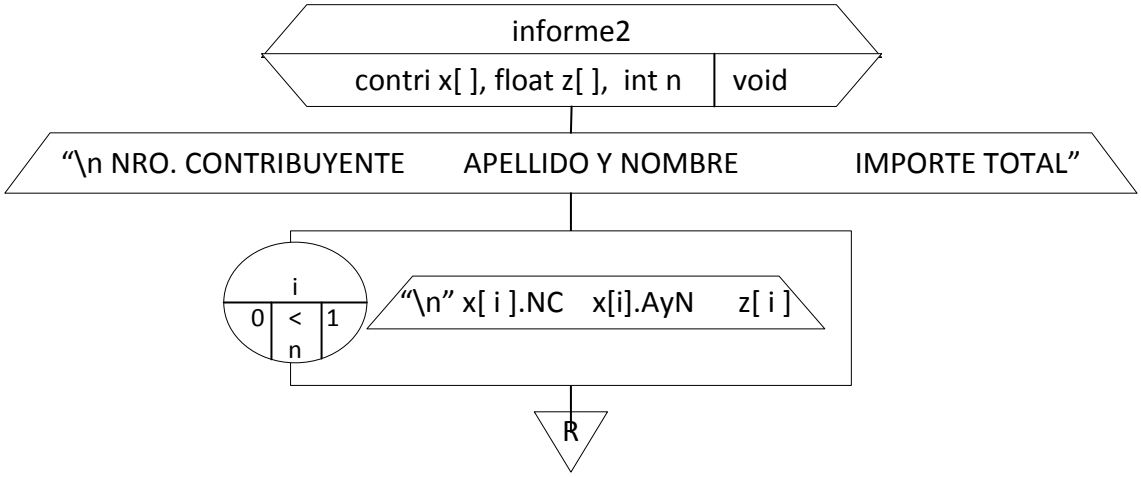



lectura
CONTRI x[ ], int n
int i
FILE *pf
CONTRI aux
i = 0
pf = fopen("CONTRIBUYENTE.dat", "rb")
pf
aux
!feof(pf) && i < n
x[i] = aux
i++
pf
aux
fclose(pf)
return(i)
R


actualiza
CONTRI x [ ], float imp[], int n
void
char b [15200][6]
FILE *pp, *pe
PAGO info
pp = fopen("PAGOS.dat", "rb")
pe = fopen("ERROR.dat", "wb")
ph
info
fread(...) Lectura
!feof (pp)
anant = INFO.AN
inicio_matriz(b, n)
impoanio = 0
La matriz se inicializa con 'x' por cada cambio de año
anant == info.IP && !feof(pp)
pos = busqueda(x, info.NCON, n)
pos != -1
imp [pos] += info.IP
b [pos] [info.BI-1] = ' '
impoanio += info.IP
pe
info
fwrite(...) Grabar
pp
info
fread(...) Lectura
informe1(x, b, anant, impoanio, n
fclose (pp)
fclose (pe)
R


inicio_matriz
char m[ ][6], int n
void
i
0 < 1
n
j
0 < 1
6
m[i][j] = 'X'
R
informe1
contri x[ ], char b[ ][6], int a, float impo, int n
void
"\n DEUDAS REGISTRADAS EN EL AÑO:", a
"\n TOTAL RECAUDADO: ", impo
"\n CONTRIBUYENTE BIMESTRES 1 – 2 – 3 – 4 – 5 - 6
"\n" x[i].NC
i
0 < 1
n
j
0 < 1
6
"\t" b[i][j]
R


orden
CONTRI x[ ], float z[ ], int n
void
CONTRI aux
float aux1
cota = n
k = 1
k != 0
SE ORDENA EN FORMA DESCENDENTE POR IMPORTE
k = 0
z [j-1] < z[j]
aux1 = z [j-1]
z [j-1] = z [j]
z [j] = aux1
aux = x [j-1]
x [j-1] = x [j]
x [j] = aux
k = j
j
1 < cota 1
cota = k
R
busqueda
contri x[], int dato, int n
int posi
posi = -1
i = 0
posi == -1 && i < n
x[i].NC == dato
posi = i
i++
return(posi)
R




```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
typedef struct {int NC;
                        char AYN[61];
                        char DOM [41];
                        } CONTRI;

typedef struct {int NCON;
                        int AN;
                        int BI;
                        float IP;
                        } PAGO;

int lectura (CONTRI[], int);
void inicio_matriz (char[][6], int);
int busqueda (CONTRI[], int, int);
void infome1 (CONTRI[], char[][6],int, float[], int);
void actualiza (CONTRI[], float[], int);
void orden (CONTRI[], float[], int);
void informe2(CONTRI[], float[],int);

void main(void)
{
    CONTRI  vc[15200];
    float imp[15200]={0};
    int cant, pos;
    cant=lectura (vc, 15200);
    actualiza(vc,imp,cant);
    orden(vc, imp, cant);
    informe2(vc, imp, cant);
    getch();
}

int lectura (CONTRI x[], int n)
{
    FILE *pf;
    CONTRI aux;
    int i=0;
    pf=fopen("CONTRIBUYENTE.dat","rb");
    if (pf==NULL)
        {
        printf ("\n No SE PUEDE ACCEDER");
```

```
        getch();
        exit(1);
        }
    fread(&aux, sizeof(CONTRI),1,pf);
    while (!feof(pf)&&i<n)
             {
        x[i]=aux;
             i++;
             fread(&aux, sizeof(CONTRI),1,pf);
             }
    fclose (pf);
    return (i);
}

void inicio_matriz(char m[][6], int n)
{
    int i, j;
    for (i=0; i>n; i++)
    for (j=0; j<6; j++)
        m[i][j]='X';
}

void informe1(CONTRI x[],char b[][6],int a,float impo, int n)
{
    int i,j;
    printf ("\n Deudas registradas en el año: %d",a);
    printf ("\n Total recaudado: %.2f", impo);
    printf ("\n CONTRIbuyente Bimestre 1-2-3-4-5-6");
    for (i=0; i<n; i++)
        {
        printf("\n %d", x[i].NC);
        for(j=0;j<6;j++)
            printf("\t%c",b[i][j]);
        }
}

void actualiza(CONTRI x[], float imp[], int n)
{
    char b[15200][6];
    FILE *pp,*pe;
    PAGO info;
    int anant, pos;
    float impoanio;
    pp=fopen ("PAGOS.dat","rb");
    if (pp==NULL)
        {
        printf("\n NO SE PUEDE ACCEDER");
        getch();
        exit(1);
        }
    pe=fopen("ERROR.dat","wb");
    if (pe==NULL)
    {
    printf("\n NO SE PUEDE ACCEDER");
    getch();
    exit(1);
    }
    fread(&info, sizeof(PAGO),1,pp);
    while (!feof(pp))
        {
```

```
        anant=info.AN;
        inicio_matriz(b,n);
        impoanio=0;
        while (anant==info.AN && !feof(pp))
                    {
            pos=busqueda(x, info.NCON, n);
                    if(pos!=-1)
                            {
                imp[pos]+=info.IP;
                            b[pos][info.BI-1]=' ';
                            impoanio+=info.IP;
                            }
                        else
                    fwrite(&info, sizeof(PAGO),1,pe);
                    fread(&info, sizeof(PAGO),1,pp);
                        }
        informe1(x, b,anant, impoanio, n);
        }
    fclose(pp);
    fclose(pe);
}
void informe2(CONTRI x[], float z[],int n)
{
    int i;
    printf ("\n NRO. CONTRIBUYENTE APELLIDO Y NOMBRE   IMP. TOTAL PAGADO");
    for (i=0; i<n; i++)
        printf ("\n %d %s %.2f", x[i].NC, x[i].AYN, z[i]);
}
int busqueda(CONTRI x[], int dato, int n)
{
    int posi, i;
    posi=-1;
    i=0;
    while(posi==-1 && i<n)
        if(x[i].NC==dato)
            posi=1;
    else
        i++;
    return(posi);
}
void orden(CONTRI x[],float z[], int n)
{
    CONTRI aux;
    float aux1;
    int cota, k, j;
    cota=n;
    k=1;
    while(k!=0)
    {
        k=0;
        for(j=1; j<cota; j++)
            if(z[j-1]<z[j])
        {
            aux1=z[j-1];
            z[j-1]=z[j];
            z[j]=aux1;
            aux=x[j-1];
            x[j-1]=x[j];
            x[j]=aux;
            k=j;
```

```
            }
            cota=k;
        }
    }
```