Universidad Nacional de la Matanza

Elementos de Programación

UNIDAD 11. Corte de Control

Anexo Ejercicios Resueltos

INDICE

UNIDAD 11
CORTE DE CONTROL – Ejercicios Resueltos

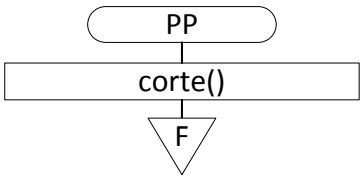
1. Ventas

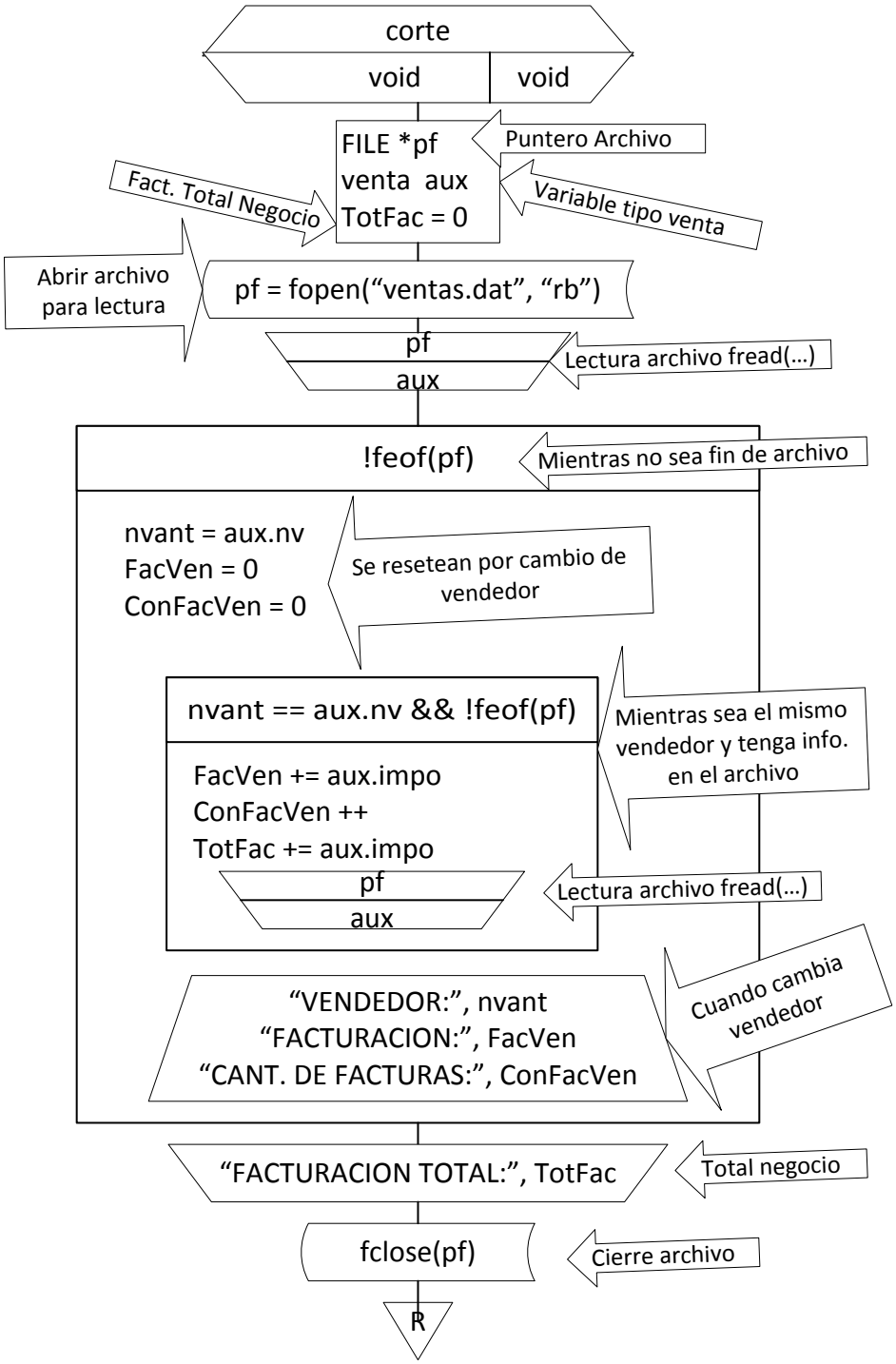
Se dispone de un archivo llamado ventas.dat que contiene la información de las ventas realizadas por la empresa a lo largo del mes. El archivo se encuentra ordenado por número de vendedor y tiene la siguiente estructura:

- 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;
```





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.

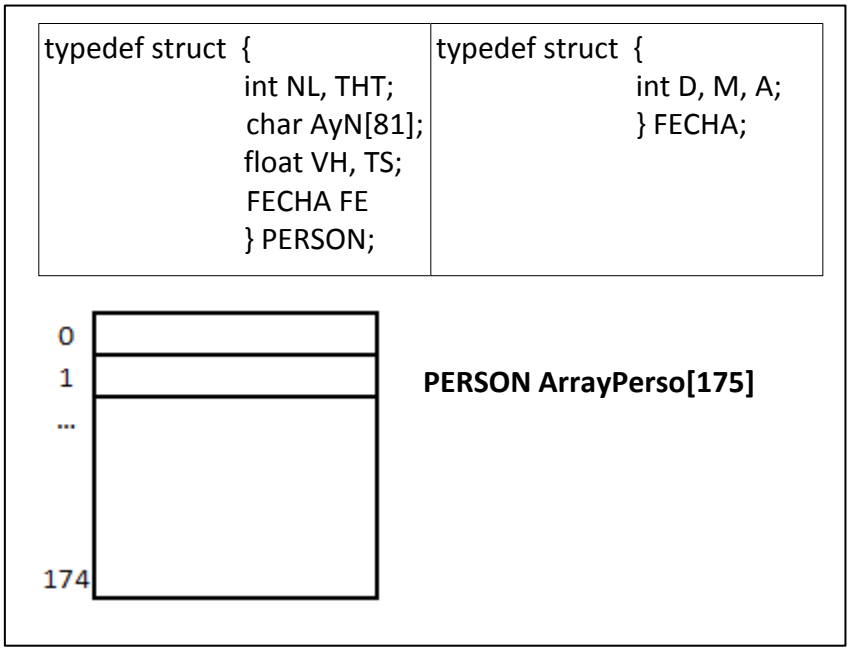
c) También se dispone del archivo **HORAS.dat** donde por cada día trabajado se registró.
   - Nro. de Legajo (Nro. entero de 3 cifras NO correlativo)
   - Cantidad de horas trabajadas

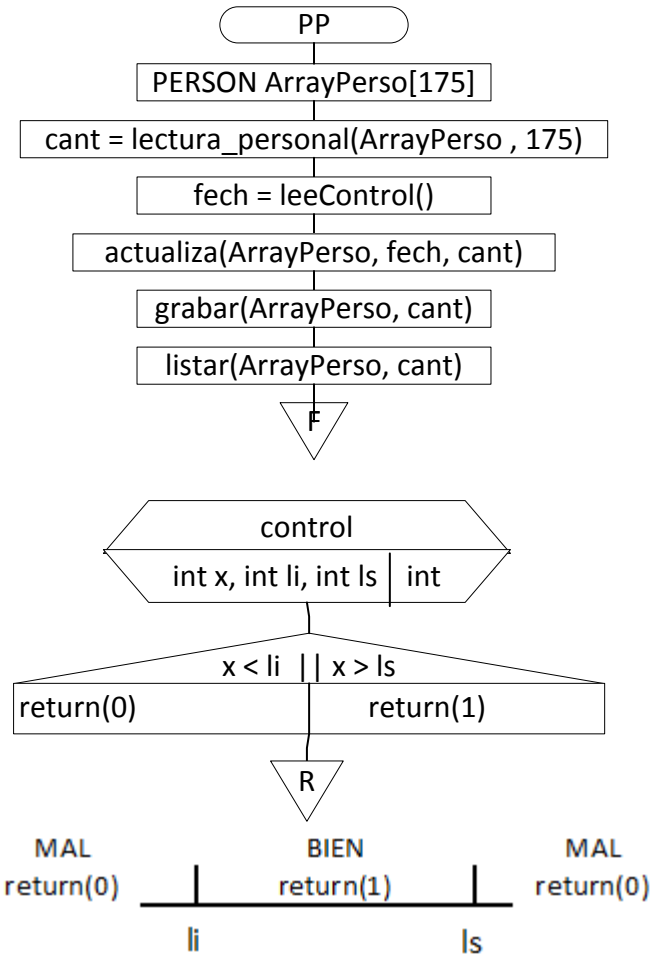
   Estos datos se encuentran en el archivo ordenados por legajo.
   Para el control de la existencia del Nro. de Legajo Confeccionar la función **BUSQUEDA** con los argumentos necesarios. Si no existe, grabar la información ingresada (Nro. de legajo – cantidad de horas trabajadas) en el archivo **ERROR.dat**.

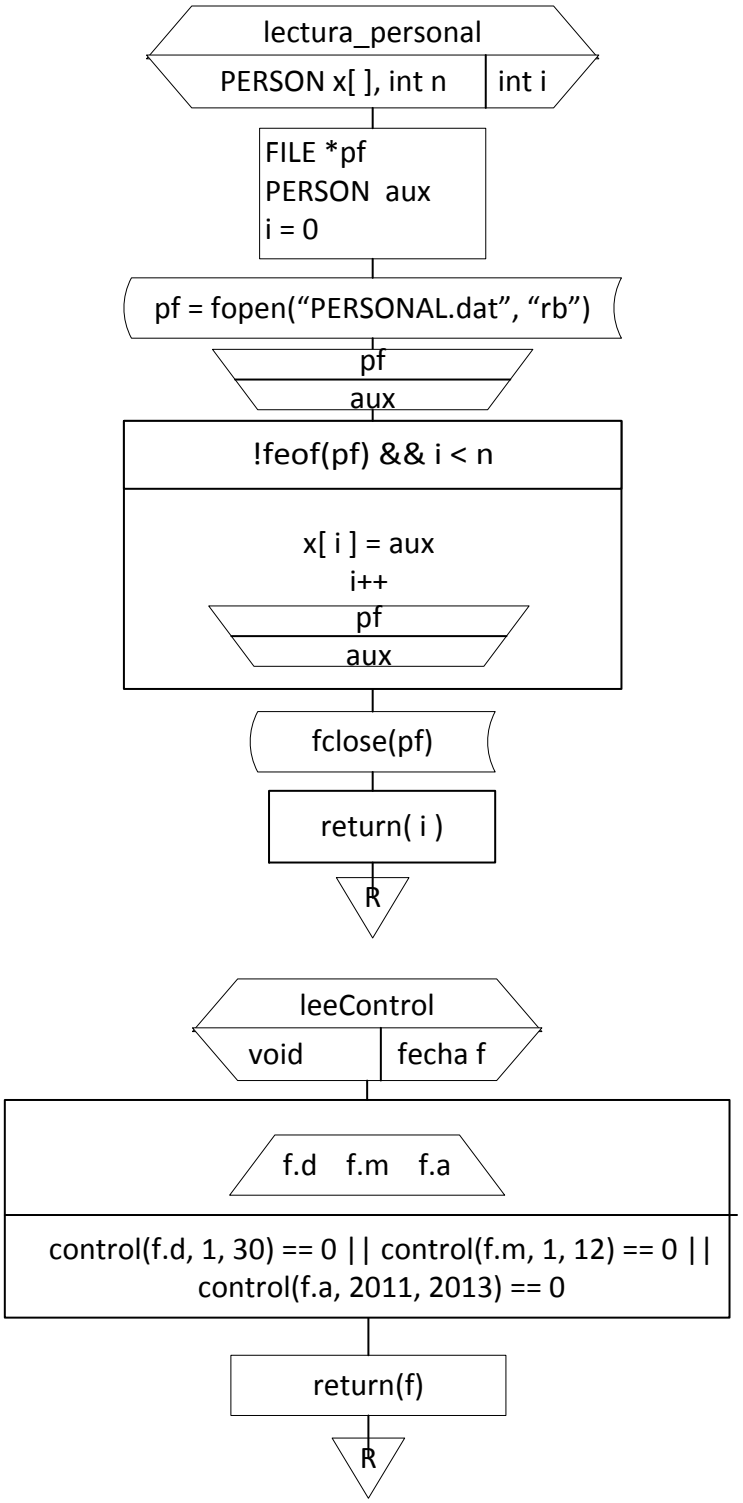
d) Actualizar **ArrayPerso** con las novedades ingresadas
   El importe a cobrar se calcula como el producto entre las horas trabajadas y el valor que cobra por hora el operario. La fecha de proceso es la ingresada por teclado.

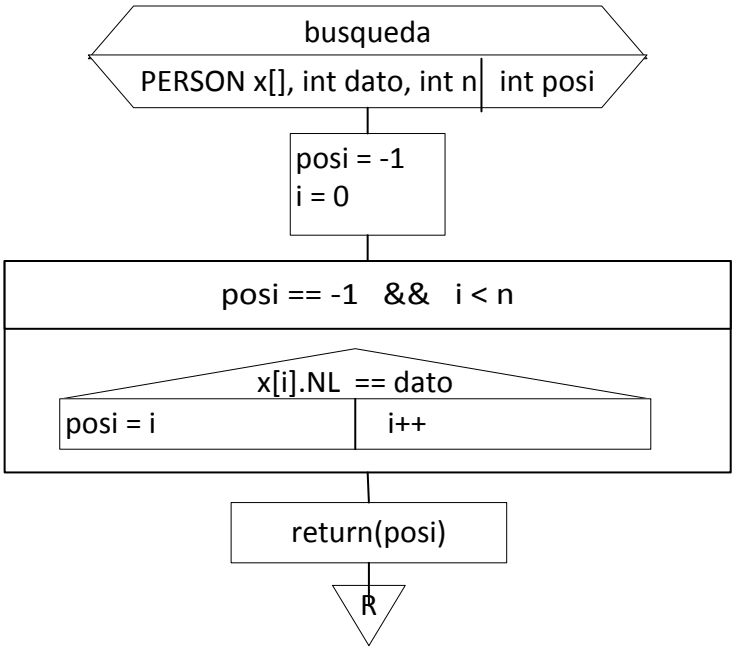
e) Actualizar el archivo **PERSONAL.dat** Mediante función **GRABAR**.

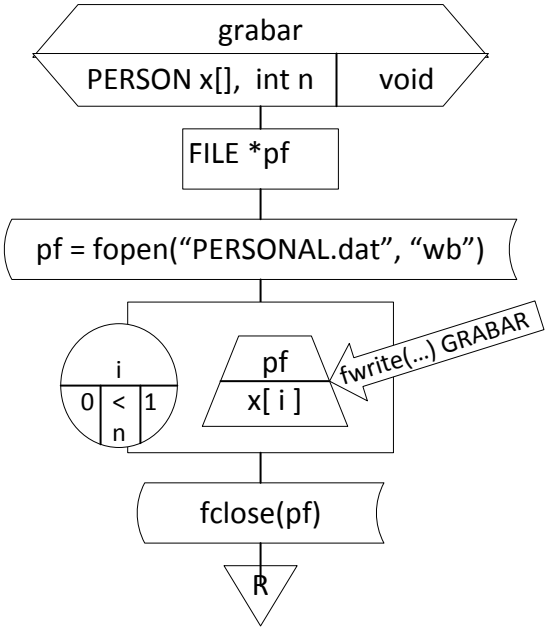
f) Informar los datos de aquellos legajos que tienen un acumulado de sueldo mayor a $ 1.500 y las horas NO exceden a 140. Función **LISTAR**

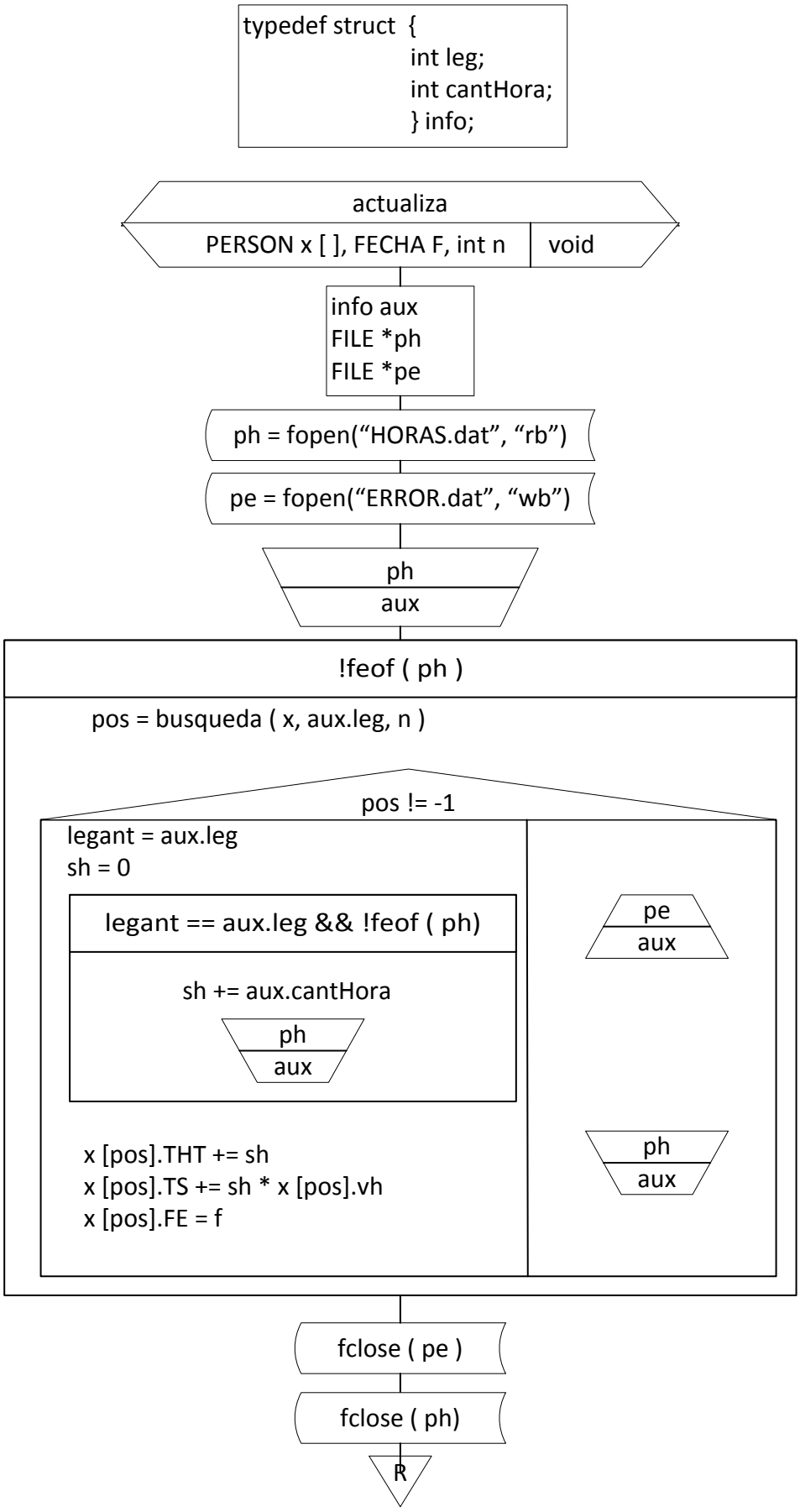


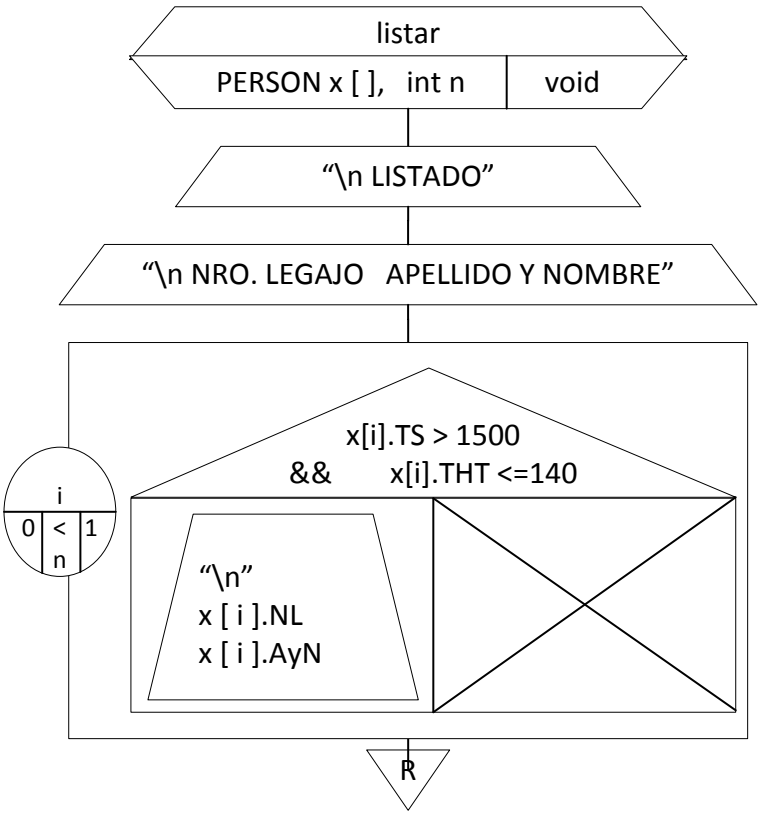

PP
PERSON ArrayPerso[175]
cant = lectura_personal(ArrayPerso , 175)
fech = leeControl()
actualiza(ArrayPerso, fech, cant)
grabar(ArrayPerso, cant)
listar(ArrayPerso, cant)
F
control
int x, int li, int ls
int
x < li || x > ls
return(0)
return(1)
R
MAL
return(0)
BIEN
return(1)
MAL
return(0)
li
ls


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:

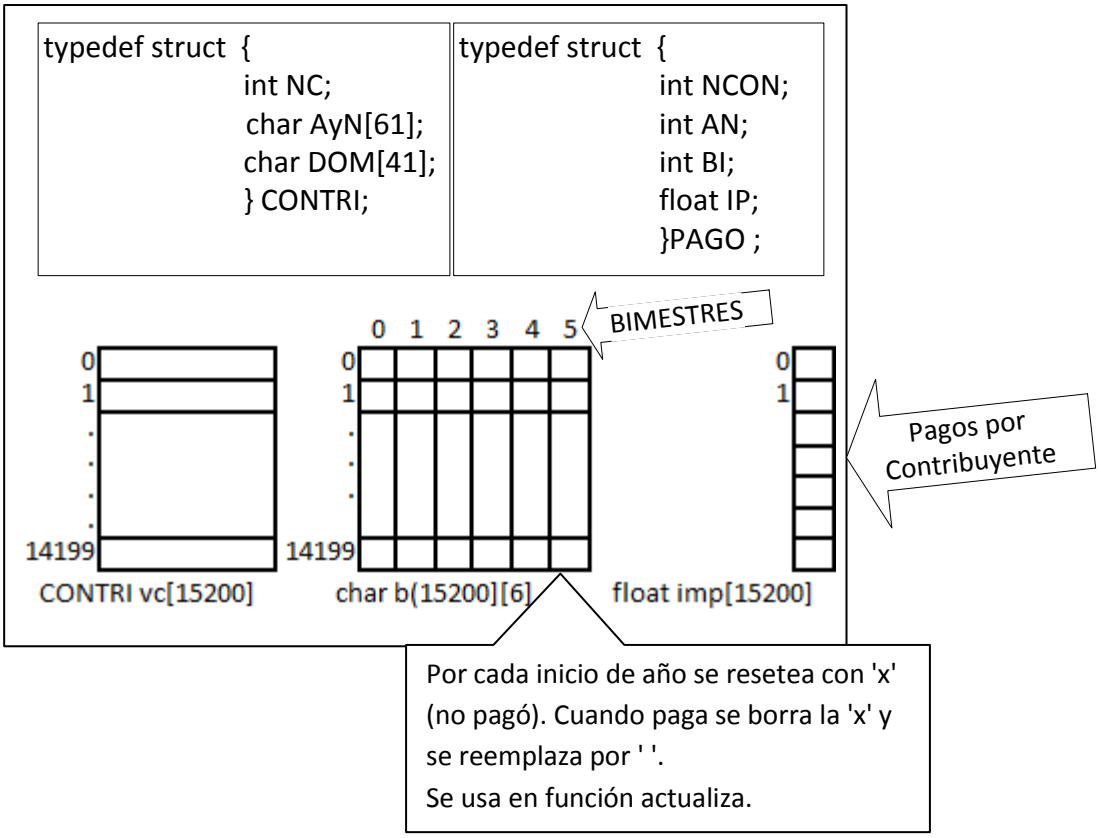
a) El importe total recaudado en cada año.

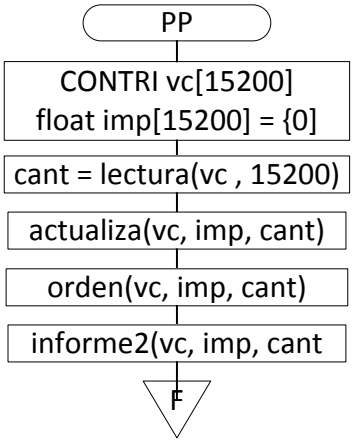
b) Las deudas de cada contribuyente, según el siguiente formato de impresión:

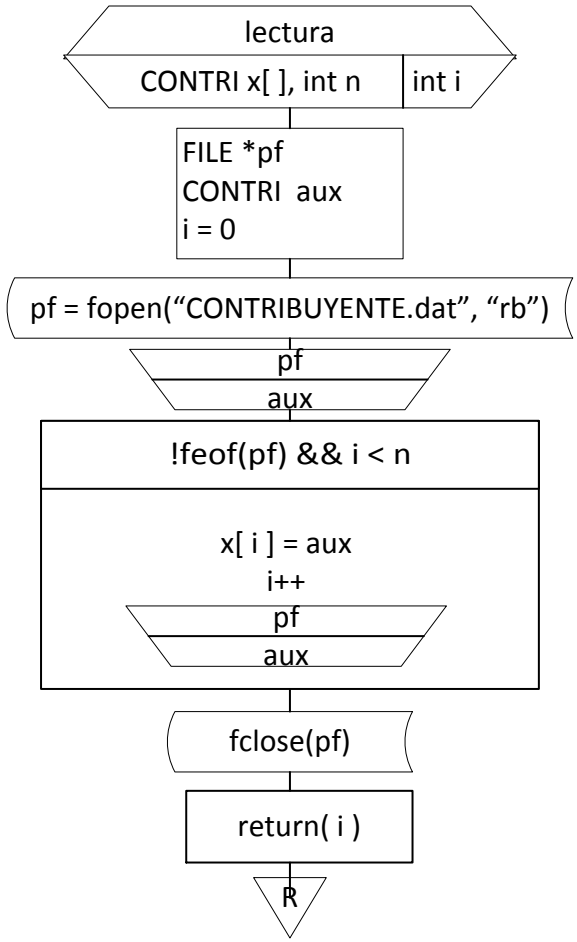
```
        DEUDAS REGISTRADAS DEL AÑO: xxxx
CONTRIBUYENTE     BIMESTRES 1-2-3-4-5-6
    xxxx                    x     x
```

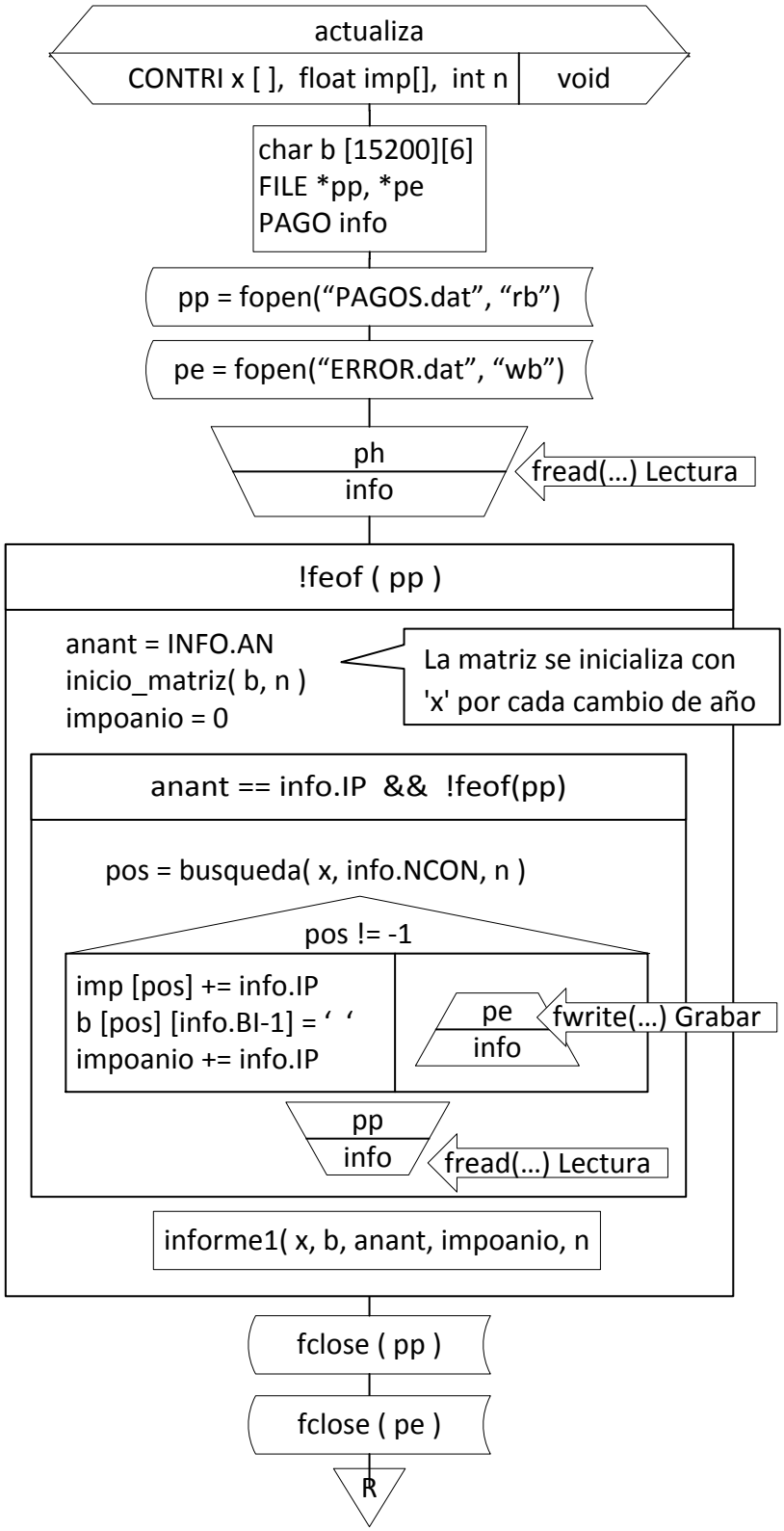
c) Importe total pagado por contribuyente, ordenado por importe total pagado, indicando:

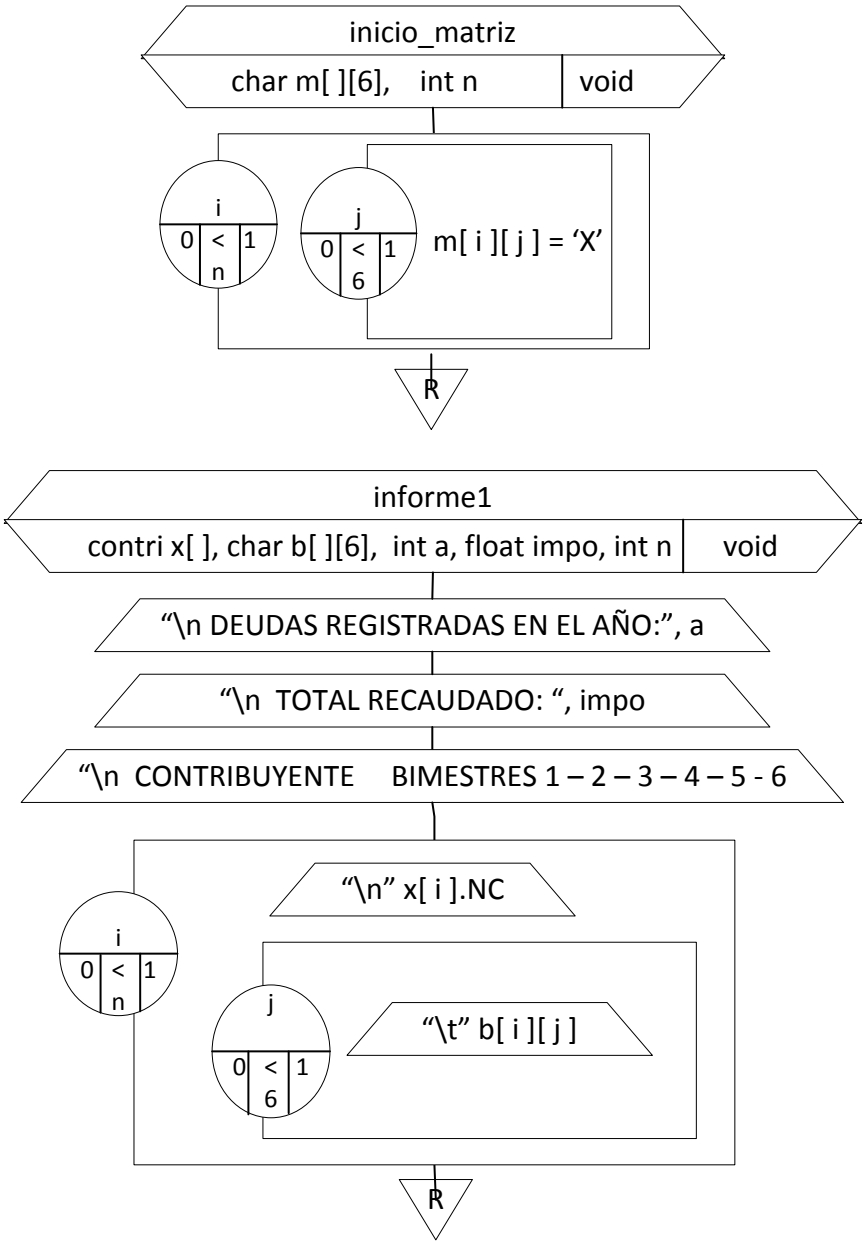
```
Nro. de Contribuyente        Apellido Y Nombre       Importe Total Pagado
```

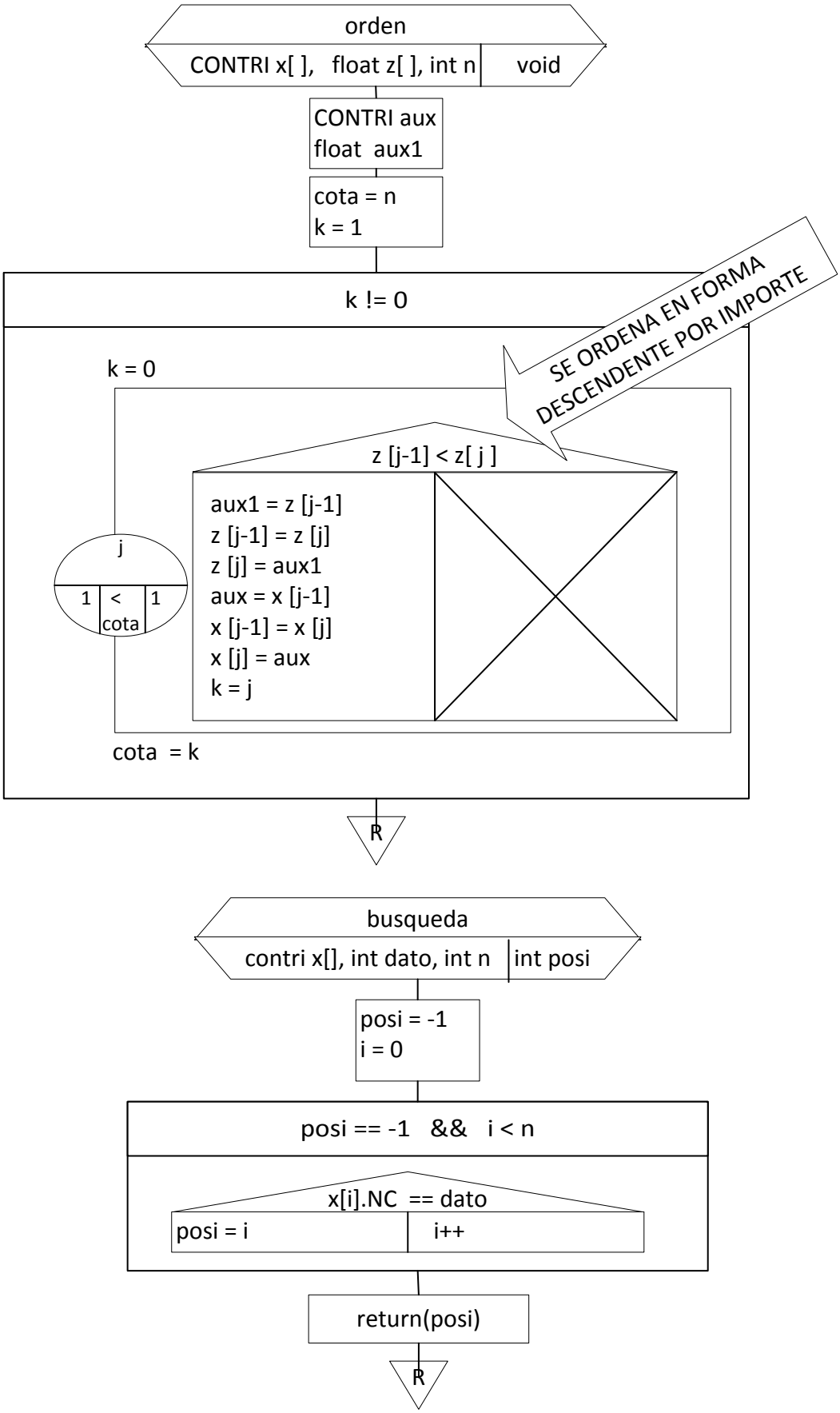

typedef struct {
int NC;
char AyN[61];
char DOM[41];
} CONTRI;
typedef struct {
int NCON;
int AN;
int BI;
float IP;
}PAGO ;
BIMESTRES
0 1 2 3 4 5
0
1
14199
CONTRI vc[15200]
char b(15200)[6]
float imp[15200]
Pagos por Contribuyente
Por cada inicio de año se resetea con 'x' (no pagó). Cuando paga se borra la 'x' y se reemplaza por ' '.
Se usa en función actualiza.

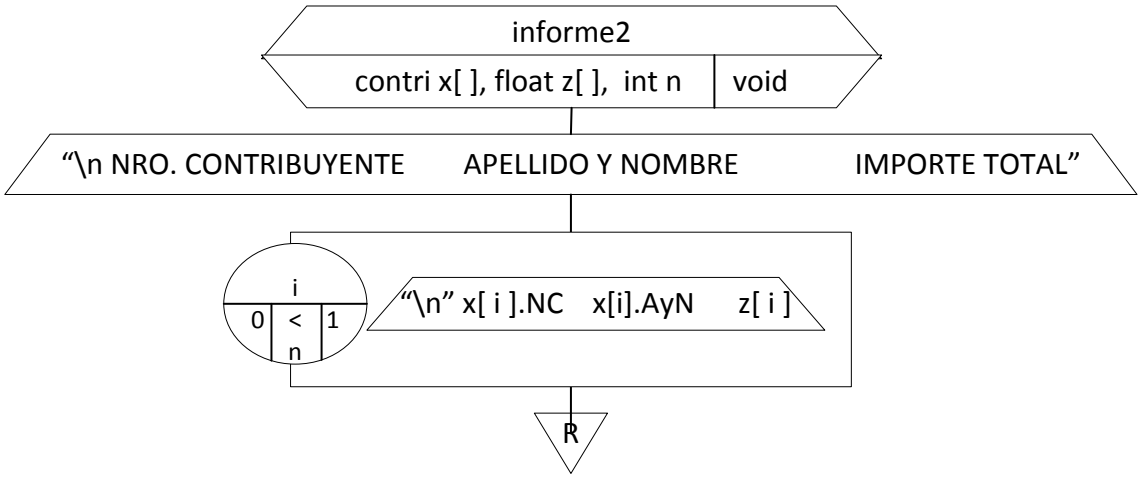

PP
CONTRI vc[15200]
float imp[15200] = {0]
cant = lectura(vc , 15200)
actualiza(vc, imp, cant)
orden(vc, imp, cant)
informe2(vc, imp, cant
F


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 < n 1
j
0 < 6 1
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 < n 1
j
0 < 6 1
"\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;
    }
}
```