INDICE

P8.1

Por medio de un procedimiento INVERTIR se quiere conseguir la cadena inversa a una leída desde el programa principal. Se compararán la cadena leída y su inversa en otro procedimiento COMPARAR (que devuelve un valor booleano) para ver si ambas cadenas son iguales. Si el resultado devuelto es True el programa principal devolverá el mensaje `ESTA CADENA ES UN PALINDROMO'. En caso contrario devolverá el mensaje `ESTA CADENA NO ES UN PALINDROMO'.

```
program palindromo; uses crt; var cad1,cad2:string;

sw:boolean;

procedure invertir(cad1:string;var cad2:string);

var

i:integer;

begin

cad2:='';

for i:=length(cad1) downto 1 do

cad2:=cad2+copy(cad1,i,1);

end;

function comparar(cad1:string;cad2:string):boolean;

var

i:integer;

sv:boolean;

begin

sv:=true;

if length(cad1)<> length(cad2) then

sv:=false

else
```

```
begin
i:=1;
while (sw) and (i<=length(cad1)) do
begin
if cad1[i]<>cad2[i] then sv:=false;
i:=i+1;
end;
end;
comparar:=sv;
end;
begin
clrscr;
write ('ESCRIBE UNA FRASE: ');
READLN (CAD1);
invertir(cad1,cad2);
sw:=comparar(cad1,cad2);
if sw=true then
begin
gotoxy(9,12);writeln('LA FRASE ESCRITA ES UN PALINDROMO');
end
ELSE
begin
gotoxy(9,12);
WRITELN ('LA FRASE ESCRITA NO ES UN PALINDROMO');
end;
REPEAT UNTIL KEYPRESSED;
```

end.

P9.1

Codifique un programa que visualice seis (6) pronósticos para la lotería primitiva. Ninguno de los pronósticos ya generados se puede volver a repetir.

```
program loteriaprimitiva;

uses crt;

type

comb=array[1..6] of integer;

var

i,j:integer;

ok:boolean;

c:comb;

begin

clrscr;

randomize;

for i:=1 to 6 do

begin

repeat

ok:=true;

c[i]:=round(int(random(49))+1);

j:=i−1

while j>0 do

begin

if c[i]=c[j] then ok:=false;

end

until ok;

write(c[i]:4);
```

```
end;

end.
```

P10.2

Codifique un programa que calcule el MAXIMO COMUN DIVISOR de dos números por el algoritmo de Euclides (Dividiendo A entre B se obtiene un resto R. Si R es 0, B será el MAXIMO COMUN DIVISOR, si no, se sigue dividiendo B entre R hasta obtener el resto 0. El último divisor B será el MAXIMO COMLUN DIVISOR).

```
program MCD;uses crt;function resto(x, y: integer): integer; begin resto := x mod y; end;var ok: boolean;

sol, x, y: integer;

begin

clrscr;

ok := false; sol := 0;

write('Escribe el primer número: '); readln(x);

write('Escribe el segundo número: '); readln(y);

repeat

sol := resto(x, y);

if sol = 0 then

begin

write('El mcd es: ', y);

ok := true;

end

else

begin

x := y;

y := sol;

resto(x, y);

end

until ok;
```

repeat until keypressed;

end.

P11.2

Escribir un programa que calcule la frecuencia de aparición de las vocales de un texto leído por teclado. La solución se debe presentar en forma de histrograma. Por ejemplo:

A 12 ==========

E 3 ===

.

.

etc.

program vocales;uses crt;var frase: string; cont: byte; i: integer;

procedure contarvoc(voc: string);

var

num, long, i, j: integer;

begin{contarvoc}

num := 0;

for i := 1 to length(frase) do

begin

if voc = frase[i] then num := num + 1;

end;

write(voc, ' ', num, ' ');

for j := 1 to num do

write('=');

end;

begin {pp}

clrscr;

write('Introduce una frase: '); readln(frase);

longitud := length(frase);

for i := 1 to length(frase) do

frase[i] := upcase(frase[i]);

contarvoc('A'); writeln;

contarvoc('E'); writeln;

contarvoc('I'); writeln;

contarvoc('O'); writeln;

contarvoc('U'); writeln;

repeat until keypressed;

end.

P12.2

Escribir un programa que realice las siguientes tareas:

- Lectura de una frase.
- Conversión de la frase a mayúsculas.
- Cómputo de las palabras de la frase.

Cada una de las tareas deben implementarse por medio de funciones o procedimientos según convenga.

Los resultados se visualizan en el programa principal.

La lectura y escritura de ambos formatos se efectúan en el programa principal.

program fncad;uses crt;var frase: string; tot: byte; num, a, e, i, o, u: byte;procedure convmayus(var frase: string); var i: byte;

begin

for i := 1 to length(frase) do

frase[i] := upcase(frase[i]);

end;

procedure contarvoc(frase: string; var a, e, i, o, u: byte);

var

j: byte;

begin

```
for j := 1 to length(frase) do
begin
if frase[j] = 'A' then a := a + 1;
if frase[j] = 'E' then e := e + 1;
if frase[j] = 'I' then i := i + 1;
if frase[j] = 'O' then o := o + 1;
if frase[j] = 'U' then u := u + 1;
end;
end;
procedure contarpalabras(frase: string; var num: byte);
var
i: byte;
begin
for i := 1 to length(frase) do
if frase[i] = ' ' then num := num + 1
end;
begin {pp}
clrscr; a := 0; e := 0; i := 0; o := 0; u := 0;
num := 0;
write('Introduce una frase: '); readln(frase); writeln;
convmayus(frase);
writeln(frase);
contarvoc(frase, a, e, i, o, u);
contarpalabras(frase, num);
tot:=a + e + i + o + u;
writeln('El número de vocales es: ', tot);
```

```
writeln('El número de palabras es: ', num + 1);

repeat until keypressed;

end.
```

P13.1

Codifique un programa que detecte cuál ha sido la tecla que se ha pulsado y visualice la misma incluidas las teclas especiales y las teclas de función.

```
program teclas;uses crt;

var c:char;begin clrscr; writeln('PULSE UNA TECLA'); c:=readkey; clrscr; write('LA TECLA PULSADA ES: '); if c= #0 then

begin

c:=readkey;

case c of

';':writeln('F1');

'<':writeln('F2');

'=':writeln('F3');

'>':writeln('F4');

'?':writeln('F5');

'@':writeln('F6');

'A':writeln('F7');

'B':writeln('F8');

'C':writeln('F9');

'D':writeln('F10');

'R':writeln('INSERT');

'G':writeln('INICIO');

'I':writeln('REPAG');

'S':writeln('SPR');

'O':writeln('FIN');
```

```
'Q':writeln('AVPAG');

'H':writeln('CURSOR ARRIBA');

'K':writeln('CURSOR IZDA.');

'M':writeln('CURSOR DCHA.');

'P':writeln('CURSOR ABAJO');

end;

end

else

case c of

#9:writeln('TAB');

#13:writeln('ENTER');

#8:writeln('BORRAR');

#27:writeln('ESC');

' ':writeln('SPACE');

else

writeln(c);

end;

readkey;

end.
```

P14.1

Escribir un programa que, leída una frase, visualice todas las letras de la misma ordenadas alfabéticamente.

```
program ordenarfrase;uses crt;var frase: string; aux: char; i, j: integer;begin clrscr; writeln('Introduce una frase: '); readln(frase);

for i := 1 to length(frase) do

begin

for i := 1 to length(frase) − 1 do

for j:=1 to i do
```

```
begin

if ord(frase[j]) > ord(frase[j + 1]) then

begin

aux := frase[j];

frase[j] := frase[j + 1];

frase[j + 1] := aux;

end;

end;

end;

writeln(frase);

repeat until keypressed;

end.
```

P15.2

Escribir un programa que halle todos los números primos menores a un número dado N y que los visualice en pantalla. (CRIBA DE ERATOSTENES).

```
program primos;

uses crt;

const

m=30;

var

n: array[1..m] of boolean;

i, j: 1..m;

begin

clrscr;

for i := 1 to m do

n[i] := true;

for i := 2 to trunc(sqrt(m)) do
```

```
begin

j := sqr(i);

repeat

n[j] := false;

inc(j, i);

until j > 30;

end;

for i := 1 to m do

if n[i] then write (i: 3);

readln;

end.
```

P18.1

Escribir un programa que rellene un vector V de N elementos, los ordene y visualice el vector desordenado y también clasificado. Cada una de las acciones del programa se realizará por medio de procedimientos que son llamados desde el programa principal.

```
program vectores;

uses crt;

const N = 20;

type

vec = array[1..N] of integer;

var

vector: vec;

procedure rellenar (var vector: vec);

var

i: integer;

begin

randomize;
```

```
for i := 1 to N do

vector[i] := random(1000);

end;

procedure vervector (vector: vec);

var i: integer;

begin

for i := 1 to N do

write(vector[i], ',');

writeln;

end;

procedure ordenar (var vector: vec);

var i, j, aux: integer;

begin

for i := 1 to N − 1 do

for j := 1 to N − 1 do

if vector[j] > vector[j + 1] then

begin

aux := vector[j];

vector[j] := vector[j + 1];

vector[j + 1] := aux;

end;

end;

begin {pp}

clrscr;

rellenar (vector);

write('Vector sin ordenar: '); writeln; vervector (vector);
```

```
ordenar (vector); writeln;

write('Vector ordenado: '); writeln; vervector (vector);

repeat until keypressed;

end.
```

P21.2

Se desea procesar los datos de los alumnos de un determinado instituto por medio de un array de registro. El tipo registro está diseñado y se ha de declarar para que pueda contener los siguientes datos:

NOMBRE DEL ALUMNO

CURSO

EDAD

CALIFICACION MEDIA DEL CURSO ANTERIOR

La aplicación posibilita las siguientes opciones:

1. Captura de datos.

2. Ordenación de los mismos por el curso y nombre.

3. Búsqueda de un alumno por su nombre y visualización de su ficha.

4. Listado de los alumnos cuya calificación media del curso anterior sea mayor o igual a 8.5 puntos.

```
program notas;

uses crt;

const n = 10;

type

alumno = record

nombre: string;

curso: byte;

edad: byte;

calif: single;

end;

alum = array[1..n] of alumno;
```

```
var
a: alum;
bus: string;
pos, menu: integer;
procedure insertar(var a: alum; n: integer);
var i: integer;
begin
for i := 1 to n do
with a[i] do
begin
clrscr;
write('Introduce nombre: '); readln(nombre);
write('Introduce curso: '); readln(curso);
write('Introduce edad: '); readln(edad);
write('Introduce calificaci¢n: '); readln(calif);
end;
end;
procedure listar(a: alum; n:integer);
var j, c: integer;
begin
c := 3; clrscr;
write(' Nombre Edad Curso Nota'); writeln;
write(' ==========================================='); writeln;
writeln;
for j := 1 to n do
with a[j] do
```

```
if calif >= 8.5 then
begin
gotoxy(2, c); write(nombre); gotoxy(18, c); write(edad);
gotoxy(33, c); write(curso); gotoxy(45, c); write(calif: 1);
writeln; c := c + 1
end;
repeat until keypressed;
end;
procedure intercambia(var a, b: alumno);
var aux: alumno;
begin
aux := a;
a := b;
b := aux;
end;
procedure ordenar_curso(var a: alum; N: integer);
var int, i, j, k: integer;
begin
int := N div 2;
while int > 0 do
begin
for i := (int + 1) to N do
begin
j := i − int;
while j > 0 do
begin
```

```
k := j + int;
if a[j].curso <= a[k].curso then
j := 0
else
intercambia(a[j], a[k]);
j := j − int;
end;
end;
int := int div 2;
end;
end;
procedure ordenar_nombre(var a: alum; N: integer);
var
aux, int, i, j, k: integer;
begin
int := N div 2;
while int > 0 do
begin
for i := (int + 1) to N do
begin
j := i − int;
while j > 0 do
begin
k := j + int;
if a[j].nombre <= a[k].nombre then
j := 0
```

```
else
intercambia(a[j], a[k]);
j := j − int;
end;
end;
int := int div 2;
end;
end;
function buscar(bus: string; a: alum; n: integer): integer;
var
p, u, c: integer;
ok: boolean;
begin
p := 1; u := n; ok := false;
while (p <= u) and (not ok) do
begin
c := (p + u) div 2;
if bus = a[c].nombre then
ok := true
else
if bus > a[c].nombre then
p := c + 1
else
u := c − 1;
end;
if not ok then
```

```
buscar := 0
else
buscar := c
end;
procedure visualizar(a: alum; pos: integer);
begin
clrscr;
with a[pos] do
begin
writeln('Nombre: ', nombre);
writeln('Edad: ', edad);
writeln('Curso: ', curso);
writeln('Nota: ', calif: 1);
repeat until keypressed;
end;
end;
begin {pp}
repeat
clrscr;
writeln('1. Introducir datos');
writeln('2. Ordenar datos por nombre');
writeln('3. Ordenar datos por curso');
writeln('4. Buscar un alumno');
writeln('5. Listado de los que tengan m s de 8.5');
writeln('6. Salir'); writeln;
write('Elija opci¢n: '); readln(menu);
```

```
case menu of

1: insertar(a, n);

2: ordenar_nombre(a, n);

3: ordenar_curso(a, n);

4: begin

clrscr;

write('Introduce nombre a buscar: '); readln(bus);

pos := buscar(bus, a, n);

if pos <> 0 then

visualizar(a, pos)

else

begin

write('Nombre no encontrado');

repeat until keypressed

end;

end;

5: listar(a, n);

end;

until menu = 6;

end.
```

P22.2

Crear un archivo de texto COLUMNAS.MAT que contendrá, como caracteres, la información relativa a las notas de los alumnos de tres clases. Cada columna contendrá las calificaciones de una clase. La primera columna contiene la información de la clase A., la segunda columna contiene la información de la clase B y la tercera, la de la clase C.

El programa deberá contemplar:

a) Escritura del archivo.

b) Lectura de los datos del archivo.

c) Visualización de los datos del archivo encolumnados por clase.

d) Cálculo y visualización de las medias aritméticas de cada columna.

```
program columnas;

uses crt;

var

fich: text;

procedure escritura( var f: text);

var

nota1, nota2, nota3: real;

j: integer;

begin

rewrite(f);

j := 1;

repeat

clrscr;

write('Introduce las notas, 0 para terminar...');

read(nota1, nota2, nota3);

gotoxy(10, j);

inc(j);

if nota1 <> 0 then write(f, nota1, nota2, nota3);

until nota1 = 0;

close(f);

end;

procedure lectura(var f: text);

var

med1, med2, med3, nota1, nota2, nota3: real;
```

```
j: integer;
begin
reset(f);
j := 1;
clrscr;
while not eof(f) do
begin
read(f, nota1, nota2, nota3);
gotoxy(10, j);
write(nota1: 4: 2,' ', nota2: 4: 2,' ', nota3: 4: 2);
inc(j);
med1 := nota1 + med1;
med2 := nota2 + med2;
med3 := nota3 + med3;
end;
gotoxy(0, j + 1);
writeln;
med1 := med1 / (j − 1); med2 := med2 / (j − 1); med3 := med3 / (j − 1);
write('================================'); writeln;
write('Media: ', med1: 4: 2,' ', med2: 4: 2,' ', med3: 4: 2);
close(f);
end;
begin {pp}
assign(fich, 'columnas.txt');
escritura(fich);
lectura(fich);
```

```
repeat until keypressed;

end.
```

P23.2

Escribir un programa que permita realizar una copia de seguridad de un archivo de texto.

```
program copia_seguridad;

uses crt;

const n1 = '.seg';

type

cadena = string[8];

cadena2 = string[12];

var

f1, f2: text;

n: cadena2;

c: string;

function nombre(n: cadena2): cadena2;

var

ok: boolean;

c: string[1];

aux: cadena;

i: integer;

begin

aux := ''; i := 1;

ok := false;

repeat

c := copy(n, i, 1);

if c = '.' then ok := true;
```

```
i := i + 1;

aux := aux + c;

until ok or (i > length(n));

if i > length(n) then aux := aux + '.';

nombre := aux + n1;

end;

begin {pp}

clrscr;

write('Nombre del archivo: ');

read(n);

write(nombre(n));

assign(f1, n);

assign(f2, nombre(n));

reset(f1);

rewrite(f2);

while not eof(f1) do

begin

readln(f1, c);

writeln(f2, c);

end;

write(nombre(n));

close(f1); close(f2);

repeat until keypressed;

end.
```

P25.1

Con los códigos de esta práctica y un procedimiento MENU que posibilite las acciones que se han descrito en el apartado APLICACIONES DE LOS ARCHIVOS SIN TIPO A DIRECTORIOS, codifique un programa

completo capaz de realizar las tareas COPIA, BORRADO Y RENOMBRADO de un archivo ya existente. No olvide validar los nombre que se refieran a archivos y que el usuario debe escribir como respuesta a la petición de un nombre de archivo.

```
Uses

Crt,Dos;

type

nomfi=string[12];

sintipo=file;

var

opcion:1..4;

Procedure Validar(nombre:nomfi;var valido:boolean);

var

x:1..12;

acun:0..8;

acue:0..3;

Finn:boolean;

begin

Finn:=False;

Valido:=False;

for x:=1 to 12 do

begin

if (nombre[x]<>'.') and (not finn) then

acun:=acun+1

else

finn:=True;

if (nombre[x]<>'.') and (finn) then

acue:=acue+1;
```

```
end;
if (acun<=8) and (acue<=3) then
Valido:=True;
end;
Function Existe(nombre:nomfi):boolean;
var
fich:sintipo;
begin
{$I−}
assign (fich,nombre);
Reset (fich);
Close (fich);
{$I+}
Existe:=(IOResult=0) and (nombre<>'');
end;
Procedure Borrar; var fich:sintipo; nombre:nomfi; valido:boolean; begin Repeat
ClrScr;
Write ('Nombre del archivo : ');
ReadLn (nombre);
Validar(nombre,valido);
If not valido then
WriteLn ('Nombre de fichero no válido');
Repeat Until KeyPressed;
Until valido;
If existe(nombre) then
begin
```

```
Assign (fich,nombre);

Erase (fich);

Close (fich)

end;

if not existe(nombre) then

WriteLn (nombre,' no se encuentra en el directorio');

end;

Procedure Renombrar; var nombrea,nombreb:nomfi; fich:sintipo; valido:boolean; begin Repeat

ClrScr;

Write ('Nombre del archivo a renombrar:');

ReadLn (nombrea);

Validar (nombrea,valido);

If not valido then

WriteLn ('Nombre de archivo no valido');

Repeat Until KeyPressed;

Until valido;

Write ('Nuevo nombre : ');

ReadLn (nombreb);

Assign (fich,nombrea);

Rename (fich,nombreb);

Erase (fich);

Close (fich)

end;

if not existe(nombrea) then

WriteLn (nombrea,' no se encuentra en el directorio.');

Repeat Until KeyPressed;
```

```
end;
begin
Repeat
Repeat
ClrScr;
WriteLn ('1.Borrar fichero');
WriteLn;
WriteLn ('2.Renombrar fichero');
WriteLn;
WriteLn ('3.Salir a DOS');
WriteLn;
Write ('Elige opción : ');
ReadLn (opcion);
Until (opcion>0) and (opcion<4);
case opcion of
1:Borrar;
2:Renombrar;
3:Halt;
end;
Until 4>5;
end.
```

P26.2

Escribir un programa que busque en un archivo de enteros los enteros mayor y menor del mismo

```
program busqueda;
uses crt;
const arr = 100;
```

```
type
fichero = file of integer;
tv = array[1..arr] of integer;
var
v: tv;
f: fichero;
min, max, i, r: integer;
procedure buscar(v: tv; i: integer; var min, max: integer);
var
k: integer;
begin
for k := 1 to i do
begin
if v[k] < min then min := v[k];
if v[k] > max then max := v[k];
end;
end;
begin {pp}
assign(f, 'vector.txt');
reset(f); i := 1;
while not eof(f) do
begin
while (i <= arr) and (not eof(f)) do
begin
read(f, r);
inc(i);
```

```
end;

buscar(v, i, min, max);

end;

clrscr;

write('M ximo = ', max, ' M¡nimo = ', min);

repeat until keypressed;

close(f);

end.
```

P28.2

Diseñar un algoritmo y escribir su correspondiente programa para convertir una expresión de notación infija a notación polaca inversa (postfija).

Observe, como ayuda al enunciado, las siguientes conversiones:

NOTACION INFIJA POLACA POLACA INVERSA

a+b +ab ab+

a−b −ab ab−

a*b *ab ab*

(a+b)*c *+abc abc*+

a AND b OR c OR AND abc abc OR AND

```
program p28_2;

uses crt;

type

ptro=^nodo;

nodo=record

info:string;

sig:ptro;

end;

var
```

```
op,dat,dat1:ptro;
cad1,frase:string;
i:integer;
function pilavacia(p:ptro):boolean;
begin
pilavacia:=p=nil;
end;
procedure apilar(var p:ptro;e:string);
var
nuevo:ptro;
begin
if pilavacia(p) then
begin
new(p);
p^.info:=e;
p^.sig:=nil;
end
else
begin
new(nuevo);
nuevo^.info:=e;
nuevo^.sig:=p;
p:=nuevo;
end;
end;
procedure desapilar(var a:ptro;var e:string);
```

```
var
aux:ptro;
begin
if not pilavacia(a) then
begin
e:=a^.info;
aux:=a^.sig;
dispose(a);
a:=aux;
end;
end;
procedure recorrer(p:ptro);
begin
while not pilavacia(p) do
begin
write(p^.info);
p:=p^.sig;
end;
end;
begin {pp}
writeln('Escribe una operacion para transformarla a notaci¢n polaca inversa ');
readln(frase);
writeln;
cad1:='';
for i:=1 to length(frase) do
begin
```

```
if frase[i] in ['+', '−', '*', '/', '^', '(', ')'] then

apilar(op,frase[i]);

if frase[i] in ['A', 'N', 'D', 'O', 'R'] then

begin

cad1:=cad1+frase[i];

if frase[i+1]=' ' then

begin

cad1:=' '+cad1;

apilar(op,cad1);

cad1:='';

end;

end;

if frase[i] in ['a'..'z'] then apilar(dat,frase[i]);

end;

while not pilavacia(dat) do

begin

desapilar(dat,cad1);

apilar(dat1,cad1);

end;

recorrer(dat1);

recorrer(op);

readkey;

end.
```

P29.1

Escribir un programa que simule el funcionamiento de una oficina de reservas para alquiler de coches que atiende al cliente por medio de llamadas telefónicas. Si el empleado está libre, atiende al cliente y si no lo está, se sitúa al cliente en una cola de espera, hasta tanto se hayan atendido todas las solicitudes anteriores a la suya.

```
program Alquiler_Coches;

uses CRT;

type

OMenu = 0..2;

PCola = ^RCola;

RCola = record

Cliente : Integer;

Sig : PCola

end;

var

PrinC, FinalC : PCola;

Opcion : oMenu;

Num : Integer;

procedure Menu (var Opcion : OMenu );

begin {procedimiento menu}

repeat

GotoXY(32,2);

Write ('MENU');

GotoXY(25,5);

Write('1. Llega otro cliente');

GotoXY(25,7);

Write('2. Atender un cliente');

GotoXY(25,10);

Write('0. SALIR');

GotoXY(29,15);

Write('OPCION: ');
```

```
GotoXY(37,15);

Readln (Opcion)

until (Opcion >= 0) and (Opcion < 3)

end; {procedimiento men£}

procedure MostrarMenu; begin {procedimiento mostrarmenu}

GotoXY(32,2);

Write ('MENU');

GotoXY(25,5);

Write('1. Llega otro cliente');

GotoXY(25,7);

Write('2. Atender un cliente');

GotoXY(25,10);

Write('0. SALIR');

GotoXY(29,15);

Write('OPCION: ');

end;

procedure MeterEnCola (var PrinC, FinalC : PCola); var Aux : PCola; begin {Procedimiento MeterEnCola} if
Princ = nil then Num := 1

else

Num := Num + 1;

New (Aux);

Aux^.Cliente := Num;

Aux^.Sig := nil;

if PrinC = nil then

PrinC := Aux

else

FinalC^.Sig := Aux;
```

```
FinalC := Aux

end;

procedure SacarDeCola ( var PrinC, FinalC : PCola );

var

Aux : PCola;

begin {procedimiento SacarDeCola}

if PrinC <> nil then

begin

Aux := PrinC;

if PrinC = FinalC then

begin

PrinC := nil;

FinalC := nil

end

else

PrinC := PrinC^.Sig;

Dispose (Aux)

end

end; {Procedimiento SacarDeCola}

procedure MostrarCola (PrinC, FinalC : PCola); var Aux : PCola; begin {Procedimiento MOstrarCola}
CLRSCR;

MostrarMenu;

Aux := PrinC;

GotoXY (5,22);

Write ('Cola : ');

while Aux <> nil do

begin
```

```
Write (Aux^.Cliente,' ');
Aux := Aux^.Sig
end
end;
begin {pp} CLRSCR;
repeat
Menu (Opcion);
case Opcion of
1 : begin
MeterEnCola (PRinC,FinalC);
MostrarCola (PrinC,FinalC)
end;
2 : begin
SacarDeCola (PrinC,FinalC);
MostrarCola (PrinC,FinalC)
end
end
until Opcion = 0
end. {pp}
```

P31.1

Escribir un programa que ejecute las operaciones siguientes para una lista enlazada que contiene nombre:

a) Declarar los elementos precisos.

b) Crear la lista.

c) Insertar elementos al final de la lista.

d) Recorrer la lista y visualizar su contenido.

```
program p31_1;
```

```
uses crt;

type

puntero=^registro;

registro=record

nombre:string[10];

link:puntero;

end;

var

cab:puntero;

x:integer;

procedure intro(var cab:puntero);

var

ant,act,aux:puntero;

begin

new(aux);

write('Introduce nombre : ');

readln(aux^.nombre);

ant:=nil;

act:=cab;

while (act<>nil) (*and (act^.nombre<aux^.nombre)*) do begin

ant:=act;

act:=act^.link;

end;

if (ant<>nil) then begin

ant^.link:=aux;

aux^.link:=act;
```

```
end else begin

aux^.link:=cab;

cab:=aux;

end;

end;

procedure show(cab:puntero); var p:puntero; begin p:=cab; while (p<>nil) do begin writeln(p^.nombre);
p:=p^.link; end; end;

begin

clrscr;

cab:=nil;

for x:=1 to 5 do intro(cab);

writeln('Salida:');

show(cab);

end.
```

P32.1

Utilizar un árbol binario para ordenar un archivo de inventario de acuerdo al número de código del artículo. Visualizar los resultados.

```
program p32_1;

uses crt;

type

puntero=^tnodo;

tnodo=record

codigo:integer;

articulo:string[20];

izq,der:puntero;

end;

var
```

```
raiz:puntero;

x:integer;

procedure busca(cab:puntero;elem:integer;var ant,act:puntero);

var

sw:boolean;

begin

sw:=false;

ant:=nil;

act:=cab;

while (act<>nil) and (not sw) do begin

if (act^.codigo=elem) then

sw:=true

else begin

ant:=act;

if (elem<act^.codigo) then

act:=act^.izq

else

act:=act^.der;

end;

end;

end;

procedure intro(var cab:puntero); var act,ant,aux:puntero; x:integer; begin write('Introduce c¢digo : ');
readln(x); busca(cab,x,ant,act); if (act<>nil) then

writeln('Clave ya existe. Ignorando nueva.')

else begin

new(aux);

aux^.codigo:=x;
```

```
write('Introduce nombre : ');
readln(aux^.articulo);
aux^.izq:=nil;
aux^.der:=nil;
if (ant=nil) then
cab:=aux
else
if (x<ant^.codigo) then
ant^.izq:=aux
else
ant^.der:=aux;
end;
end;
procedure show(cab:puntero);
begin
if (cab<>nil) then begin
show(cab^.izq);
writeln(cab^.articulo,' (cod:',cab^.codigo,')');
show(cab^.der);
end;
end;
begin
raiz:=nil;
clrscr;
for x:=1 to 3 do intro(raiz);
clrscr;
```

```
show(raiz);

repeat until keypressed;

end.
```