



Elementos de Programación

UNIDAD 10. Archivos - Anexo Ejercicios Resueltos

INDICE

1.	NOTAS DE ALUMNOS	2
2.	SALDOS.....	6
3.	EXPENSAS	15
4.	ALQUILER DE AUTOS	22

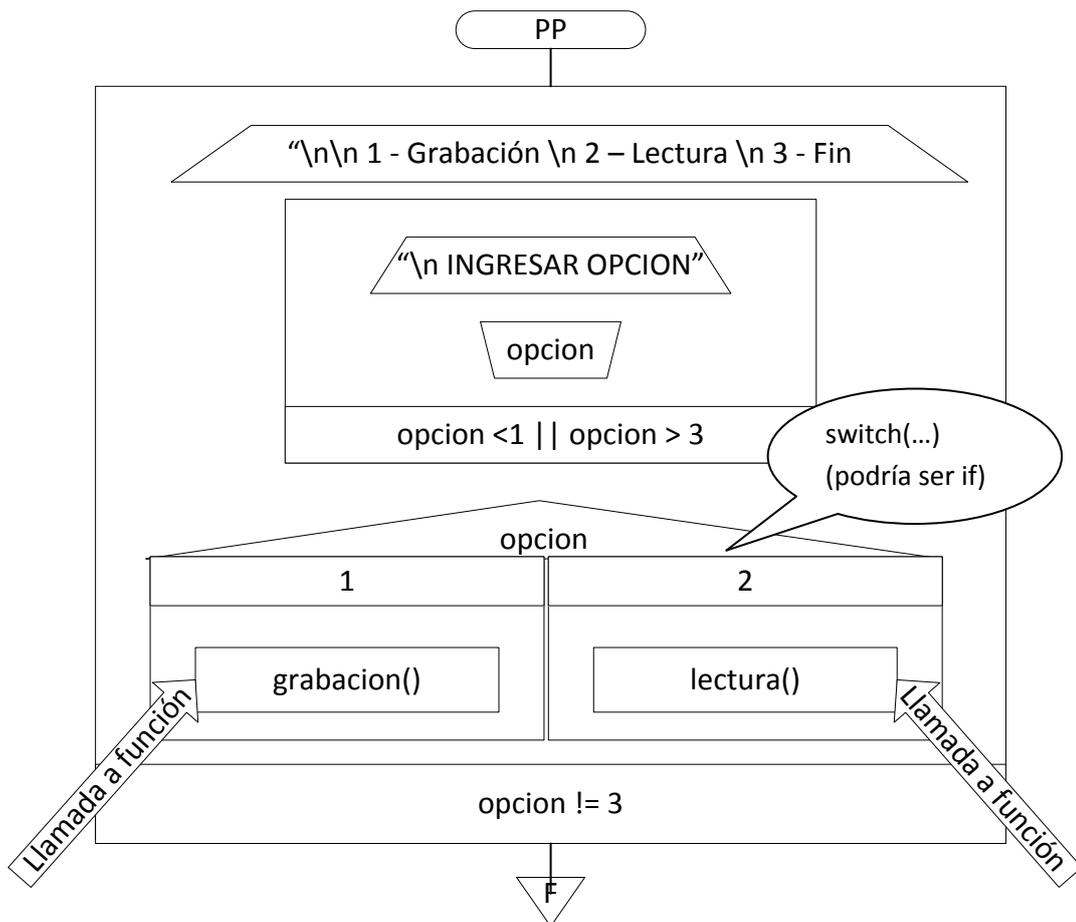
UNIDAD 10 – Archivos: Ejercicios Resueltos

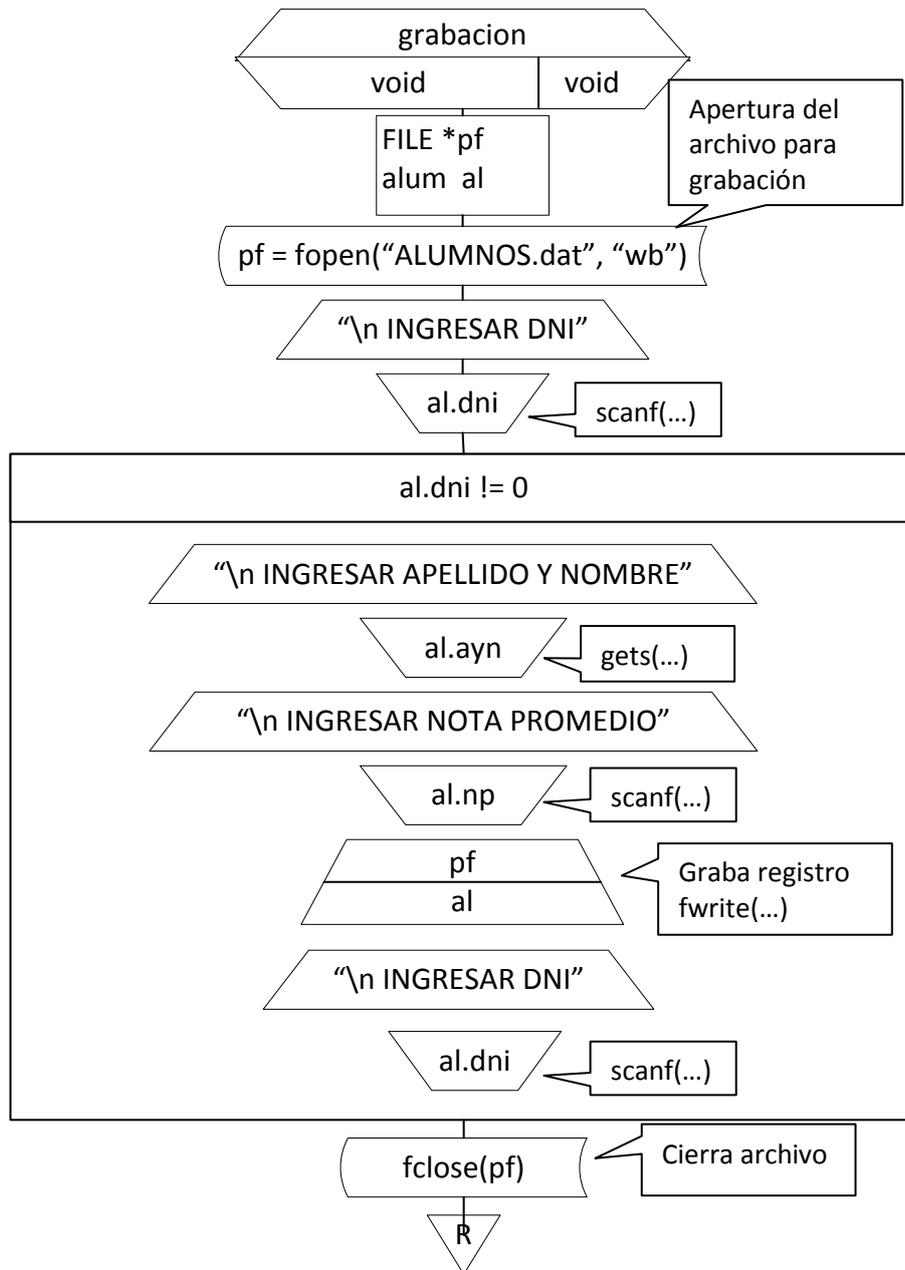
1. Notas de alumnos

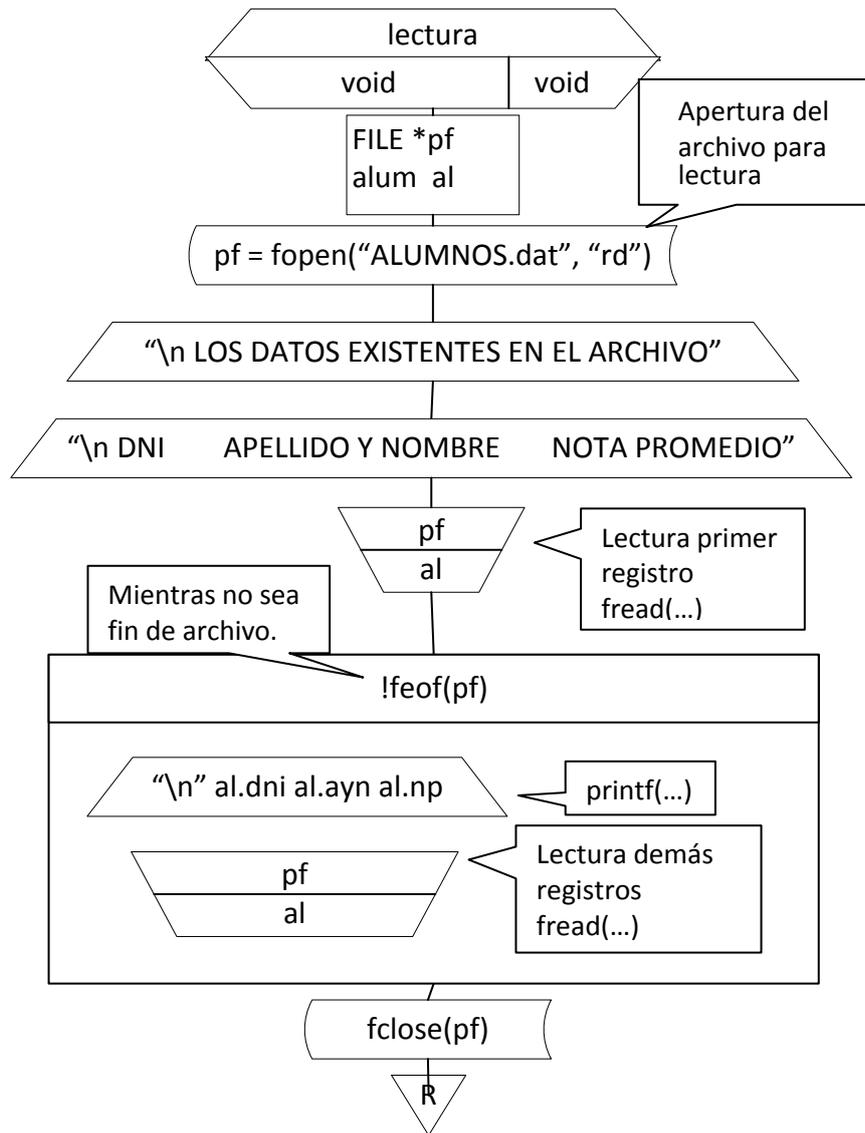
Se ingresa DNI, apellido y nombre y nota promedio de alumnos de un curso. Dicha información termina con DNI igual a cero.

Se pide:

- Grabar toda la información en el archivo **alumno.dat**
- Leer la información grabada en el archivo.







```
#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:

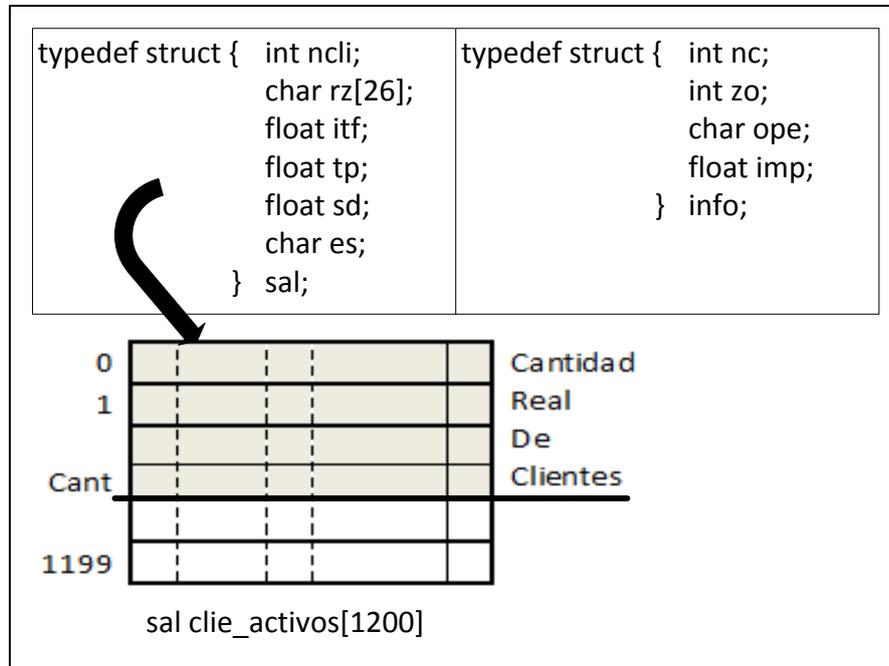
Confeccionar el diagrama de lógica y la respectiva codificación para:

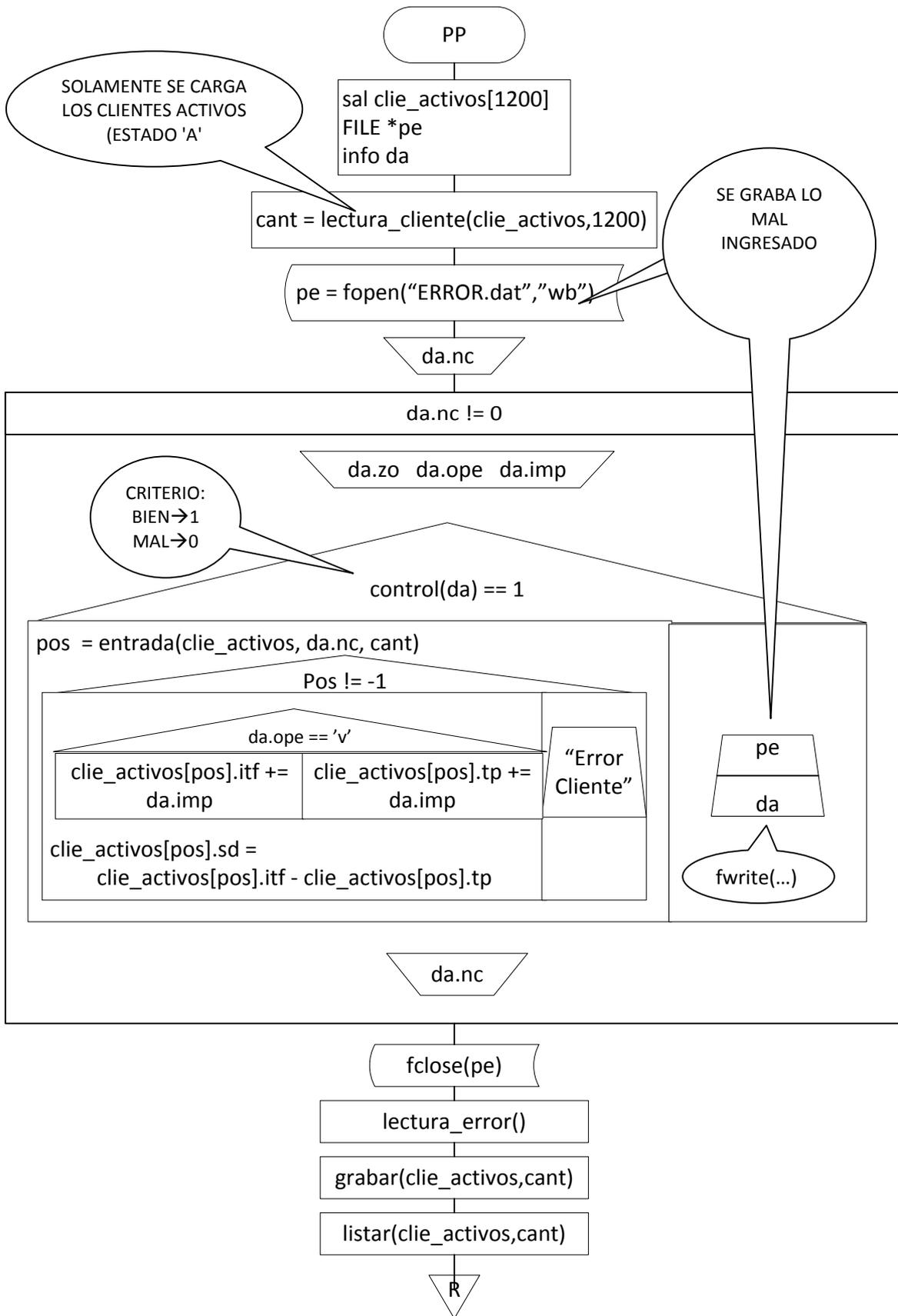
- Generar el array de estructuras llamado **Clie_Activos**, con solamente los datos de los clientes activos (Código de Estado 'A') del archivo SALDOS. Utilizar función **LECTURA_CLIENTE**.
- Ingresar y controlar las operaciones del día. Para controlar la validez se debe confeccionar una función **CONTROL** 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.
- 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.
- Informar los datos grabados en el archivo **ERROR.dat**. Mediante la función **LECTURA_ERROR**.
- Actualizar el archivo SALDOS. Mediante la función **GRABAR**.

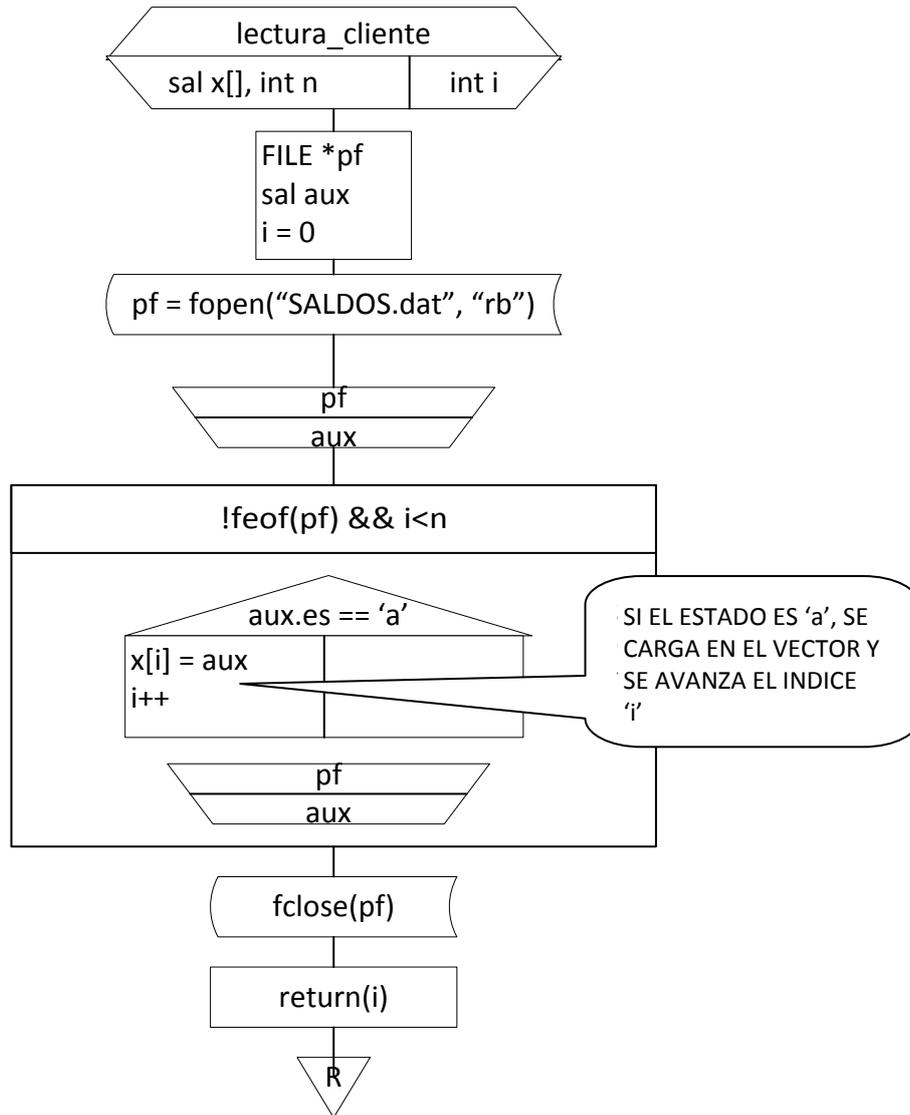
- f) Al finalizar el proceso, confeccionar una lista con aquellos clientes, que han quedado con saldo deudor mayor de 1500 pesos. Mediante la función **LISTAR**.

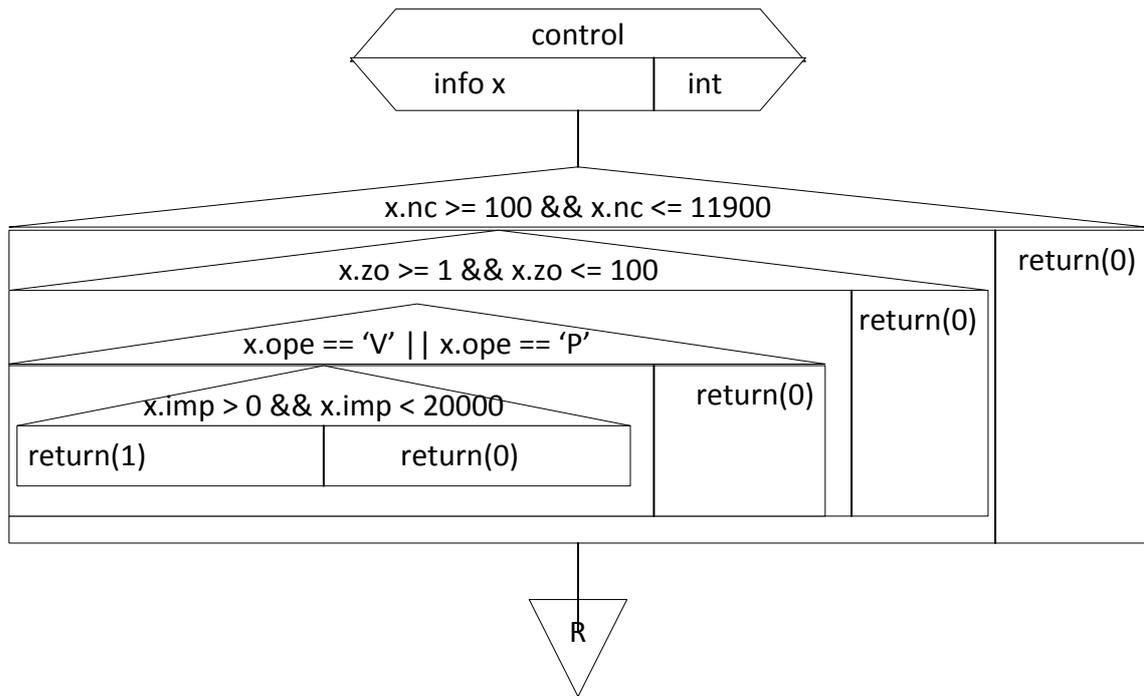
```

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

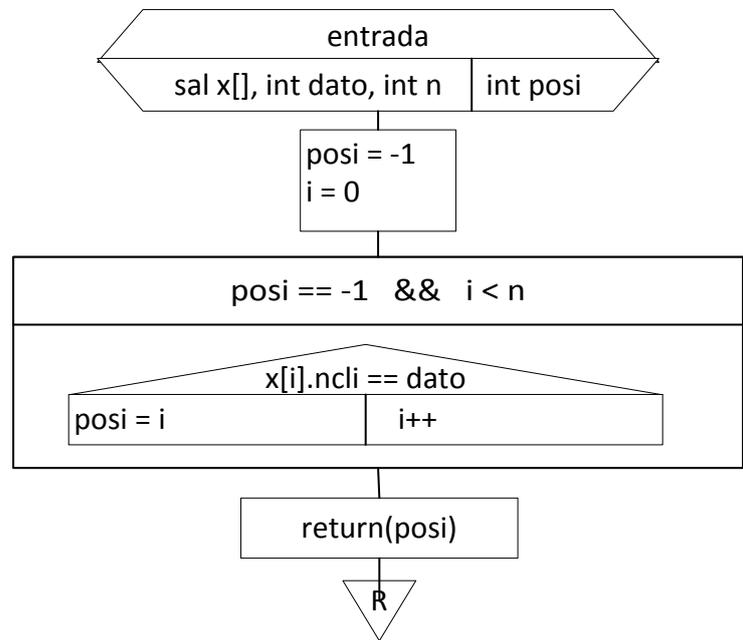


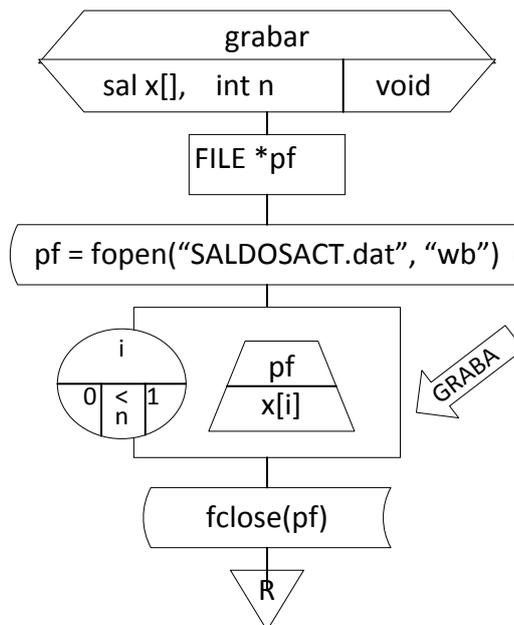
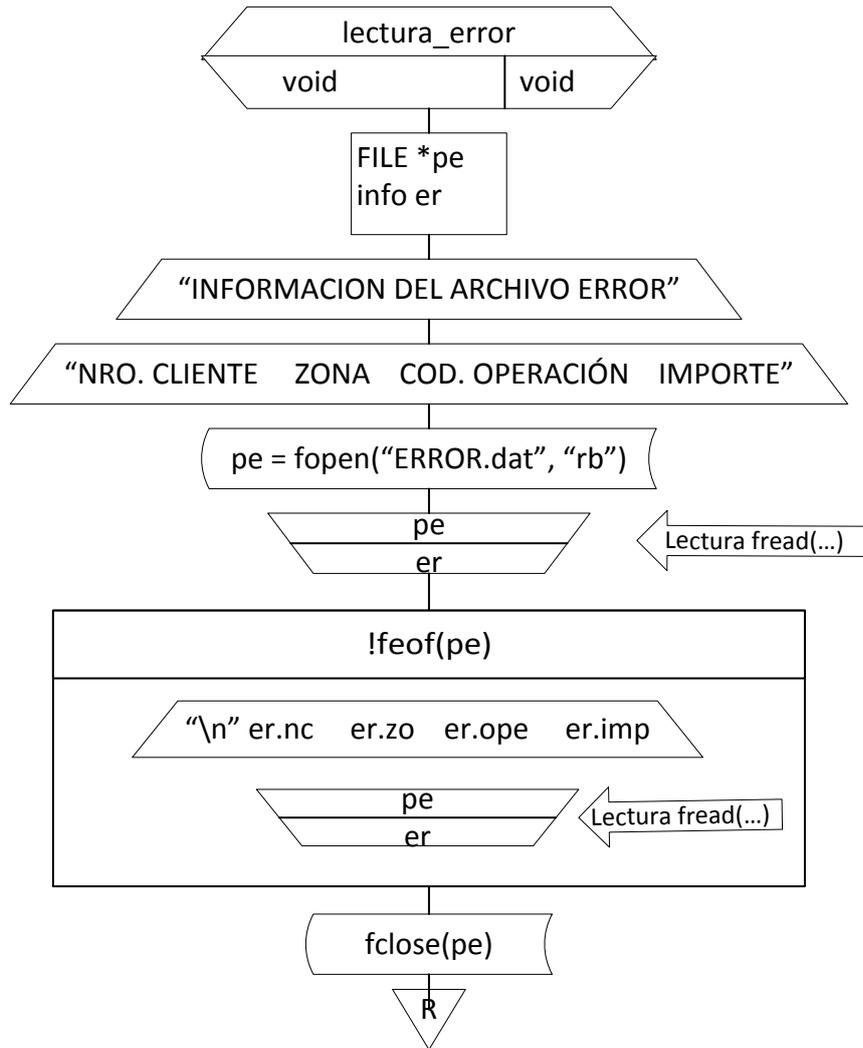


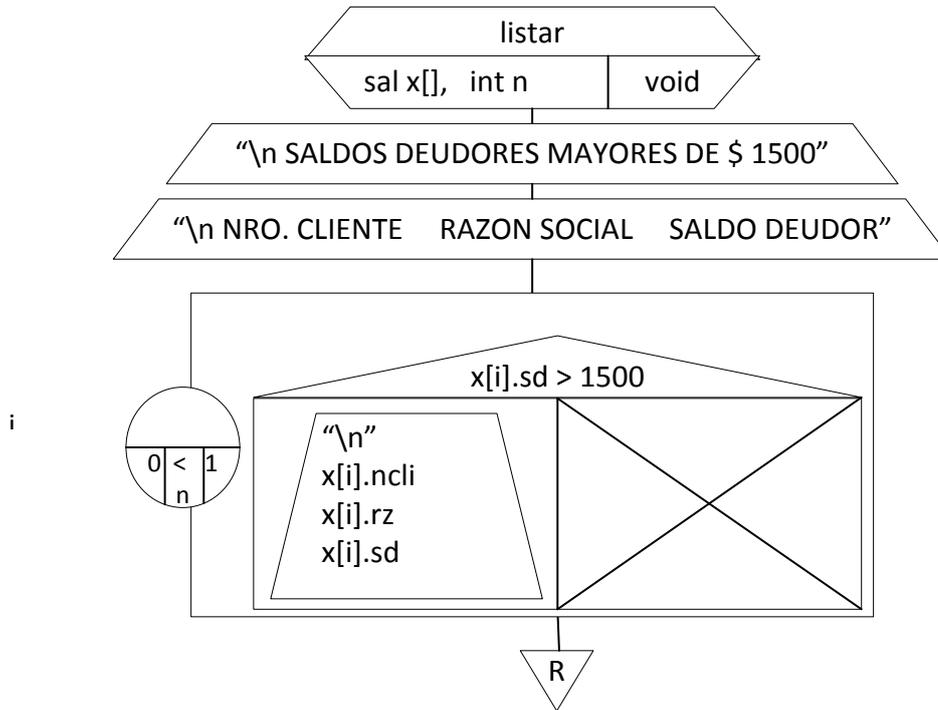




SI TODOS LOS DATOS SE INGRESARON CORRECTAMENTE, LA FUNCIÓN RETORNA 1, SI ALGUNO SE INGRESO MAL, LA FUNCIÓN RETORNA 0.







```

#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:

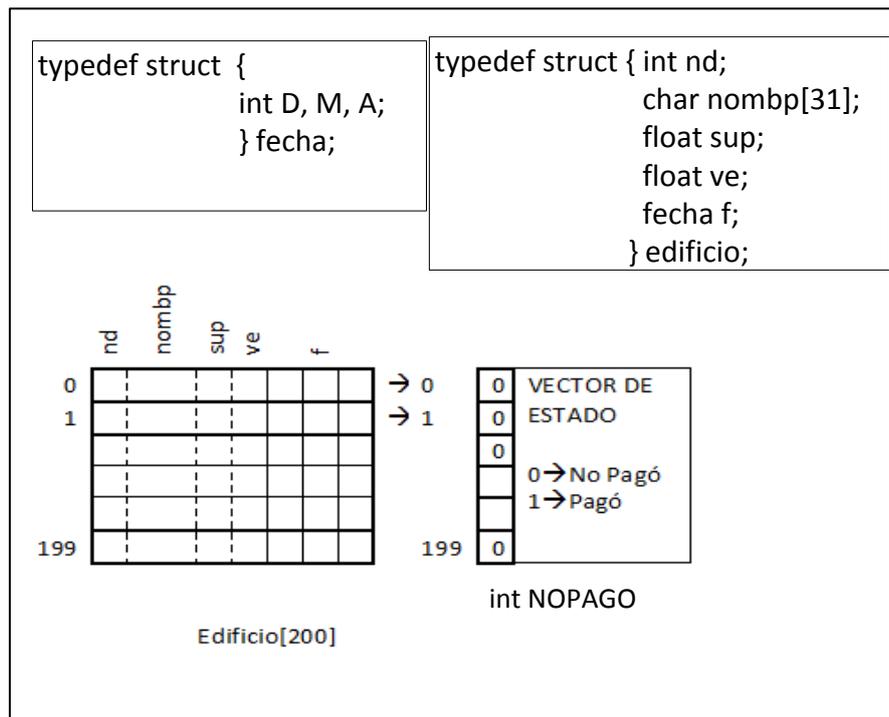
- a) Se dispone de un archivo, solamente con la información de los departamentos vendidos llamado **EDIFICIO.dat**, cuyo diseño es:
 - Nro. de departamento (numérico NO correlativo de 3 cifras)
 - Nombre del propietario (máximo 30 caracteres)
 - Superficie en metros cuadrados (xxx.xx)
 - Valor expensas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

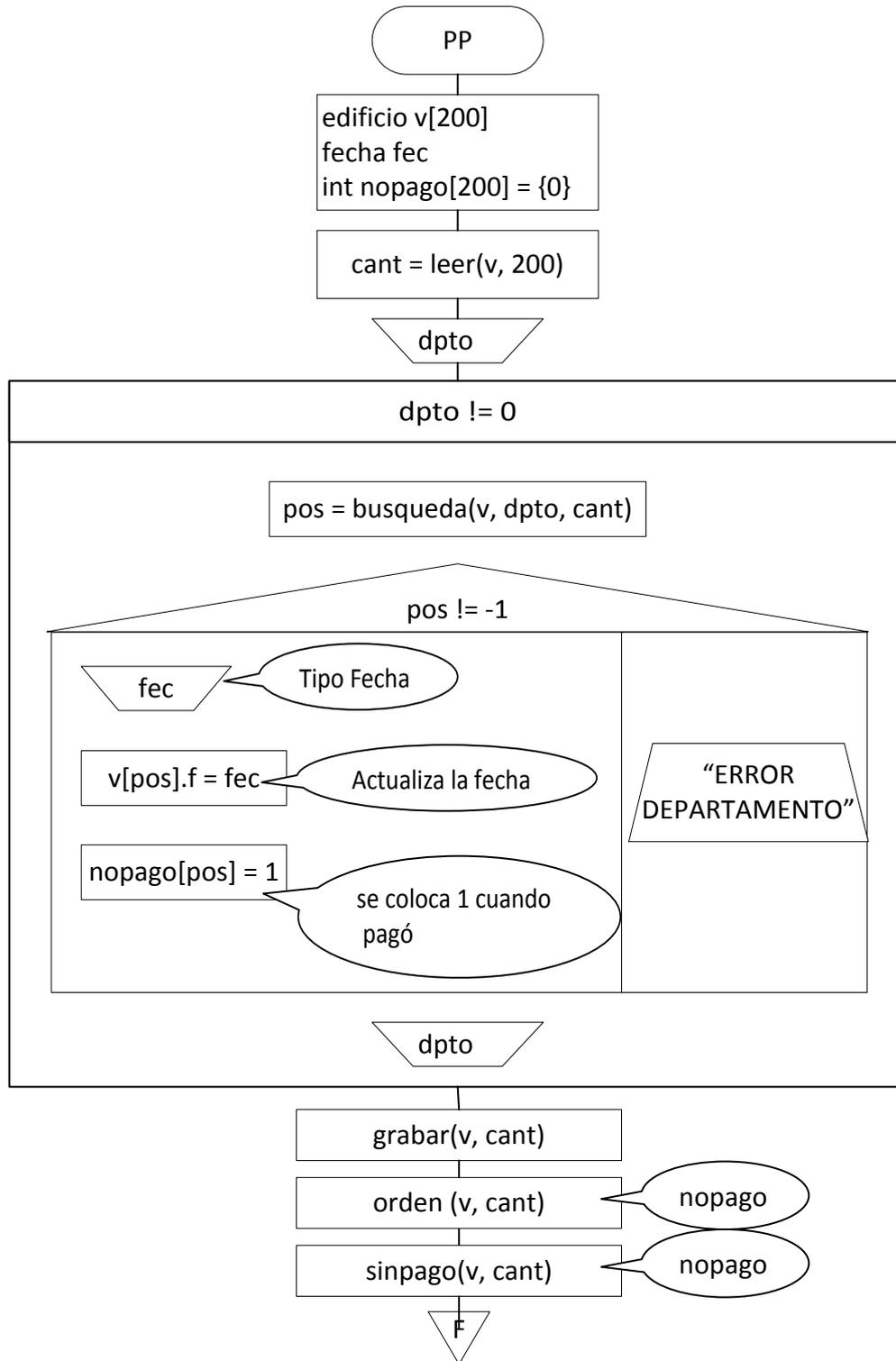
- e) Confeccionar una función llamada **SINPAGO**, con parámetros que reciba la información necesaria para determinar el siguiente listado con los morosos (no han pagado), ordenado en forma descendente por valor expensa.

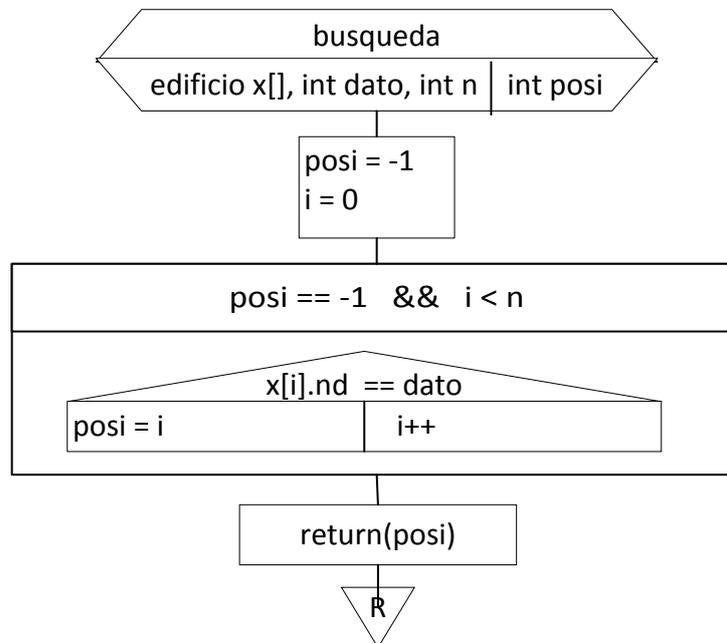
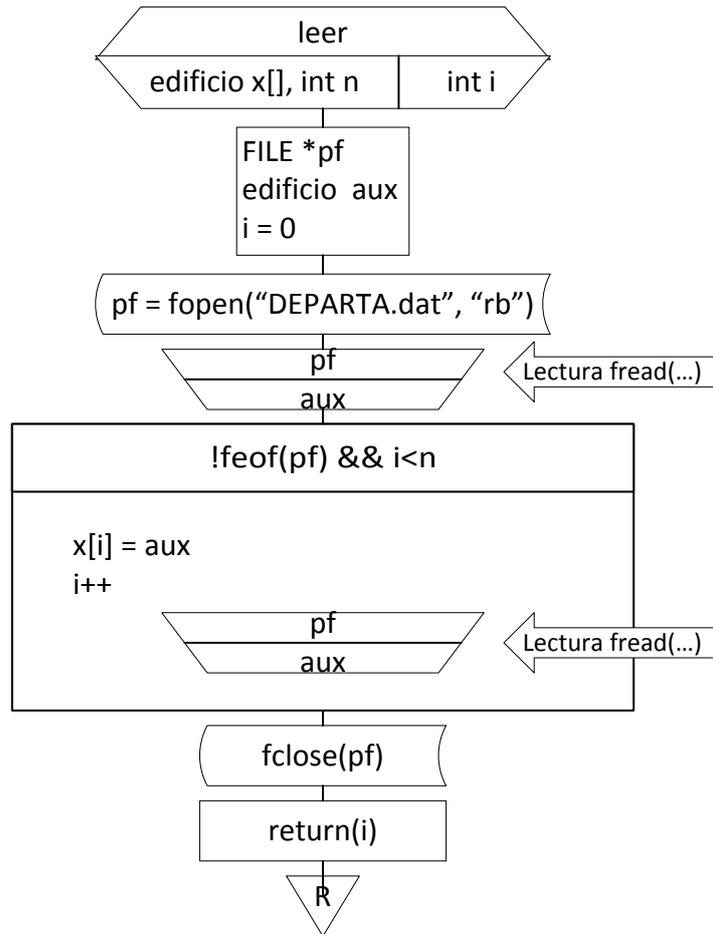
```

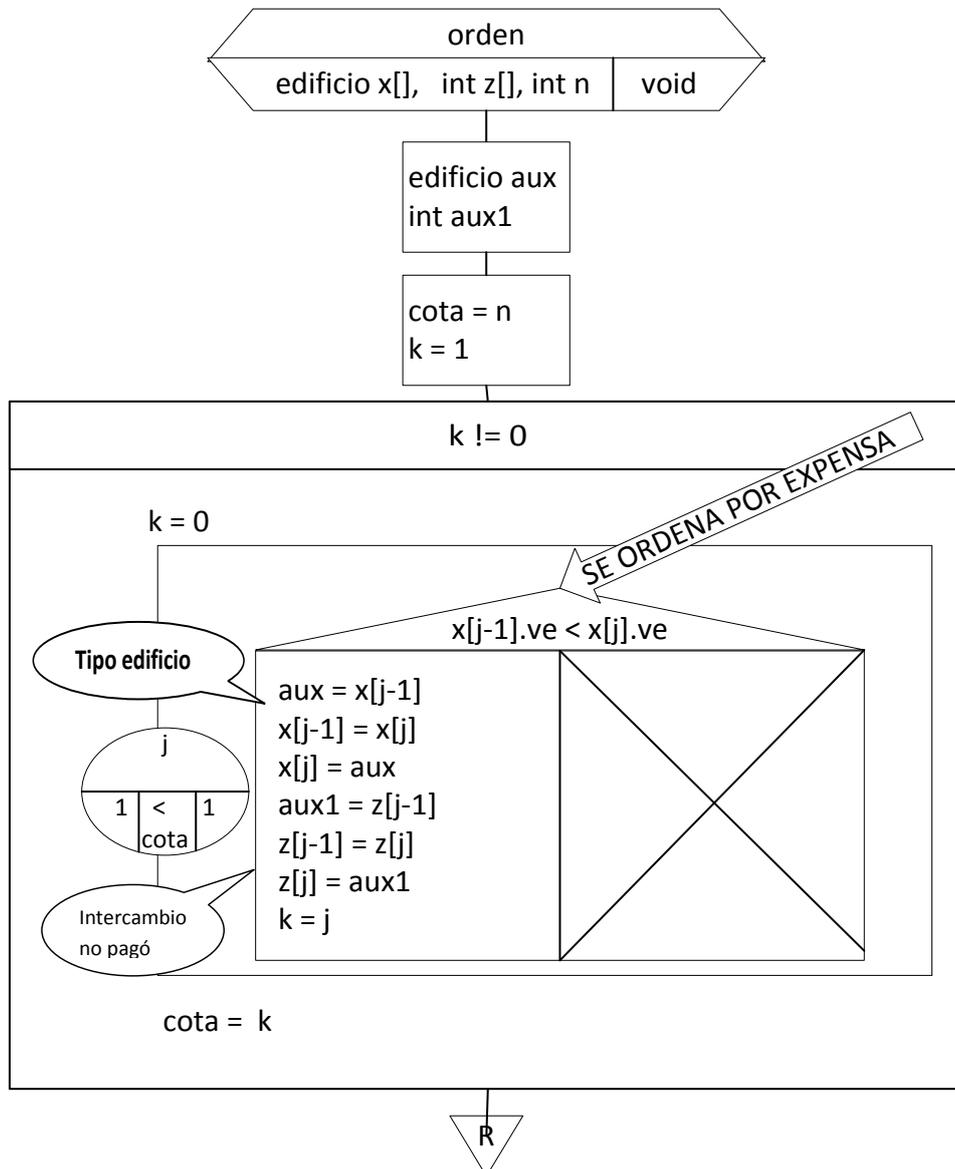
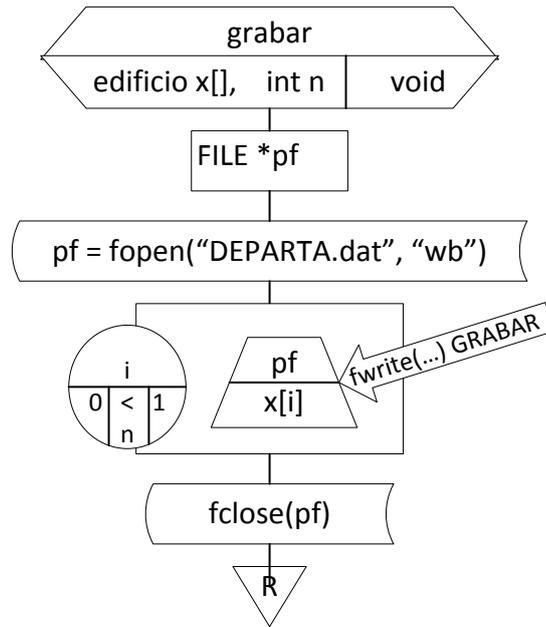
EXPENSAS IMPAGAS
DEPARTAMENTO      PROPIETARIO      VALOR EXPENSA
XXX              XXXXXXXXXXXXX    XXXX.XX
    
```

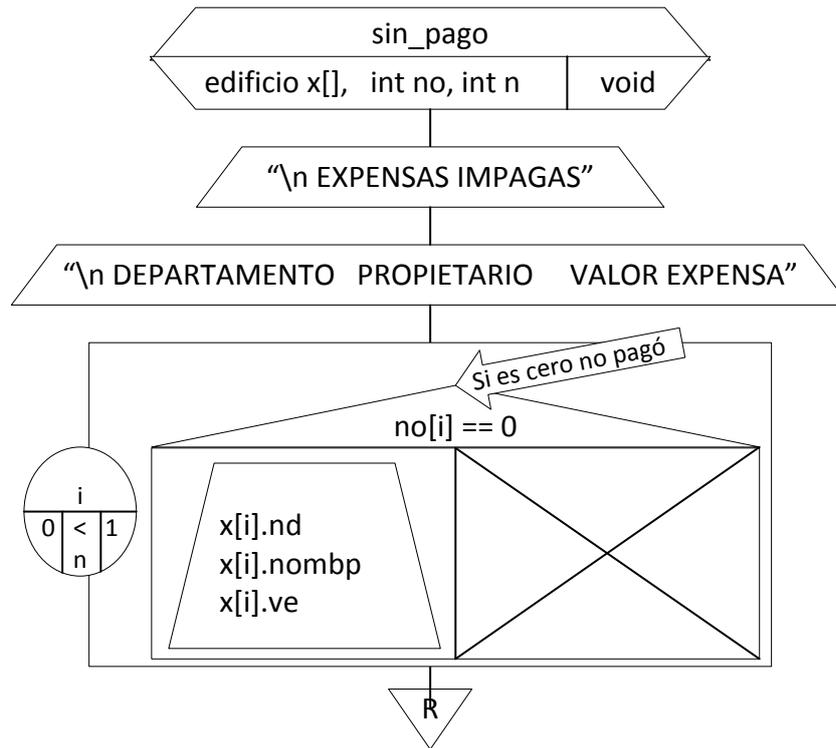
- f) Determinar el porcentaje de departamentos que hayan pagado la expensa sobre el total de departamentos vendidos utilizando la función **PORCEN**.











```
#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=i;
    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
XXXXXXXXX              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
CJU236           x    x
AHP888           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;
```


		DÍA				
		1	2	3	...	30
		0	1	2	...	29
0	np ayn	X		X		X
1			X	X	X	X
		X	X	X		
cant		X		X		
		X			X	
49						X

propi a[50]

char marca [50][30]
 Esta matriz se inicializa con 'X' y cuando realiza un viaje se asigna un espacio ' '

float rec[30]

