Problema 4.1

Escribir un programa que acepte tres números por teclado, los ordene de menor a mayor, y los muestre por pantalla.

```
program Clasificador (input,output);

{El programa clasifica tres valores reales introducidos por teclado

y los muestra en el nuevo orden.

1.− Entrada de datos por teclado

2−. Clasificación ascendente de los valores introdcidos

3.− Salida por pantalla Clasificados }

uses crt; {en la Librería Standard de Pascal}

var {variables globales}

x,y,z:real;

procedure EntDatos (var x,y,z:real);

{Postcondición: input=[m,n,q] " x=m " y=n " z=q}

begin

write('Escribe el valor real de X: ');

readln (x);

write('Escribe el valor real de Y: ');

readln (y);

write('Escribe el valor real de Z: ');

readln (z);

end;

procedure Confirma (x,y,z:real);

{Precondición: [x,y,z] " a R}

{Postcondición: output=[x,y,z]}

begin {Confirmación de datos}
```

```
gotoxy(1,5);
writeln (' Has escrito los siguientes datos');
writeln (' X= ',x:7:2);
writeln (' Y= ',y:7:2);
writeln (' Z= ',z:7:2);
readln;
end; {de Confirma}
procedure Clasifica (VAR x,y,z:real);
procedure Intercambia (var x,y:real);
{Precondición: [x,y] "a R " x=m " y=n}
{Postcondición: x=n " y=m}
var
aux:real; {local}
begin {Intercambio de valores}
aux:=x;
x:=y;
y:=aux;
end; {de Intercambia}
begin {De Clasifica}
if x>y then
Intercambia(x,y);
if y>z then
Intercambia(y,z);
if x>y then
Intercambia (x,y)
end; {De Clasifica}
```

```
procedure Mostrar(x,y,z:real);

{Precondición: [x,y,z] "a R " x"y"z }

{Postcondición: output=[x,y,z]}

begin {De Mostrar}

writeln (' Los valores ordenados son ',x:5:2, ' <= ',y:5:2,

' <= ',z:5:2);

readln;

end; {de Mostrar}

begin {Programa Principal}

ClrScr;

EntDatos(x,y,z);

Confirma (x,y,z);

Clasifica(x,y,z);

Mostrar(x,y,z)

end. {Del Programa Principal}
```

Problema 4.2

Escribir un programa que acepte una letra minúscula por teclado , y muestre por pantalla la misma letra en mayúscula.

```
program Mayuscula1 (input,output);

{Muestra por pantalla la mayúscula de una letra minúscula aceptada por teclado.

Esta versión considera la posibilidad de introducir una Mayúscula

pero no distingue si es otro tipo de carácter. Por ejemplo un 2 o

un blanco, ...}

uses crt;

var

Minus,

Mayus:char;
```

```
function EsMinuscula(Minus:char):boolean;
{ Precondición: Minus e C
Invariante: Minus = mn
Postcondición:EsMinuscula= ('a'<= Minus) and (Minus <= 'z') }
begin {De EsMinúscula}
EsMinuscula:= ('a'<= Minus) and (Minus <= 'z');
end; {De EsMinúscula}
procedure TomaDato (var Minus:char);
{ Postcondición:Input= mn y Minus=mn}
begin
write ('Escribe letra min£scula: ');
readln(Minus);
end;
procedure Encuentra (Minus:char;var Mayus:char);
{ Precondición: Minus e C
Invariante: Minus = mn
Postcondición: Mayus= chr( ord(Minus) + ord('A') − ord('a')) Or Mayus:=Minus; }
begin {De Encuentra}if EsMinuscula(Minus) then Mayus:=chr( ord(Minus) + ord('A') − ord('a'))
else Mayus:=Minus;
end; {De Encuentra}
procedure Muestra(Minus,Mayus:char);
{ Precondición: [Minus,Mayus] e C
Postcondición:Output=[(Minus or Mensaje) y Mayus]}
begin
writeln;
if EsMinuscula(Minus) then
```

writeln ('La Minúscula es: ',Minus)

else

writeln ('Pusiste Mayúscula, pero...');

writeln('La Mayúscula es: ',Mayus);

readln

end;

begin {Programa Principal}

clrscr;

TomaDato(Minus);

Encuentra(Minus,Mayus);

Muestra(Minus,Mayus);

end. {del Programa Principal}

Problema 4.3

Vuelve a escribir el programa Maximo de tres números, utilizando la estructura de decisión anidadas.

program **maximo3b** (input,output); {Calcula el máximo de 3 enteros introducidos por teclado 1.− Recoge los datos del teclado 2.− Calcula el máximo de los tres

3.− Muestra por pantalla el máximo }

var

N1,N2,N3,

Maximo:integer;

procedure **TomaDatos**(VAR x,y,z:integer);

{Toma los datos por teclado y los comprueba}

{Precondicion: [x,y,z] " a R

Postcondicion: Input=[a,b,c] " x=a" y=b" z=c}

procedure **CompruebaDatos**(x,y,z:integer);

{Precondicion: [x,y,z] pertenecen a R

Inveriante: x=a"y=b" z=c

```
{Postcondicion: output=[x,y,z] }
begin {De CompruebaDatos}
writeln ('Los datos Inclu¡dos son: ',x:5,y:5,z:5);
readln;
end; {De Compruebadatos}
begin {De TomaDatos}
write('Introduce N1: ');
readln(x);
write('Introduce N2: ');
readln(y);
write('Introduce N3: ');
readln(z);
CompruebaDatos(x,y,z);
end; {De TomaDatos}
procedure MuestraResultado (x:integer);
{Precondicion: x "a R
Invariante: x=Maximo
Postcondicion: Output=x}
begin
writeln ('El Máximo es: ',x);
readln;
end;
function Max3(x,y,z:integer):integer;
{Precondicion: x,y,z " a R
Postcondicion: Max3 = x " y " z}
begin
```

```
if x>y then

if x>z then

Max3:=x

else

Max3:=z

else

if y>z then

Max3:=y

else

Max3:=z;

end;

begin {Programa Principal}

TomaDatos(N1,N2,N3);

Maximo:= Max3(N1,N2,N3);

MuestraResultado(Maximo);

end. {del Programa Principal}
```

Problema 4.4

Escribir un programa que acepte un valor por teclado, muestre el siguiente menú por patalla:

F Para convertir de Celsius a Farhenheit C Para convertir de Farhenheit a Celsius K Para convertir de Farhenheit a Kelvin

Elige Opcion (Pulsa F, C o D):

Y termine mostrando el valor introducido, la conversión realizada y el resultado de la conversión.

program **ConvierteTemperaturas** (input,output); { 1.− Acepta por pantalla un valor real 2.− Muestra en pantalla el siguiente menú: F Para convertir de Celsius a Farhenheit

C Para convertir de Farhenheit a Celsius

K Para convertir de Farhenheit a Kelvin

Elige Opcion (Pulsa F, C o D):

3.− Muestra en pantalla el valor introducido, la operación elegida y el resultado de la conversión elegida

sabiendo que ºC= k − 273

ºC= (5/9) * (ºF − 32.0)

Esta versión no utiliza estructura CASE } **uses** crt; **var** x, xConvertido:real; opcion:char;

procedure **TomaDato**(var x:real);

{Postcondición Input=n " n>=−459.69 " x=n }

begin

writeln ('Introduce valor: ');

readln (x);

end;

procedure **EligeOpcion** (var opcion:char);

{Postcondición Input=opcion " (opcion=F " opcion=C " opcion=K)

Esta postcondición no se cumple en esta versión

}

begin

writeln;

writeln ('F para convertir de Celsius a Farhenheit');

writeln ('C para convertir de Farhenheit a Celsius ');

writeln ('K para convertir de Farhenheit a Kelvin ');

writeln;

write('Elige Opcion (Pulsa F, C o K: ');

readln (opcion);

end; {De eligeOpcion}

function **Mayuscula**(x:char):char; function EsMinuscula(Minus:char):boolean; begin {De EsMin£scula} EsMinuscula:= ('a'<= Minus) and (Minus <= 'z');

end; {De EsMinúscula}

begin {De Mayuscula} if EsMinuscula(x) then Mayuscula:=chr(ord(x) + ord('A') − ord('a')) else

```
Mayuscula:=x; end; {De Mayuscula} function Conversion (opcion:char;x:real):real;
function CaF(x:real):real;
begin {de CaF}
CaF:= (9/5) * x + 32;
end; {de CaF}
function FaC(x:real):real;
begin {de FaC}
FaC:= (5/9) * (x − 32.0);
end; {De FaC}
function FaK(x:real):real;
var
C:real;
begin {de FaK}
C:=FaC(x);
FaK:= C+273;
end; {De FaK}
begin {De conversion}
opcion:=Mayuscula(opcion);
writeln;
if opcion='F' then
Conversion:=CaF(x)
else
if (opcion='C') then
Conversion:=FaC(x)
else
if (opcion='K') then
```

```
Conversion:=FaK(x)
else
Conversion:=0;
end; {De Conversion}
procedure Mostrar (opcion:char;x,xConvertido:real);
begin {De mostrar}
opcion:=Mayuscula(opcion);
writeln;
if opcion='F' then
write ('Se Convierte de Celsius a Farhenheit: ', x:4:2, ' ºC = ',xConvertido:4:2, ' ºF')
else
if (opcion='C') then
write ('Se Convierte de Farhenheit a Celsius: ', x:4:2, ' ºF = ',xConvertido:4:2, ' ºC')
else
if (opcion='K') then
write ('Se Convierte de Farhenheit a Kelvin: ', x:4:2, ' ºF = ',xConvertido:4:2, ' K')
else
writeln ('Te has confundido. Debes pulsar F,C o K. El resultado es erróneo')
end; {De Conversion}
begin {Programa Principal}
clrscr;
TomaDato(x);
EligeOpcion(opcion);
xConvertido:=Conversion(opcion,x);
Mostrar (opcion,x,xConvertido);
end. {del Programa Principal}
```

Problema 4.5

Escribir un programa que acepte un valor por teclado, muestre el siguiente menú por patalla:

F Para convertir de Celsius a Farhenheit C Para convertir de Farhenheit a Celsius

K Para convertir de Farhenheit a Kelvin

Elige Opcion (Pulsa F, C o D):

Y termine mostrando el valor introducido, la conversión realizada y el resultado de la conversión.

Solución:

Esta nueva versión debe utilizar estructura CASE. Será excatamente igual que el ejercicio 4.4, cambiando la función Conversion y el procedimiento mostrar, como se muestran a continuación:

Modificación de Mostrar

procedure **Mostrar** (opcion:char;x,xConvertido:real); begin {De mostrar} opcion:=Mayuscula(opcion);

writeln;

case opcion of

'F': write ('Se Convierte de Celsius a Farhenheit: ', x:4:2, ' ºC = ',xConvertido:4:2, ' ºF');

'C': write ('Se Convierte de Farhenheit a Celsius: ', x:4:2, ' ºF = ',xConvertido:4:2, ' ºC');

'K': write ('Se Convierte de Farhenheit a Kelvin: ', x:4:2, ' ºF = ',xConvertido:4:2, ' K');

{ Alternativa para Turbo Pascal

else

writeln ('Te has confundido. Debes pulsar F,C o K.')

}

end; {del case }

if (opcion<>'F') and (opcion <> 'C') and (opcion<>'K') then writeln ('Te has confundido. Debes pulsar F,C o K.') end; {De Mostrar}

Modificación de Conversion

function **Conversion** (opcion:char;x:real):real;

function **CaF**(x:real):real;

begin {de CaF}

```
CaF:= (9/5) * x + 32;
end; {de CaF}
function FaC(x:real):real;
begin {de FaC}
FaC:= (5/9) * (x − 32.0);
end; {De FaC}
function FaK(x:real):real;
var
C:real;
begin {de FaK}
C:=FaC(x);
FaK:= C+273;
end; {De FaK}
begin {De conversion}
opcion:=Mayuscula(opcion);
writeln;
case opcion of
'F': Conversion:=CaF(x);
'C': Conversion:=FaC(x);
'K': Conversion:=FaK(x);
{ Opcion de Turbo Pascal
else
Conversion:=0;
}
end; {del case }
if (opcion<>'F') and (opcion <> 'C') and (opcion<>'K') then
```

```
Conversion:=0;

end; {De Conversion}
```

NOTA: Las estructuras repetitivas ofrecerán una versión mejorada del programa.

Problema 4.6

Escribir un programa PASCAL que acepte caracteres por teclado hasta introducir el carácter @, y muestre por pantalla el número de caracteres que se han introducido excluyendo el carácter centinela (@).

```
program ContadorCaracteres (input,output);

{ Acepta caracteres por teclado, seguidos de <enter> hasta introducir el carácter @

Muestra el número de caracteres introducidos, excluyendo el centinela.

}

uses

crt;

var

numCar:integer;

procedure Cuenta (var numCar:integer);

{ c debe tener un valor inicial conocido antes de entrar en el ciclo.

Se escribe readln (c). Cada carácter debe seguir de su retorno de línea.

}

const

centinela='@';

var

c:char;

begin {de cuenta}

numcar:=0; {aseguramos este valor inicial}

writeln ('Introduce caracteres. (Pulsa ',centinela,') para terminar');

writeln;

readln(c);
```

```
while c<>centinela do

begin

numCar:=numCar+1; { Si entra ya hemos incluido uno válido}

readln(c)

end; {del while}

end; { de cuenta}

begin {Programa Principal}

clrscr;

Cuenta (numCar);

writeln ('El número de caracteres es: ',numCar);

end. {del Programa Principal}
```

Problema 4.7

Escribe un programa en PASCAL que acepte por teclado dos valores enteros como límites del intervalo cerrado [CotaInferior,CotaSuperior]. Muestre por pantalla los ordinales y los caracteres ascii de los números comprendidos en este intervalo.

```
program TablaAscii (input,output); {1.Acepta por teclado dos valores enteros comprendidos entre (0−256)
2.Muestra en pantalla el ordinal y el código ascii de los números comprendidos en el intervalo.

}

uses crt;

var CotaInferior, CotaSuperior:integer;

procedure TomaDatos(var CotaInferior,CotaSuperior:integer); {Esta versión de TomaDatos puede ser
mejorada aceptando sólo valores que estén comprendidos entre 0 y 255. Pensar en un REPEAT UNTIL

PostCondicion: [ Input=[CotaInferior,CotaSuperior] CotaInferior,CotaSuperior] "  CotaInferior=ci "
CotaSuperior=cs " 0"ci"255 " 0"cs"255 } begin write ('Introduce Cota Inferior (0−255): ');

readln(CotaInferior);

write ('Introduce Cota Superior (0−255): ');

readln(CotaSuperior);

end;

procedure EscribeASCII(CotaInferior,CotaSuperior:integer);
```

```
{Precondicion: CotaInferior,CotaSuperior] "  "

COtaInferior=ci " CotaSuperior=cs

" 0"ci"255 " 0"cs"255

}

var

cont:integer;

begin

clrscr;

cont:=CotaInferior;

writeln ('Valor Ordinal Código ASCII');

writeln ('−−−−−−−−−−−−− −−−−−−−−−−−−');

writeln;

while cont<=CotaSuperior do { Precondicion: cont"CotaSuperior Invariante: Output=[cont,chr(cont)]
Postcondicion:cont=cont+1 } begin writeln (' ',cont, ' ',chr(cont));

cont:=cont+1 {modificador del contador}

end;

end;{ de EscribeAscii}

begin {Programa Principal} clrscr; TomaDatos(CotaInferior,CotaSuperior);
EscribeAscii(CotaInferior,CotaSuperior) end.{Del Programa Principal}
```

Problema 4.8

Programa que permita a un jugador jugar un número indeterminado de veces AdivinaNumero que consiste en lo siguiente:

1.− genere un número aleatoriamente comprendido entre 0−100

2.− permita al jugador adivinar el número generado, permitiéndole introducir por teclado números mientras que no lo acierte. Con cada número introducido se le informará, por pantalla, si el número generado es mayor o menor.

3.− El programa cuenta el número de intentos y muestra al jugador en pantalla el momento de la victoria, con el número de intentos que ha utilizado. Si el número de intentos es:

<= 5 se indica al jugador que es BUENO 5 < número de intentos < 15 se indica al jugador que es REGULAR

el número de intentos es > 15 se invita al jugador a que se entrene por ser MALO.

NOTA: utilizar la funcionn random(N) que acepta un número entero N, (semilla) y devuelve aleatoriamente un número entero comprendido entre 0 y N. Está definido en la librería crt (para MSDOS) y wincrt (para WINDOWS)

program **AdivinaN**(input,output); { Permita a un jugador jugar un número indeterminado de veces al juego AdivinaNumero que consiste en lo siguiente: 1.− Genera un número aleatorio comprendido entre 0−100 2.− Permite al jugador adivinar el numero, indicándole por pantalla por cada número introducido, si el generado es mayor o menor. En el momento de la victoria, muestra el número de intentos utilizados.

Si el numero de intentos es <= 5 se indica al jugador que es BUENO

Si 5 < numero de intentos < 15 se indica al jugador que es REGULAR

Si el numero de intentos es > 15 se invita al jugador a que se

entrene por ser MALO.

}

```
uses

crt;

const

semilla=100;

var

numero,

intentos:integer;

jugar:char;

procedure Clasifica(intentos:integer);

begin

case intentos of

1..5: writeln('Eres Bueno');

6..15:writeln ('Eres regular');

15..100:writeln('Eres Malo. Debes jugar más');

end;

end;
```

```
function Mayuscula(x:char):char;
function EsMinuscula(Minus:char):boolean;
begin {De EsMinúscula}
EsMinuscula:= ('a'<= Minus) and (Minus <= 'z');
end; {De EsMinúscula}
begin {De Mayuscula}
if EsMinuscula(x) then
Mayuscula:=chr( ord(x) + ord('A') − ord('a'))
else
Mayuscula:=x;
end; {De Mayuscula}
function Adivina (numero:integer):integer;
var
intentos,
prueba:integer;
jugar:boolean;
begin
intentos:=0;
prueba:=semilla+1;{Se asegura de forma artificial que entra 1 vez. Estudia la alternativa REPEAT..UNTIL}
while prueba <> numero do
begin
write ('Introduce Número: ');
readln (prueba);
if numero > prueba
then writeln (' El número es Mayor que ',prueba)
else
```

```
if numero < prueba then
writeln (' El número es Menor que ',prueba);
intentos:=intentos+1;
end;
adivina:= intentos;
end;
begin {Programa Principal}
clrscr;
write ('Quieres Jugar (S/N): ');
readln (jugar);
jugar:=Mayuscula(jugar);
while (jugar='S') do
begin
clrscr;
writeln ('******** A JUGAR ********');
writeln ('***** ADIVINA NUMERO*****');
writeln ('_____');
writeln;
numero:=random(semilla); {definido en crt}
intentos:=adivina (numero);
writeln;
writeln ('*********VICTORIA*********');
writeln;
writeln ('N§ de Intentos = ',intentos:4);
Clasifica (intentos);
write ('Quieres Jugar (S/N): ');
```

readln (jugar);

jugar:=Mayuscula(jugar);

end;

end. {del Programa Principal}

Problema 4.9

Modificación de 4.4. Utiliza las estructuras repetitivas para obligar a que se cumplan las postcondiciones de los subprogramas TomaDatos y EligeOpcion. Observa como se simplifica la definición de otros suprogramas, al asegurar que los datos son correctos

program ConvierteTemperaturas (input,output); { 1.− Acepta por pantalla un valor real

2.− Muestra en pantalla el siguiente men£:

F Para convertir de Celsius a Farhenheit

C Para convertir de Farhenheit a Celsius

K Para convertir de Farhenheit a Kelvin

Elige Opcion (Pulsa F, C o D):

3.− Muestra en pantalla el valor introducido, la operaci¢n elegida y el resultado de la conversión elegida sabiendo que ºC= K + 273

ºC= (5/9) * (ºF − 32.0)

}

uses crt; var

x,

xConvertido:real;

opcion:char;

procedure **TomaDato**(var x:real); {Poscondicion: Input=n y n >= −459.69 y x=n} const CeroAbsoluto=−459.69; begin repeat writeln ('Introduce valor: ');

readln (x);

if x< CeroAbsoluto then

begin

writeln ('Te has confundido');

```
writeln;
end;
until x >= CeroAbsoluto;
end;
procedure EligeOpcion(var opcion:char);
{Poscondicion: Input=c y (c='F' o c='C'o c='k') y opcion=c}
function Mayuscula(x:char):char;
function EsMinuscula(Minus:char):boolean;
begin {De EsMinúscula} EsMinuscula:= ('a'<= Minus) and (Minus <= 'z');
end; {De EsMin£scula}
begin {De Mayuscula}
if EsMinuscula(x) then
Mayuscula:=chr( ord(x) + ord('A') − ord('a'))
else
Mayuscula:=x;
end; {De Mayuscula}
begin {De Elige Opcion}
repeat
writeln;
writeln ('F para convertir de Celsius a Farhenheit');
writeln ('C para convertir de Farhenheit a Celsius ');
writeln ('K para convertir de Farhenheit a Kelvin ');
writeln;
write('Elige Opcion (Pulsa F, C o K: ');
readln (opcion);
opcion:=Mayuscula(opcion)
```

until (opcion='F') OR (opcion='C')OR (opcion='K')

end; {de EligeOpcion}

function **Conversion** (opcion:char;x:real):real;

{Precondición: opcion='F' " opcion='C' " opcion='K'} function CaF(x:real):real; begin {de CaF} CaF:= (9/5) * x + 32; end; {de CaF} function FaC(x:real):real;

begin {de FaC}

FaC:= (5/9) * (x − 32.0);

end; {De FaC}

function FaK(x:real):real;

var

C:real;

begin {de FaK}

C:=FaC(x);

FaK:= C−273;

end; {De FaK}

begin {De conversion}

writeln;

case opcion of

'F': Conversion:=CaF(x);

'C': Conversion:=FaC(x);

'K': Conversion:=FaK(x);

end; {del case}

end; {De Conversion}

procedure **Mostrar** (opcion:char;x,xConvertido:real);

{Precondición: opcion='F' " opcion='C' " opcion='K'} begin {De mostrar} writeln; case opcion of 'F': write ('Se Convierte de Celsius a Farhenheit: ',

x:4:2, ' ºC = ',xConvertido:4:2, ' ºF');

```
'C': write ('Se Convierte de Farhenheit a Celsius: ', x:4:2, ' ºF = ',xConvertido:4:2, ' ºC');

'K': write ('Se Convierte de Farhenheit a Kelvin: ', x:4:2, ' ºF = ',xConvertido:4:2, ' K');

end; {del case}

end; {De Mostrar}

begin {Programa Principal}

clrscr;

TomaDato(x);

EligeOpcion(opcion);

xConvertido:=Conversion(Opcion,x);

Mostrar (opcion,x,xConvertido);

end. {del Programa Principal}
```

Problema 4.10

Aborda el Ejemplo 4.6. Contador de Caracteres con una estructura repeat until. Evita la lectura anticipada.

```
program ContadorCaracteres (input,output);

{ Acepta caracteres por teclado, seguidos de <enter> hasta introducir el

carácter '@'

Muestra el n£mero de caracteres introducidos, excluyendo el centinela.

}

uses

crt;

var

numCar:integer;

procedure Cuenta (var numCar:integer);

{ c debe tener un valor inicial conocido antes de entrar en el ciclo.

Se escribe readln (c). Cada car cter debe segir de su retorno de línea.

}
```

```
const

centinela='@';

var

c:char;

begin

numcar:=−1; {aseguramos este valor inicial}

writeln ('Introduce caracteres. (Pulsa ',centinela,') para terminar');

writeln;

repeat

readln(c);

numcar:=numcar+1;

until c=centinela;

end; { de cuenta}

begin {Programa Principal}

clrscr;

Cuenta (numCar);

writeln ('El número de caracteres es: ',numCar);

end. {del Programa Principal}
```

Problema 4.11.

Modifica el ejemplo 4.7. TabASCII.pas de forma que se cumplan las postcondiciones del subprograma TomaDatos. Asegura la precondición de EscribeAscii

```
procedure TomaDatos(var CotaInferior,CotaSuperior:integer); { PostCondicion: [
Input=[CotaInferior,CotaSuperior] CotaInferior,CotaSuperior] " N COtaInferior=ci y CotaSuperior=cs
CotaInferior<CotaSuperior

y 0"ci"255 y 0"cs"255

}

begin

{ Está entre llaves porque a continuación se ofrece otra alternativa
```

repeat

write ('Introduce Cota Inferior (0−255): ');

readln(CotaInferior);

write ('Introduce Cota Superior (0−255): ');

readln(CotaSuperior);

until (cotaInferior>=0) and (cotaInferior<=255)

and (cotaSuperior >=0)

and (cotaSuperior <= 255)

and (CotaInferior<CotaSuperior)

}

{ Otra Alternativa para TomaDatos:

repeat anidados. Trata independientemente las cotas y luego en conjunto

} repeat

repeat write ('Introduce Cota Inferior (0−255): '); readln(CotaInferior); until (cotaInferior>=0) and (cotaInferior<=255); writeln;

repeat

write ('Introduce Cota Superior (0−255) y mayor que CotaInferior: ');

readln(CotaSuperior);

until (cotaSuperior >=0) and (cotaSuperior <= 255)

until (CotaInferior<CotaSuperior)

end; {de TomaDatos}

Problema 4.12.

Modifica el procedimiento EscribeASCII del Ejemplo 4.7, utilizando ahora la estructura repetitiva FOR

procedure EscribeASCII(CotaInferior,CotaSuperior:integer); {Precondicion: CotaInferior,CotaSuperior] "N y CotaInferior=ci y CotaSuperior=cs CotaInferior<CotaSuperior

y 0"ci"255 y 0"cs"255

}

```
var

cont:integer;

begin

clrscr;

writeln ('Valor Ordinal C¢digo ASCII');

writeln ('−−−−−−−−−−−−− −−−−−−−−−−−−');

writeln;

for cont:=CotaInferior to CotaSuperior do

{ Precondicion: cont= CotaInferior y cotaInferior"CotaSuperior

Invariante: cont<CotaSuperior Postcondicion: Output=[cont,chr(cont)] } writeln (' ',cont, ' ',chr(cont)); end;{ de Escribe Ascii}
```

Problema 4.13.

Codifica un programa PASCAL que calcule el factorial de un número entero positivo, comprendido entre 0−15. El valor se introduce por teclado y se muestra el resultado por pantalla. Utiliza un método iterativo.

NOTA: El factorial de 0 es 1

Ejemplos de entradas:

- Introduce Número (0−15): 2
- Introduce Número (0−15): −2
- Introduce Número (0−15): 16

Ejemplos de Salidas correspondientes:

- El factorial de 2 es 4
- El número debe ser positivo

Introduce Número (0−15):

- El número debe estar en el intervalo [0,15]

Introduce Número (0−15):

```
program FactorialIterativo (input,output); {Calcula el factorial de un número entero positivo}

uses crt; var x, f:integer;

procedure TomaDato (var x:integer); {Postcondición: Input=n " n>=0" n<=15 " x=n }

begin
```

```
repeat
write ('Introduce Número Entero Positivo (0−15): ');
readln (x); {Asegura la salida del bucle}
if x<0 then
writeln ('El número ha de ser entero positivo')
else
if x>15 then
writeln ('El número ha de estar en el intervalo [0,15]');
writeln;
until (x>=0) and (x<=15);
end; {de TomaDato}
function Factorial (x:integer):integer;
{Precondición: 0 "x " 15
Postcondicion: factorial= x!
}
var
i,
f:integer;
begin f:=1; for i:=x downto 1 do {Precondición f=1 " i=x " i>=1 Invariante de ciclo f=f*i}
f:=f*i;
factorial:=f;
end; {de factorial}
begin {Programa Principal}
clrscr;
TomaDato (x);
f:=Factorial(x);
```

```
writeln ('El factorial de ',x,' = ',F);

end. {del Programa Principal}
```

Ejemplo 4.14

Escribe un programa en PASCAL, utilizando subprogramación y las instrucciones estructuradas, que permita mostrar las tablas de verdad de tres variables para las operaciones AND, OR, y NOT, según las opciones del menú.

El programa ha de iniciar con el siguiente menú. El usuario introduce su opción y el programa no aceptará otra opción que las dispuestas en el menú (1−4).

1.− Mostrar la Tabla AND

2.− Mostrar la Tabla OR

3.− Mostar la Tabla NOT

4− Mostrar las tres Tablas

Elige Opcion (1−4):

Una vez mostrada la tabla elegida por el usuario, el programa volverá a solicitar de éste si desea mostrar otra tabla. En caso afirmativo, se volverá a mostrar el menú tantas veces como tablas quiera ver el usuario.

```
program TablasLogicas (input,output); {Muestra las tablas lógicas para tres variables} uses crt; var
opcion:integer;

seguir:char;

function Mayuscula(x:char):char;

function EsMinuscula(Minus:char):boolean;

begin {De EsMin£scula}

EsMinuscula:= ('a'<= Minus) and (Minus <= 'z');

end; {De EsMin£scula}

begin {De Mayuscula}

if EsMinuscula(x) then

Mayuscula:=chr( ord(x) + ord('A') − ord('a'))

else

Mayuscula:=x;

end; {De Mayuscula}
```

```
procedure TOR ;
var
x,y,z:boolean;
begin
{Escribe la cabecera de la tabla}
writeln;
writeln ('Tabla OR');
writeln ('________');
writeln;
writeln ('|X|':5,'|Y|':10,'|Z|':10,'|X or Y or Z|':20);
writeln ('_____________________________________________');
{Inicializa los valores de las variables}
x:=false;
y:=false;
z:=false;
{Escribe la primera l¡nea de la tabla}
writeln (X:5, Y:10, Z:10, X or Y or Z:20 );
{Escribe el resto de las lneas de la tabla }
repeat
{cambio de valores de variables}
if x and y then
z:=not z;
if x then
y:= not y;
x:=not x;
writeln (X:5, Y:10, Z:10, X or Y or Z:20 );
```

```
until x and y and z;
writeln;
end;
procedure TAND ;
var
x,y,z:boolean;
begin
{Escribe la cabecera de la tabla}
writeln;
writeln ('Tabla AND');
writeln ('_________');
writeln;
writeln ('|X|':5,'|Y|':10,'|Z|':10,'|X and Y and Z|':20);
writeln ('_____________________________________________');
{Inicializa los valores de las variables}
x:=false;
y:=false;
z:=false;
{Escribe la primera l¡nea de la tabla}
writeln (X:5, Y:10, Z:10, X and Y and Z:20 );
{Escribe el resto de las l¡neas de la tabla }
repeat
{cambio de valores de variables}
if x and y then
z:=not z;
if x then
```

```
y:= not y;

x:=not x;

writeln (X:5, Y:10, Z:10, X and Y and Z:20 );

until x and y and z;

writeln;

end;

procedure TNOT;

var

x,y,z:boolean;

begin

writeln;

writeln ('Todas las NOT');

writeln ('_____');

writeln;

writeln ('|X|':5,'|Y|':10,'|Z|':10,'|NOT X|':10,'NOT Y|':10,'NOT Z|':10);

writeln ('_____');

x:=false;

y:=false;

z:=false;

writeln (X:5, Y:10, Z:10, NOT X:10,NOT Y:10,NOT Z:10 );

repeat

if x and y then

z:=not z;

if x then

y:= not y;

x:=not x;
```

```
writeln (X:5, Y:10, Z:10, NOT X:10,NOT Y:10,NOT Z:10 );
until x and y and z;
writeln;
end;
procedure TTodas;
var
x,y,z:boolean;
begin
writeln;
writeln ('Todas las Tablas');
writeln ('_________________');
writeln;
writeln ('|X|':5,'|Y|':7,'|Z|':7,'|XorYorZ|':11, '|XandYandZ|':13,'|NOT X|':9,'|NOT Y|':9,'|NOT Z|':8);
writeln ('_____________________________________________________________________');
x:=false;
y:=false;
z:=false;
writeln (X:5, Y:7, Z:7, x or y or Z:11,X and Y and Z:13, NOT X:9,NOT Y:9,NOT Z:8 );
repeat
if x and y then
z:=not z;
if x then
y:= not y;
x:=not x;
writeln (X:5, Y:7, Z:7, X or Y or Z:11,X and Y and Z:13, NOT X:9,NOT Y:9,NOT Z:8 );
until x and y and z;
```

```
writeln;
end;
begin {Programa Principal}
clrscr;
repeat
{Muestra menu}
repeat
writeln ('1.− Tabla OR');
writeln ('2.− Tabla AND');
writeln ('3.− Tabla NOT');
writeln ('4.− Las tres tablas');
writeln;
write ('Elige Opcion: ');
readln (Opcion);
until (opcion >=1) and (opcion <=4);
{Ejecuta opcion}
{Precondicion: 1<=opcion<=4}
case opcion of
1: TOR ;
2: TAND ;
3: TNOT ;
4: TTodas ;
end;
writeln;
write ('Si quieres ver otra tabla pulsa S ');
readln(seguir);
```

```
seguir:=Mayuscula(seguir);
until (seguir<>'S')
end. {del Programa Principal}
```

Ejemplo 4.15

Modifica el ejemplo 4.13, incluyendo precondiciones, postcondiciones, invariantes y límite de ciclo

```
program FactorialIterativo (input,output);
{Calcula el factorial de un número entero positivo}
uses
crt;
var
x,
f:integer;
procedure TomaDato (var x:integer);
{Postcondición: Input=n y n>=0 y x=n }
begin
repeat
write ('Introduce Número Entero Positivo (0−15): ');
readln (x); {Asegura la salida del bucle}
if x<0 then
writeln ('El número ha de ser entero positivo')
else
if x>15 then
writeln ('El número ha de ser menor que 16');
writeln;
until (x>=0) and (x<=15);
{Poscondicion del ciclo:Input=n y 0<=n<=15 y x=n }
```

```
end; {de TomaDato}

function Factorial (x:integer):integer;

{Precondición: 0<=x<=15

Poscondición: factorial= x!

}

var

i,

f:integer;

begin

f:=1;

{Precondición de ciclo: f=1 y i=x i>=0}

for i:=x downto 0 do

begin

{Invariante de ciclo:0<=i<=x+1 " ( f=f*1 " f=f*i )

Límite de Ciclo: factores por multiplicar = i}

if i=0 then

f:=f*1

else

f:=f*i;

writeln ('Me quedan ',i);

end;{del for}

factorial:=f;

{Postcondicion de ciclo:f=x!}

end; {de factorial}

begin {Programa Principal} clrscr; TomaDato (x); f:=Factorial(x); writeln (x,'!=',F,' El factorial de ',x,' = ',F);
end. {del Programa Principal}
```

Ejemplo 4.16

Modifica el ejemplo 4.15. Elimina del bucle la ejecución del factorial de 0 y observa ahora el número de iteraciones.

function Factorial (x:integer):integer; {Precondición: 0<=x<=15 Postcondicion: factorial= x! } var i, f:integer;

begin

f:=1;

{**Precondici¢n de ciclo**: f=1 y i=x i>=0}

for i:=x downto 1 do

begin

{**Invariante de ciclo**:1<=i<=x+1 " f=f*i

Límite de Ciclo: factores por multiplicar = i−1}

f:=f*i;

writeln ('Me quedan ',i−1);

end;{del for}

factorial:=f;

{**Postcondicion de ciclo**:f=x!}

end; {de factorial}

Ejemplo 4.17

Utiliza las estructuras repetitivas para escribir una función en PASCAL que pasándole un real (n) y un entero (e) devuelva la potencia ne

function **Eleva** (n:real;e:integer):real;

function elevaexppos (n:real;e:integer):real;

{Precondici¢n e>=0

Postcondici¢n producto=n ** e}

var

producto:real;

begin

producto:=1;

```
{Precondici¢n: e>0 y Producto=n ** 0

Postcondici¢n: Producto=n ** e }

while (e>0) do

{Invariante: e>0

L¡mite de Ciclo: e−1}

begin

producto:=producto*n;

writeln ('Me quedan ',e−1, ' pasos');

e:=e−1;

end;

elevaexppos:=producto;

end; {De elevaexppos}

{Como alternativa podríamos tener definida la función

elevaexpneg con la funcionalidad de la instrucción del caso

−99..−1

}

{

function elevaexpneg (n:real;e:integer):real;

var

producto:real;

begin

elevaexpneg:=1/elevaexppos(n,abs(e));

end;

}

var {de eleva}

ne:real;
```

```
begin

case e of

−99..−1:eleva:=1/elevaexppos(n,abs(e));

0..99:eleva:=elevaexppos(n,e);

end;

end; {de eleva}

begin {Programa Principal}

TomaDatos(n,e);

resultado:=Eleva(n,e);

writeln (n:5:2,' ** ',e, ' es ',resultado:5:2);

end. {del Programa Principal}
```

Ejemplo 4.18

Utiliza las estructuras repetitivas para escribir una función en PASCAL que pasándole dos enteros devuelva su máximo común divisor

```
function Calcula (x1,x2:integer):integer;

{Si x1<x2, se ejecuta una iteración más (Cambia de orden y sigue)}

{Precondición de Calcula: (x1 >0) " (x2 > 0)}

var

resto:integer;

begin

repeat

{Invariante de Ciclo: 0 " x2 " max(x1,x2)

y 0 " x1 " max(x1,x2)

Límite de Ciclo: x2

}

resto:=x1 mod x2;

x1:=x2;
```

x2:=resto;

until x2=0;

{Postcondicion: x2=0 y Calcula(x1,x2)=Calcula(x1,0)

y x1" max(x1,x2) }

calcula:=x1

end;

NOTA: Cada iteración consiste en sustituir x1 por x2 y x2 por x1 mod x2

Ejecuciones Calcula (25,15) y Calcula (15,25)

Paso	**X1**	**X2**	**resto**
0	25	15	?
1	15	10	10
2	10	5	5
3	5	0	0
Paso	**X1**	**X2**	**resto**
0	15	25	?
1	25	15	15
2	15	10	10
3	10	5	5
4	5	0	0