**Fundamentos de la programacio´n y la inform´atica Examen pr´actico. 21 de Diciembre de 2018** Grado en ingenier´ıa aeroespacial en aeronavegaci´on

Grado en ingenier´ıa aeroespacial en veh´ıculos aeroespaciales Universidad Rey Juan Carlos

## Preparativos

Ejecuta el script prepara\_fpi

Esto crear´a en tu cuenta el directorio ~/fpi.diciembre.18

y los ficheros

~/fpi.diciembre.18/coordenadas\_aeropuertos.txt y

~/fpi.diciembre.18/cuadrado.pas

## Ejercicio 1 (3 puntos)

Escribe un programa en Pascal en el fichero ~/fpi.diciembre.18/cuadrado.pas que escriba en pantalla un cuadrado, de taman˜o *lado*, construido con los caracteres almohadilla y espacio, segu´n la siguiente especificaci´on:

1. El programa pedir´a al usuario un nu´mero entero entre 1 y 40, ambos inclusive. Si la entrada no es un nu´mero entero o est´a fuera de este rango, mostrar´a un error y repetir´a la petici´on, todas las veces necesarias hasta que la entrega sea correcta.

En el fichero tienes procedimiento que puedes aprovechar, aunque tendr´as que modificarlo.

1. El programa tendr´a un procedimiento llamado *escribe cuadrado*, con un par´ametro entero lla- mado *lado*.
2. Este procedimiento comprobar´a (de nuevo) que *lado* sea un nu´mero mayor o igual que 1, menor o igual que 40. Si el valor est´a fuera de este rango, mostrar´a un mensaje de error y finalizar´a la ejecuci´on del programa.
3. Si *lado* vale 1, el programa escribir´a

*#*

1. Si *lado* vale 2, el programa escribir´a

*# #*

*# #*

1. Si *lado* vale 3, el programa escribir´a

*# # # # #*

*# # #*

1. Para el resto de valores de *lado*, el programa se comportar´a de forma semejante.

Sugerencia: haz que tu programa considere los valores 1 y 2 como casos particulares, en ese caso escribir´a el cuadrado usando solamente sentencias *writeln*. Para los valores entre 3 y 40, que considere el caso general, donde escriba el lado superior, los lados laterales y el lado inferior.

## Soluci´on

*{£mode objfpc}{£H-}{£R+}{£T+}{£Q+}{£V+}{£D+}{£X-}{£warnings on}*

program cuadrado;

const

LadoMinimo = 1;

LadoMaximo = 40;

procedure lee\_lado(var lado:integer); var

sal : boolean = FALSE; s : string;

codigo : integer; begin

repeat

write(’Introduce un n´umero entero entre ’, LadoMinimo); writeln(’ y ’, LadoMaximo);

readln(s) ;

val(s, lado, codigo);

if (codigo = 0 ) and (lado >= 1) and (lado <= 40) then begin sal := True;

end else

writeln(’Error, ’,s,’ no es un entero entre 1 y 40’); until sal;

end;

procedure barra\_horizontal(lado: integer; tinta:string); var

i : integer; begin

for i := 1 to lado do begin write(tinta);

end; writeln;

end;

procedure barras\_verticales(lado:integer; tinta, papel:string);

*// Las barras verticales est´an compuestas de lado-2 l´ıneas*

*// Cada l´ınea es un punto de tinta, lado-2 espacios y otro punto*

var

i, j : integer;

begin

for i := 1 to lado-2 do begin write(tinta);

for j := 1 to lado-2 do begin write(papel);

end; write(tinta); writeln;

end; end;

procedure escribe\_cuadrado(lado:integer; tinta,papel:string); begin

if (lado < LadoMinimo) or (lado > LadoMaximo) then begin write(’Error. Lado vale ’, lado);

writeln(’ . Deber´ıa estar entre ’, LadoMinimo , ’ y’ , LadoMaximo); halt;

end;

if (lado = 1) then writeln(’# ’)

else if (lado = 2) then begin writeln(’# # ’);

writeln(’# # ’); writeln;

end

else begin

barra\_horizontal(lado, tinta); barras\_verticales(lado, tinta, papel); barra\_horizontal(lado, tinta);

end end;

var

lado : integer = 0; tinta : string = ’# ’; papel : string = ’ ’;

begin

lee\_lado(lado); escribe\_cuadrado(lado, tinta, papel);

end.

Observaciones:

El caso de lado 2 podr´ıa generalizarse tambi´en: una l´ınea horizontal, cero l´ıneas verticales y otra horizontal. Donde cada l´ınea horizontal es un punto de tinta, cero espacios y otro punto. Pero la soluci´on escrita aqu´ı es m´as f´acil de escribir y de entender.

## Ejercicio 2 (3.5 puntos)

En la pr´actica 8.4, fichero ~/fpi/practica08/ordena.pas, ordenaste un array sencillo donde cada elemento era un nombre y una puntuaci´on. En este ejercicio, crear´as un nuevo array solo con los *mejores* jugadores, que ser´an aquelos cuya puntuaci´on sea mayor o igual a una puntuaci´on m´ınima.

Copia tu programa ordena.pas en ~/fpi.diciembre.18/mejores.pas y modif´ıcalo para que siga la siguiente especificaci´on:

1. Tendr´a una funci´on *solo mejores* que

Recibir´a como par´ametros el array de registros y un entero *puntuacion minima*.

Devolver´a un array que contendr´a aquellos registros de jugadores con puntuaci´on mayor o igual a *puntuacion minima*.

Si *puntuaci´on minima* est´a fuera del rango permitido (de 0 a 15), la funci´on mostrar´a un error y el programa finalizar´a.

El array devuelto tendr´a la misma estructura que el original, esto es, ser´a del mismo tipo y marcar´a la u´ltima posici´on con una entrada especial de nombre ZZZ.

1. El cuerpo principal del programa tendr´a una variable local que usar´as para dar valor a *puntua- cion minima*. No leas ningu´n valor desde el teclado.
2. El cuerpo principal del programa mostrar´a el array original sin ordenar, el array original ordenado y el array con los mejores jugadores (desordenado).
3. En tu programa, la funci´on *solo mejores* usar´a el array original sin ordenar. Pero deber´a poder funcionar con cualquier array de este tipo, por ejemplo el array ordenado.

### Soluci´on

No publicamos una soluci´on completa porque ser´ıa tanto como publicar una implementaci´on de la pr´actica 8.4. Pero la funci´on a escribir deber´ıa parecerse a la siguiente:

function solo\_mejores( jugadores:tipo\_jugadores; puntuacion\_minima:integer

):tipo\_jugadores;

var

i : integer; *// Indice del array ’viejo’* j : integer; *// Indice del array ’nuevo’* nuevo\_array : tipo\_jugadores;

ultimo : integer;

begin

ultimo := posicion\_ultimo(jugadores); j := 1;

for i := 1 to ultimo do begin

if (jugadores[i].puntuacion >= puntuacion\_minima) then begin nuevo\_array[j] := jugadores[i];

j := j+1;

end end;

nuevo\_array[j].nombre := ’ZZZ’; result := nuevo\_array;

end;

Observaciones:

Esta funci´on recibe el array *viejo* (*jugadores*) con todos los jugadores y la puntuaci´on m´ınima para considerar a un jugador entre los mejores. Devuelve el array *nuevo*, solo con los mejores jugadores.

La variable *i* recorre el array *viejo*. Hay que tratar el array completo, todos los elementos desde la posici´on 1 hasta la posici´on de ZZZ (exclusive), por tanto usamos un bucle *for*

Usamos *j* como ´ındice para el array *nuevo*. No sabemos a priori cu´antos elementos tendr´a. Lo iniciamos a 1, y cada vez que un elemento (un jugador) tenga una puntuaci´on mayor o igual a la requerida, lo copiamos del array nuevo al viejo e incrementamos *j*.

Finalmente, an˜adimos ZZZ tras el u´ltimo elemento del array nuevo, para marcar el fin.

Alternativas: como cualquier programa, se podr´ıa hacer de muchas formas distintas, similares, peores o mejores. Comentamos aqu´ı alguna de las alternativas principales.

En este ejemplo usamos una funci´on *posicion ultimo* solamente para saber la posici´on de u´ltimo elemento, previo a ZZZ. Para un programador principiante posiblemente esto es lo m´as claro, permite usar un bucle sencillo. Un programador con m´as experiencia probablemente preferir´a recorrer la variable *i* en un bucle *while*, que se vaya incrementando mientras no se alcance la posici´on ZZZ del array *viejo*.

Si no fuera porque el enunciado pide una funci´on, tambi´en podr´ıamos haber usado un procedi- miento que reciba los dos arrays, pasando el nuevo por referencia.

## Ejercicio 3 (3.5 puntos)

En la pr´actica 9.2, fichero ~/fpi/practica09/aeropuerto\_cercano.pas, escribiste un programa que indicaba el aeropuerto m´as cercano a otro aeropuerto dado. Ahora lo modificar´as para devolver todos los aeropuertos que est´en a una distancia menor o igual que cierto valor.

Copia tu programa ~/fpi/practica09/aeropuerto\_cercano.pas en

~/fpi.diciembre.18/cercanos.pas y modif´ıcalo para que siga la siguiente especificaci´on:

1. Al igual que en la versi´on anterior del programa, se le pedir´a al usuario un c´odigo de aeropuerto, se le notificar´a que puede acabar pulsando *f*, se mostrar´a el c´odigo y el nombre del aeropuerto. En caso de error en el c´odigo, se mostrar´a un error.
2. A diferencia de la versi´on anterior, cuando el programa disponga de un c´odigo correcto, ya no se le mostrar´a al usuario el aeropuerto m´as cercano, sino que se le pedir´a que introduza una distancia, en kil´ometros. Si el valor introducido no es un nu´mero real o no es un nu´mero positivo, el programa lo notificar´a y volver´a a pedir un valor num´erico real, todas las veces que sea necesario hasta recibir un nu´mero v´alido. Consideramos *positivo* todo nu´mero estrictamente mayor que 0.
3. El programa escribir´a en pantalla todos los aeropuertos (c´odigo y nombre) que est´en a una distancia menor o igual que la indicada por el usuario.
4. En todo lo dem´as, este programa ser´a igual que aeropuerto\_cercano.pas. Esto es, leer´a el mismo fichero, con la misma estructura, lo meter´a en un array, trabajar´a sobre el array, etc. Cualquier cosa que no especifique este enunciado, se entiende que es igual que la pr´actica 9.2.

No es necesario que elimines de esta version del programa los subprogramas necesarios para la versi´on anterior.

### Soluci´on

No publicamos una soluci´on completa porque ser´ıa tanto como publicar una implementaci´on de la pr´actica 9.2. Pero la funci´on a escribir deber´ıa ser similar a esta:

procedure lee\_distancia(var distancia: real); var

linea: string; num\_real: real; codigo: integer;

begin

repeat

write(’Introduce la distancia m´axima, en Kms: ’); readln(linea);

val(linea, num\_real, codigo); if (codigo <> 0) then

writeln(’Solo se admiten numeros reales. ’)

else if (num\_real <= 0.0) then

writeln(’Solo se admiten numeros reales positivos. ’) else

distancia := num\_real;

until (codigo = 0) and (num\_real > 0); end;

procedure escribe\_cercanos( var aeropuertos: TipoAeropuertos;

num\_aeropuertos: integer; dist\_maxima: real; indice\_aerop\_elegido: integer);

var

i: integer; dist: real;

begin

for i:=1 to num\_aeropuertos do begin

if i <> indice\_aerop\_elegido then begin

dist := haversine\_distance( aeropuertos[indice\_aerop\_elegido].latitud,

aeropuertos[indice\_aerop\_elegido].longitud, aeropuertos[i].latitud, aeropuertos[i].longitud);

if (dist <= dist\_maxima) then begin

write(aeropuertos[i].codigo\_IATA, ’ ’, aeropuertos[i].nombre, ’ ’); writeln(dist:0:3, ’ Kms’);

end; end;

end; end;

Alternativas:

El procedimiento *lee distancia* podr´ıa usar una variable booleana (llamada por ejemplo *sal* ) de forma que cuando valga *TRUE*, el bucle finalice. La versi´on escrita aqu´ı es m´as compacta.

El procedimiento *escribe cercanos* podr´ıa recibir el c´odigo de aeropuerto inicial y sus coordena- das. Pero la versi´on escrita aqu´ı es mejor, basta con pasar el ´ındice de ese aeropuerto en el array para que el propio procedimiento obtenga estos datos.

El procedimiento *escribe cercanos* recibe como argumento el nu´mero de aeropuertos. Tambi´en podr´ıa averiguarse desde el mismo procedimiento, bien llamando a una funci´on que lo busque o bien incluyendo la bu´squeda en el bucle principal (como indicamos en el ejercicio anterior). Pero para un programador principiante, es preferible separar estas dos cosas: por un lado la bu´squeda del u´ltimo elemento, y por otro, la bu´squeda de los elementos que cumplan la condici´on.

La sentencia *write* y la sentencia *writeln* podr´ıan mezclarse en una u´nica sentencia *writeln*. Pero para el programador principiante es preferible separarlo, de esta forma los posibles errores se localizan m´as f´acilmente.

## Fundamentos de la programacio´n y la inform´atica Examen de teor´ıa. 21 de Diciembre de 2018

Grado en ingenier´ıa aeroespacial en aeronavegaci´on

Grado en ingenier´ıa aeroespacial en veh´ıculos aeroespaciales Universidad Rey Juan Carlos

# Preparativos

Ejecuta el script prepara\_fpi

Esto crear´a en tu cuenta el directorio ~/fpi.diciembre.18 y el fichero

~/fpi.diciembre.18/teoria.TULOGIN.txt, donde TULOGIN es tu nombre de usuario en el laboratorio. Escribe tus respuestas en este fichero.

# Ejercicio 1

Sean *a*, *b*, *c*, *d*, *e* variables booleanas. Sean *x* e *y* variables reales.

Sean las expresiones

e1 := (a or b) and not ( (a and c) and (d or e) ); e2 := (x > 15) and not (y <= 23 );

1. A partir de *e1*, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre *s2*.
3. Escribe una expresi´on *n1* que sea la negaci´on de *s1*.
4. Escribe una expresi´on *n2* que sea la negaci´on de *s2*.

## Soluci´on

s1 := (a or b) and (((not a) or (not c)) or ((not d) and (not e))); s2 := (x > 15) and (y > 23);

n1 := ((not a) and (not b)) or ((a and c) and (d or e)); n2 := (x <= 15) or (y <= 23);

# Ejercicio 2

Como sabes, en un programa mal escrito nos podemos encontrar con errores de compilaci´on, errores de ejecuci´on, errores l´ogicos y defectos en la claridad del c´odigo. El siguiente programa est´a muy mal escrito. Sen˜ala y describe brevemente todos los errores y todos los defectos que veas. Deja claro si se trata de un error o un defecto, aunque no es necesario que especifiques si el error es de compilaci´on, ejecuci´on o l´ogico.

1 {$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}

2 program errores-1;

3

1. var
2. character: char;
3. marks\_table: array[1..100] of real;
4. i: integer;

8

1. function valornumerico(car´acter: char): integer;
2. begin
3. result := ord(character)-ord(’0’);
4. end;

13

1. procedure esmultiplode3(number: integer; result:boolean);
2. begin
3. result := ((number mod 3) = 0);
4. end;

18

1. function notapractnumerica(notapract: real; notaejerc: real): real;
2. begin
3. case notapract of

22 5.0:

1. notaejerc := result;
2. otherwise
3. result := 0.0;
4. end;
5. end;

28

1. function media( number1, number2: real): real;
2. begin
3. result := number1 + number2 / 2.0;
4. end;

33

1. begin
2. // Escribe True si la media de 5.0 y 7.0 es multiplo de 3
3. writeln(esmultiplode3(trunc(media(5.0, 7.0)))); 37
4. // Rellena la tabla y escribe sus elementos en orden descendente
5. for i = 100 downto 0 do
6. marks\_table[i] := notapractnumerica(5.0, 6.0 + real(i div 10));
7. write(marks\_table[i], ’ ’);
8. writeln;
9. end.

## Solucio´n

* 1. Defecto. General: Mezcla de dos idiomas, espan˜ol e ingl´es. Deber´ıa estar todo en ingl´es, o excepcionalmente, todo en espan˜ol.
  2. Error. L´ınea 2: El Identificador *Errores-1*, no es v´alido porque usa el car´acter ’-’, que no est´a permitido.
  3. Defecto. L´ınea 4: Variables globales. Estas variables deber´ıan declararse despu´es de los

subprogramas para que solo sean visibles en el cuerpo del programa principal.

* 1. Defecto. L´ınea 6: El *nu´mero m´agico* 100 deberia ser declarado como constante (como el resto de nu´meros del programa).
  2. Error. L´ınea 9: El identificador *car´acter* no es v´alido porque usa el car´acter ’´a’, que no est´a permitido.
  3. Error. L´ınea 11: Uso de la variable global *character* dentro de un *procedure*.
  4. Error. L´ınea 14: El segundo par´ametro, *result*, debe pasarse por referencia (*var result:boolean*).
  5. Error. L´ınea 21: *Case* est´a usando una variable real para seleccionar una rama u otra. Esto no est´a permitido, solo pueden usarse tipos discretos.
  6. Error. L´ınea 23: La asignaci´on debe ser al rev´es, *result := notaejerc ;*
  7. Error. L´ınea 31: La expresi´on para calcular la media es incorrecta, deber´ıa ser: *(number1*

*+ number2) / 2.0*.

* 1. Error. L´ınea 36: La llamada a *esmultiplode3()* debe hacerse con 2 argumentos, falta el segundo.
  2. Error. L´ınea 36: El argumento de un subprograma, en este caso *writeln*, no puede ser un procedimiento, porque un procedimiento no devuelve ningu´n valor.
  3. Error. L´ınea 39: El operador de asignaci´on es :=, no =. Por tanto, la asignaci´on deber´ıa ser *i := 100*, no *i = 100*.
  4. Error. L´ınea 39: El bucle for deber´ıa ser *100 downto 1*, no *100 downto 0*.
  5. Error. L´ıneas 39-42: Falta el *begin-end* del *for*. Tal y como est´a escrito el programa, la sentencia *write* est´a fuera del bucle, a pesar de que la tabulaci´on indica lo contrario.

### Fundamentos de la programacio´n y la inform´atica Examen pr´actico. 11 de junio de 2019

Grado en ingenier´ıa aeroespacial en aeronavegaci´on Grado en ingenier´ıa aeroespacial en veh´ıculos aeroespaciales

Universidad Rey Juan Carlos

## Preparativos

Ejecuta el script ~mortuno/prepara

Esto crear´a en tu cuenta el directorio ~/fpi.junio.19

y los siguiente ficheros

* ~/fpi.junio.19/ventana.pas
* ~/fpi.junio.19/aeropuertos.txt
* ~/fpi.junio.19/ejercicio\_tabla.pas

## Ejercicio 1 (3 puntos)

Escribe un programa en pascal en el fichero ~/fpi.junio.19/ventana.pas que haga un dibujo de taman˜o variable, al que llamaremos *ventana*. El taman˜o depender´a de un par´ametro al que llamaremos *hueco*. La *ventana* de *hueco* 1 ser´a la siguiente:

*# # # # # # # # # # # # # # # # # # # # #*

La *ventana* de *hueco* 2 ser´a:

*# # # # # # # # # #*

*# # # # # # # # # # # # #*

*# # # # # # # # # #*

La de *hueco* 3 ser´a

*# # # # # # # # # # # #*

*# # #*

*# # # # # # # # # # # # # # #*

*# # #*

*# # # # # # # # # # # #*

Y as´ı sucesivamente, hasta un hueco m´aximo de 20. El programa pedir´a al usuario que escriba por teclado el valor para *hueco*, un nu´mero entero entre 1 y 20. Si el usuario se equivoca, la pregunta se repetir´a todas las veces necesarias.

Con esto ya tienes informaci´on suficiente para resolver el ejercicio, pero para que resulte m´as sencillo, aqu´ı tienes instrucciones adicionales

* + 1. La figura est´a formada una sucesi´on de l´ıneas horizontales que contienen

Un elemento al que llamamos *tinta*, que es una almohadilla seguida de un espacio. Un elemento al que llamamos *papel*, que son dos espacios.

* + 1. La figura puede descomponerse como:

Una barra horizontal. Unos *barrotes*.

Otra barra horizontal.

Unos barrotes iguales a los anteriores. Otra barra horizontal.

* + 1. Los *barrotes* est´an formados por un nu´mero de l´ıneas igual a *hueco*, donde cada l´ınea est´a formada por

Tinta.

Tantos elementos *papel* como el valor de *hueco*. Tinta.

Tantos elementos *papel* como el valor de *hueco*. Tinta.

## Soluci´on

*{£mode objfpc}{£H-}{£R+}{£T+}{£Q+}{£V+}{£D+}{£X-}{£warnings on}*

program ventana;

procedure lee\_hueco(var hueco:integer); var

sal : boolean = FALSE; s : string;

codigo : integer; begin

repeat

writeln(’Introduce un n´umero entero entre 1 y 20’); readln(s) ;

val(s, hueco, codigo);

if (codigo = 0 ) and (hueco >= 1) and (hueco <= 20) then begin sal := True;

end else

writeln(’Error, ’,s,’ no es un entero entre 1 y 20’); until sal;

end;

procedure barra\_horizontal(hueco: integer; tinta:string);

var

i : integer;

begin

for i := 1 to hueco do begin write(tinta);

end; writeln;

end;

procedure barrotes(hueco:integer; tinta, papel:string);

*// Las barras verticales est´an compuestas de hueco-2 l´ıneas*

*// Cada l´ınea es un punto de tinta, hueco-2 espacios y otro punto*

var

i, j : integer;

begin

for i := 1 to hueco do begin write(tinta);

for j := 1 to hueco do write(papel);

write(tinta);

for j := 1 to hueco do write(papel);

write(tinta); writeln();

end; end;

procedure escribe\_cuadrado(hueco:integer; tinta,papel:string); begin

barra\_horizontal(hueco\*2+3, tinta); barrotes(hueco, tinta, papel); barra\_horizontal(hueco\*2+3, tinta); barrotes(hueco, tinta, papel); barra\_horizontal(hueco\*2+3, tinta);

end;

var

hueco : integer = 0; tinta : string = ’# ’; papel : string = ’ ’;

begin

lee\_hueco(hueco); escribe\_cuadrado(hueco, tinta, papel);

end.

## Ejercicio 2 (3.5 puntos)

Modifica tu pr´actica 9.2 para que el programa no pida al usuario un c´odigo de aeropuerto, sino que elija un aeropuerto al azar, entre todos los disponibles. Una vez elegido, el programa se comportar´a de la misma forma que la versi´on anterior, esto es, mostrar´a el aeropuerto m´as cercano.

El programa deber´a llamarse ~/fpi.junio.19/aleatorio.pas. Asegu´rate de que el nombre sea exactamente ese, de lo contrario, ser´a como no haberlo escrito.

## Ejercicio 3 (3.5 puntos)

En el fichero ~/fpi.junio.19/ejercicio\_tabla.pas encontrar´as un programa que genera un array de valores aleatorios. Modif´ıcalo para que calcule y muestre:

El valor m´aximo.

El valor m´ınimo.

La media aritm´etica de los valores.

Puedes organizar tu c´odigo como creas conveniente, pero ten en cuenta que se valorar´a la calidad del disen˜o. Lo u´nico que no puedes hacer es modificar los tres subprogramas que ya est´an escritos (*dado*, *escribe valores* e *inicia valores*).

## Fundamentos de la programacio´n y la inform´atica Examen de teor´ıa. 11 de Junio de 2019

Grado en ingenier´ıa aeroespacial en aeronavegaci´on

Grado en ingenier´ıa aeroespacial en veh´ıculos aeroespaciales Universidad Rey Juan Carlos

# Preparativos

Ejecuta el script prepara

Esto crear´a en tu ordenador el fichero ~/fpi/TULOGIN/teoria.txt, donde TULOGIN es tu nombre de usuario en el laboratorio. Escribe tus respuestas en este fichero.

# Ejercicio 1 (2 puntos)

Sean *a*, *b*, *c*, *d* variables booleanas. Sean *x* e *y* variables reales.

Sean las expresiones

e1 := not ( (b or c) and not (b and d or a) ); e2 := not (x <= 10) or not (y = 0);

1. A partir de *e1*, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre *s2*.
3. Escribe una expresi´on *n1* que sea la negaci´on de *s1*.
4. Escribe una expresi´on *n2* que sea la negaci´on de *s2*.

## Solucio´n

s1 := (not b and not c) or (b and d or a); s2:= (x > 10) or (y <> 0);

n1 := (b or c) and not(b and d or a); n2:= (x <= 10) and (y = 0);

# Ejercicio 2 (8 puntos)

Como sabes, en un programa mal escrito nos podemos encontrar con errores de compilaci´on, errores de ejecuci´on, errores l´ogicos y defectos en la claridad del c´odigo. El siguiente programa est´a muy mal escrito. Encuentra y describe brevemente todos los errores y todos los defectos que encuentres.

Deja claro si se trata de un error o un defecto, aunque no es necesario que especifiques si el error es de compilaci´on, ejecuci´on o l´ogico.

Obviamente, el nu´mero que aparece al principio de cada l´ınea no forma parte del progra- ma, es el nu´mero de l´ınea. Para cada error que encuentres, indica en qu´e linea est´a.

Por ejemplo, un error encontrado en el c´odigo podr´ıa describirse de la siguiente forma: *L´ınea 23: Error: El identificador ’in´existente’ no es v´alido porque contiene el car´acter ’´e’, que no est´a permitido.*

5

|  |  |  |
| --- | --- | --- |
| 1 | {$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on} |  |
| 2 | program aeropuertos\_01; |
| 3 | {Este programa inicia un array de registros conteniendo c´odigo y altitud | de |
| 4 | aeropuerto, luego suma todas las altitudes.} |  |
| 6 | const |  |
| 7 | Num\_aeropuertos = 2; |  |
| 8 | type |  |
| 9 | tipo\_aeropuerto = record |  |
| 10 | c´odigo : string; |  |
| 11 | altitud : integer; |  |
| 12 | end; |  |
| 13 | TipoAeropuertos = array[1..Num\_aeropuertos] of tipo\_aeropuerto; |  |
| 14 |  |  |
| 15 | function suma\_altitudes(aeropuertos:TipoAeropuertos): integer; |  |
| 16 | var |  |
| 17 | i, suma : integer; |  |
| 18 | begin |  |
| 19 | for i = 1 to Num\_aeropuertos-1 do begin |  |
| 20 | if aeropuertos[i].altitud > 0 then |  |
| 21 | suma := suma + aeropuertos[i].altitud; |  |
| 22 | else |  |
| 23 | suma := suma + aeropuertos[i].codigo; |  |
| 24 | writeln(’WARNING: Encontrado aeropuerto con altitud cero’); |  |
| 25 | end; |  |
| 26 | aeropuertos.result := suma; |  |
| 27 | end; |  |
| 29 | procedure inicia\_aeropuertos(var aeropuertos:TipoAeropuertos); |  |
| 30 | begin |  |
| 31 | aeropuertos[1].codigo := ’MAD’; |  |
| 32 | aeropuertos[1].altitud := 667; |  |
| 33 | aeropuertos[2].codigo := ’LHR’; |  |
| 34 | aeropuertos[2].altitud := 11; |  |
| 35 | end; |  |
| 36 |  |  |
| 37 | var |  |
| 38 | tipo\_aeropuerto : TipoAeropuertos; |  |
| 39 | begin |  |
| 40 | writeln(inicia\_aeropuertos(tipo\_aeropuerto)); |  |
| 41 | writeln(suma\_altitudes((tipo\_aeropuerto)); |  |
| 42 | end; |  |

28

## solucion

L´ınea 9: Defecto. Segu´n el criterio que seguimos un tipo deber´ıa estar escrito en Notaci´onCa- mello, es decir sin barra baja ni ningu´n otro elemento que separe las palabras y con la primera letra de cada palabra en mayu´scula.

L´ınea 10: Error. El car´acter ”´o”no es v´alido para identificadores, habr´ıa que quitarle la tilde.

L´ınea 18: Error. Falta inicializar la variable suma, concretamente con el valor 0 para que funcione correctamente.

L´ınea 19: Error. El operador de asignaci´on es :=, no =. Por tanto, la asignaci´on deber´ıa ser i := 1, no i = 1.

L´ınea 19: Error. El bucle deber´ıa ser de 1 a Num aeropuertos para recorrer el array en su totalidad.

L´ınea 21: Error. Sobra el ”;.al final de la l´ınea ya que interrumpe el if then else.

L´ınea 23: Error. No es posible sumar ”suma”que es un integer con .aeropuertos[i].codigo”que es un string.

L´ıneas 22-25: Falta el begin-end del else. Tal y como est´a escrito el programa, la sentencia writeln est´a fuera del bucle, a pesar de que la tabulaci´on indica lo contrario.

L´ınea 24: Error. Es una funci´on que tiene efectos laterales.

Linea 26: Error. Deber´ıa poner simplemente result:= suma;”, ya que es como se fija el resultado de una funci´on. Adema´s la sentencia que aparece contiene varios errores: aeropuertos no se puede modificar ya que no se ha pasado por referencia (var), y aunque se pudiese modificar el record no contiene ningu´n campo result.

Linea 38: Error. El nombre de la variable es el mismo que el del record, identificador dupli- cado.

L´ınea 40: Error. El argumento de un subprograma, en este caso writeln, no puede ser un procedimiento, porque un procedimiento no devuelve ningu´n valor.

L´ınea 41: Error. Sobra un par´entesis de apertura, deber´ıa ser writeln(suma altitudes(tipo aeropuerto));. L´ınea 42: Error. El programa debe terminar con un ”.”no un ”;”.

### Fundamentos de la programacio´n y la inform´atica Examen pr´actico. 19 de diciembre de 2019

Grado en ingenier´ıa aeroespacial en veh´ıculos aeroespaciales Universidad Rey Juan Carlos

## Preparativos

Ejecuta el script ~mortuno/prepara

Esto crear´a en tu cuenta el directorio ~/fpi.practico.dic.19

y los siguiente ficheros

* ~/fpi.practico.dic.19/aeropuertos.txt
* ~/fpi.practico.dic.19/haversine.pas

## Ejercicio 1 (4 puntos)

Escribe un programa en Pascal en el fichero ~/fpi.practico.dic.19/densidad.pas que dibuje un cuadrado de taman˜o variable y densidad variable, segu´n la siguiente especificaci´on

Primero pedir´a al usuario que indique el taman˜o del cuadrado, un entero entre 5 y 25, ambos inclusive. Repetir´a la petici´on hasta que el usuario haga lo indicado.

Luego pedir´a al usuario que indique la densidad del cuadrado, un entero entre 1 y 10, ambos inclusive. Repetir´a la petici´on hasta que el usuario haga lo indicado.

El programa escribir´a en pantalla una figura a la que llamaremos *cuadrado de densidad variable*. Estar´a formado por N filas de N puntos, donde N es el taman˜o. Cada punto podr´a ser o bien un *punto relleno* o bien un *punto vac´ıo*.

Cuando la densidad sea 10, todos los puntos estar´an rellenos. Cuando la densidad sea alta pero no 10 (por ejemplo 8), habr´a muchos puntos rellenos y algunos vac´ıos. Cuando la densidad sea baja (por ejemplo 1) habr´a muchos puntos vac´ıos y pocos puntos llenos. No permitiremos que la densidad sea nula porque entonces todos los puntos estar´ıan vac´ıos, ser´ıa un cuadrado vac´ıo que consideramos que no nos vale.

Conseguir esto es muy sencillo. Para dibujar cada punto

* 1. Generaremos un valor aleatorio entre 1 y 10 (*lanzamos un dado de 10 caras*).
  2. Si el valor obtenido es menor o igual que la densidad solicitada, escribiremos un *punto relleno*, compuesto por el car´acter X y el car´acter espacio.
  3. En otro caso, escribiremos un *punto vac´ıo*, compuesto por dos espacios en blanco.

Observa que si la densidad es 10, cualquier valor de nuestro dado de 10 caras es menor o igual que 10, por tanto dibujaremos todos los puntos. Si la densidad es por ejemplo 2, solo el 20 % de las tiradas de nuestro dado ser´an menores o iguales que 2, por tanto solo se rellenar´an el 20 % de los puntos.

Aqu´ı tienes un ejemplo de ejecuci´on, tu programa tendr´a que hacer algo similar esto:

koji@mazinger:~/fpi.practico.dic.19 ./densidad\_cuadrado Indica el tama~no del cuadrado

Introduce un n´umero entero entre 5 y 25 7

Indica la densidad

Introduce un n´umero entero entre 1 y 10 9

X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X

X X X X X X X

koji@mazinger:~/fpi.practico.dic.19 ./densidad\_cuadrado Indica el tama~no del cuadrado

Introduce un n´umero entero entre 5 y 25 12

Indica la densidad

Introduce un n´umero entero entre 1 y 10 3

X X X

X X X

X

X X X

X X X X X X X

X X X X X X X

X X X X

X X X X

X X X

### Solucio´n

[http://ortuno.es/densidad cuadrado.pas](http://ortuno.es/densidad_cuadrado.pas)

## Ejercicio 2 (6 puntos)

Este ejercicio est´a basado en tu pr´atica 9.2. Copia tu fichero ~/fpi/practica09/busca\_codigo.pas en ~/fpi.practico.dic.19/distancia.pas y haz que cumpla la siguiente especificaci´on. Todo lo que este enunciado no diga, se entiende que ha de ser igual que en la pr´actica que hiciste en el laboratorio. [1](#_bookmark0)

1. Si todo ha ido bien y el programa ha encontrado el aeropuerto cuyo c´odigo ha introducido el usuario, no solo mostrar´a sus datos, sino que indicar´a cu´al es el aeropuerto m´as cercano, a qu´e distancia se cuentra, cu´al es el m´as lejano y a qu´e distancia se encuentra.
2. Naturalmente, para el punto anterior tu programa tendr´a que quedarse con las coordenadas del aeropuerto solicitado y luego volver a recorrer el array de aeropuertos (no el fichero) para calcular la distancia entre ese aeropuerto y todos los dem´as del mundo, buscar el m´aximo y buscar el m´ınimo.
3. El aeropuerto m´as cercano a un aeropuerto es el mismo aeropuerto, que est´a a 0 Km de distancia. Pero este lo descartamos, cuando proceses todos los aeropuertos ignora el aeropuerto pedido (o lo que es equivalente, ignora la distancia si es 0).
4. Este ejercicio se parece a tu pr´actica 8.2 donde calculaste la distancia entre un punto del plano y el origen de coordenadas, para luego buscar el m´aximo y el m´ınimo. Pero observa que hay dos diferencias importantes:

En la pr´actica calculabas la distancia entre el punto y el origen de coordenadas. En este ejercicio calcular´as la distancia entre los aeropuertos del array y el aeropuerto pedido.

La tierra no es plana, es redonda. Para distancias cortas podemos ignorarlo y aplicar el teorema de Pit´agoras, pero en otro caso es necesario tener en cuenta la curvatura de la tierra. Esto lo re- suelve la *f´ormula del semiverseno*. En el fichero [https://gsyc.urjc.es/˜mortuno/haversine.pas](https://gsyc.urjc.es/~mortuno/haversine.pas) encontrar´as una funci´on que implementa esta f´ormula (*haversine* es la palabra inglesa para *semiverseno*). Comp´ılalo, pru´ebalo y copia y pega en tu ejercicio todo lo que necesites.

1Nota para alumnos repetidores: este ejercicio se parece mucho a una pr´actica del an˜o pasado, pero la estructura del array y el interface de usuario cambia un poco. Si algu´n ejercicio se corresponde exactamente con la especificacion del an˜o

pasado y no con la de este, su nota sera´ 0.

### Solucio´n

No publicamos el ejercicio 2 resuelto porque ser´ıa casi tanto como publicar la pr´actica 9.1 resuelta. Pero puedes auto-revisar tu examen: escribiendo los c´odigos AAAA, ZSY y MAD el resultado tiene que ser similar a este:

Escribe el codigo IATA del aeropuerto a buscar (3 caracteres) AAAA

C´odigo incorrecto. Por favor escribe 3 caracteres

Escribe el codigo IATA del aeropuerto a buscar (3 caracteres) ZSY

C´odigo no encontrado

Escribe el codigo IATA del aeropuerto a buscar (3 caracteres) MAD

Adolfo Su´arez Madrid-Barajas Airport. Codigo IATA: MAD. Coordenadas: 40.472 -3.563 Aeropuerto m´as pr´oximo:

Torrej´on Airport. Codigo IATA: TOJ. Coordenadas: 40.497 -3.446. Distancia: 10.29km

Aeropuerto m´as lejano:

Palmerston North Airport. Codigo IATA: PMR. Coordenadas: -40.321 175.617. Distancia: 19965.99km

**Fundamentos de la programacio´n y la inform´atica Examen de teor´ıa. 19 de diciembre de 2019** Grado en ingenier´ıa aeroespacial en veh´ıculos aeroespaciales

Universidad Rey Juan Carlos

# Preparativos

Ejecuta el script ~mortuno/prepara

Esto crear´a en tu ordenador el fichero ~/fpi.teoria.dic.19/teoria.TULOGIN.txt, don- de TULOGIN es tu nombre de usuario en el laboratorio. Escribe tus respuestas en este fichero.

# Ejercicio 1 (2 puntos)

Sean *a*, *b*, *c*, *d* variables booleanas. Sean *x* e *y* variables reales.

Sean las expresiones

e1 := not ( (b and c) or not (b or d or a) ); e2 := not (x > 15) and not (y = 0);

1. A partir de *e1*, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre *s2*.
3. Escribe una expresi´on *n1* que sea la negaci´on de *s1*.
4. Escribe una expresi´on *n2* que sea la negaci´on de *s2*.

## Solucio´n

1. (not b or not c) and (b or d or a)

2. (x <= 15 and y <> 0)

3. (b and c) or (not b and not d and not a)

4. (x > 15) or (y = 0)

# Ejercicio 2 (8 puntos)

Un programador desea resolver el siguiente problema: en una asignatura el profesor da puntos extra a los alumnos que participan activamente en clase. Algunos alumnos tienen hasta 15 puntos, pero el profesor desea que la nota *sature* a partir de 10, esto es, las notas como 11, 12, etc, ser´an reemplazadas por un 10.

As´ı que escribe el programa mostrado a continuaci´on para simular esto: genera notas al azar entre 0 y 15 (ambos inclusive), las guarda en un array y luego genera otro array filtrando el array inicial. El programador es novato y escribe el programa muy mal. Como sabes, en un programa mal escrito nos podemos encontrar con errores de compilaci´on, errores de ejecuci´on, errores l´ogicos y defectos en la claridad del c´odigo. Encuentra y describe brevemente todos los errores y todos los defectos que encuentres.

Deja claro si se trata de un error o un defecto, aunque no es necesario que especifiques si el error es de compilaci´on, ejecuci´on o l´ogico.

Obviamente, el nu´mero que aparece al principio de cada l´ınea no forma parte del progra- ma, es el nu´mero de l´ınea. Para cada error que encuentres, indica en qu´e linea est´a.

Por ejemplo, un error encontrado en el c´odigo podr´ıa describirse de la siguiente forma:

*L´ınea 2: Error: los programas en Pascal no empiezan por ’Programa’ sino por ’Program’*

01 {$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on} 02 programa filtra\_tabla;

1. uses crt;
2. const
3. tamanio\_array : 8;
4. type
5. TipoTabla : array[1..tamanio\_array] of integer; 08 var

09 tabla\_bruto, tabla\_filtrada = TipoTabla; 10

11

1. function tira\_dado(caras\_dado: integer):integer;
2. begin
3. result := random( caras\_dado ) + 1 ;
4. end;

16

17

1. procedure rellena\_tabla(var tabla:TipoTabla; valor\_maximo:integer);
2. var
3. i : integer;
4. begin
5. for i := 1 to tamanio\_array do
6. tabla[i] := tira\_dado(valor\_maximo);
7. end;

25

26

1. procedure filtra\_tabla(tabla\_entrada, tabla\_salida:TipoTabla; limite\_Saturaci´on:integer);
2. var
3. i : integer;
4. begin
5. for i = 1 to tamanio\_array do begin
6. if tabla\_entrada[i] < limite\_Saturaci´on then
7. tabla\_salida[i] := limite\_Saturaci´on;
8. else
9. tabla\_salida[i] := tabla\_entrada[i]
10. end;
11. end;

38

39

1. procedure muestra\_tabla(var tabla: TipoTabla);
2. var
3. i : integer;
4. begin
5. for i = 1 to tamanio\_array do
6. writeln(tabla[i]);
7. end;

47

48

1. const
2. ValorMaximo = 15;
3. LimiteSaturaci´on = 10;
4. begin
5. rellena\_tabla(tabla\_bruto, ValorMaximo);
6. tabla\_filtrada := filtra\_tabla(tabla\_bruto, LimiteSaturaci´on);
7. writeln(’Tabla en bruto:’);
8. muestra\_tabla(tabla\_bruto);
9. writeln(’Tabla filtrada:’);
10. muestra\_tabla(tabla\_filtrada);
11. end;

## Solucio´n:

(En la p´agina siguiente)

*Código dado en el examen (mal) Solución (bien)*

l'l2■ **1** rogrma a • f í1 t ra t atíl.a¡ l'l2■ **1** rog ram• f fl t ra tabt a ·

1')3

1')4

■

1')5

1')6

uses " e rt ;

**const**

* • • • t ma an i o\_a rray • : • 8¡ t ype

1')3 uses ·cr t ;

1')4 const

**"I** ···

* • • •

l'l5 • Tama nio A rr ay • = ·8; 1')6 t ype

**1**

**11**

12

13

14

15

16

171[

, • • • TipoTabl a" : • a r ray [ **l. .** tma anio \_array] •of, i nt eger ¡

var

·· ·· t abl a\_brut o, ·t abl a\_f i l t rada ·=· Ti pol abl a;

fo ncti on •t ir a\_d ado( ca ras \_dado : • i nt eg erl : i nt eger ; beqi n

···· res u l **t** · : = ·· random ( ·**caras** dado · ) · +· l · ; end ; -

1')8

1')9

11')

**11**

12

13

lt

**TipoTabl a· • ·amy [l .** .**T aoioMcay]** ·**of** · **i ot, g" ¡**

funct i o, n ti ra\_dado I ca ras\_dado : • i nt ege• rl : i nt e ger; beqin

· ··· res ul t · : =" · random ( · ca ras dado· l ·+ · l · ; end; -

1. proced ur e • rell ena\_t abl a{ va r • t abl a : Ti pol ablai-;
2. var

val orm\_ axmi

o:i nt eger l ;

1. proced u re • re ll ena\_t abt a{ va r • t abl a:Ti poTabl a-;
2. va r

val orm\_ axmi

o: i nt ege r) ;

21')

21

# 221

23

o l ;

, • • • i • : • i nt ege r; beqi n

···· f or· i · : =· l ·t o· t ma an i o a rray • d o·

• • • • ·• •-•

t abl a l il • : =• ti ra dadolva l orm axmi

1. • • • • i • : • i nt e ger;
2. beqin

**I**

· ··· f or· i · : =· l ·t o· Tma ani oA r ra-y do ·

• • • • • •-•

t abl a l il • : = •ti ra dado ( val orm a xmi

o ) -1 ;

24 end; - - 25

**t**

,

procedu re·fi l t ra\_t a bl a (t abl a\_ent rada , ·t abl a\_sal i da :Ti pol abl a ; ·l imi t e\_Sat u rac i ón : i nt e ge r) ;

28 va, r

* • • • f or• i ·=· l " t o • t ma an i o a r ray- do • beq i n

21')

**221**

21

23

end; - -

procedu re · f i l t ra\_t ab l a (v ar- t abl a\_ent rada , · t abl a\_sal i da : Ti poTabl a; · l imi te\_sat u raci on : i n t eger ); va r

29

31')

···· i ·-: beqi n

i nt ege r;

1. i · :- i nt e ger;
2. beqin

**;1**

• •-•

fo r· i · : = • l •t -o Tma ani oA r ra-y do • beqi n

**3;1**

34

35

* • . . ,. • • · i f ·t abl a e nt rada[ i ] •-e lmi

i t e Sat u raci ón· t he n

* , • , , , • •• , , • t abl a sali d a[ i ] , : = • l ímit e Sat u raci ón;

······ ·· ·e l se - -

······ ·· ·· ···tabl a sa l id a[ i ] · : =· t a bl a ent rad a[ i ]

29

31')

· ··· · ··· i f · t abl a ent rada [i ] ·>· l mi

i t e sat u raci on · t he n

* • • • , • • • , • , • t abl a sa li da [i ] , : = • l imi t e• s at u raci on

· ··· · ·· ·e l se - -

···· · ··· ·· ··t abl a sa li da [i ] · : =· t abl a ent rada[ i ]

36 • • • • ·• •-•

37 end; 38

39

end ; - -

31 • • • • .. -•

32 end;

33

34

end ; - -

41')

**41**

proced ur em· uest ra\_t abl a{ va r· t abl a : · Ti pol ab l a )· ; va, r

1. proced u rem-
2. va r

ues t ra\_t abl a (v a r ·t abl a: ·Ti pol ab l a ) ;

42 i ·-:

43 beqi n

i nt ege r;

1. i · :- i nt e ger;
2. beqi n

**44**■

45

46

47

48

**1**

li l

* • • • f or• i ·=· l " t o • t aman i o a r ray- do •

······ ·· ·wri t el n ( t abl a[ i ll ; end ;

cons t

····Val o rMaxmi

**39'111**

41')

**41**

42

43

**:1**

46

* •-• fo r· i · : = • l •t -o Tma anio.A r ra-y do •

· ··· · ·· ·wri t el n ( t abl a[ i ] ) ; end;

var

· ··· t abla \_ brut o, · t abl a\_fi l t rada · : ·Ti poTabl a ; const

· ···Valo rMa,xmi

o· =· 1 6; ·/ / ·Ge nera rme os · *va*l o res· ent re· **1** ·y ·16, · l uego · res t a rme os · l

···· Lmi

o· =· 1 5;

i t e5at u raci ón · = · l8;

**:1** / / ·para ·que· vaya ·de ·l'l· a • 15

52 1 begi n

**1**

* · • • Lmi i t eSat u racLo n • =· l B ;

51') beqin

**il**

53 , • , •

rell ena tabl a ( t abl a brut o, •V a l orMaxmi

* J ;

· ···d el ay (l llfl0 ) ;

* + • • • randmo i ze; • •

1. • • • • rell ena , t abl a (t abl a brut o , • Val orMaxmi

* ) ;

**54**■

···· t abl a f i l t rada · : =· f i l t ra t abl a (t abl a brut o, · · Lmi

i t eSat u raci ó n ) ;

55■

· ··· f il t ra t a bl a ( t abl a brut o , · t abl a f il t ra da, ·· ·Um it eSat uraci on l ;

1. ···w· ri t el n ( ' Tabl a·en • b,rut o: ' ) ; -
2. • •-•m ues t ra t abl a ( t abt a brut ol ;
3. wri t el n ( ' Tabl a·f i l t rada: ' ) ;
4. · ·· ·wri t el n, ( ' Tabl a·en· b rut o: ' ) ; -
5. • • •-m ue st ra, t abl a (t abt a brut o) ;
6. writ el n ( ' Tabl a·f il t rnda : ' ) ;
7. , • •m• ues t ra tabl a (t abl a f il t rada ) ;
8. • • •m• ue st ra, t abl a (t abl a f il t rada ) ;

# 591

end; - -

end. - -

### Fundamentos de la programacio´n y la inform´atica Examen de ejercicios de laboratorios

**1 de Febrero de 2021**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

## Ejercicio 1 (5 puntos)

En tu cuenta del laboratorio encontrar´as los siguiente ficheros [1](#_bookmark1).

~/fpi2021feb/todos\_mayores.pas

~/fpi2021feb/limita\_matriz.TULOGIN.pas.

El programa todos\_mayores.pas deber´ıas conocerlo. Est´a publicado en la pg. 84 del tema 8. Sirve para indicar si todos los valores de una matriz son mayores que una constante K. **No toques el fichero todos mayores.pas**. Est´a aqu´ı por si necesitas copiarlo de nuevo o consultarlo.

El fichero limita\_matriz.TULOGIN.pas de momento es una copia id´entica del programa fichero ante- rior. **Modifica el fichero limita matriz.TULOGIN.pas** para que cumpla la siguiente especificaci´on:

* 1. El programa generar´a una matriz de nu´meros aleatorios y la mostrar´a por pantalla, exactamente igual que en la versi´on anterior.
  2. El programa generar´a una nueva matriz, donde los valores mayores que k ser´an reemplazados por k.
  3. El resultado ser´a similar a este:

Matriz original:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3.89 | 1.38 | 3.63 | 1.12 | 9.52 |
| 9.17 | 0.84 | 0.30 | 3.05 | 8.59 |
| 1.53 | 7.34 | 0.28 | 4.75 | 8.00 |

K: 7

Matriz limitada:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3.89 | 1.38 | 3.63 | 1.12 | 7.00 |
| 7.00 | 0.84 | 0.30 | 3.05 | 7.00 |
| 1.53 | 7.00 | 0.28 | 4.75 | 7.00 |

* 1. Observa que el programa tiene que generar una nueva matriz. No es suficiente con que escriba en pantalla una salida como la mostrada anteriormente, sino que dentro del programa tendr´a que haber una matriz que contenga el resultado deseado.

En otras palabras:

#### No debes hacer esto

si posicion > k entonces escribe k

y si no

escribe posici´on

Debes hacer algo similar a esto:

escribe\_matriz(matriz); [bla bla bl´a] escribe\_matriz(matriz);

O bien

escribe\_matriz(matriz); [bla bla bl´a]

escribe\_matriz(matriz\_modificada);

1TULOGIN sera´

tu nombre de usuario en el laboratorio

### Solucio´n

*{£mode objfpc}{£H-}{£R+}{£T+}{£Q+}{£V+}{£D+}{£X-}{£warnings on}*

program limita\_matriz;

*// Reemplaza por K aquellos valores que sean mayores que K*

uses crt; *// Necesario para delay*

const

Filas = 3;

Columnas = 5; type

TipoMatriz = array[1..Filas, 1..Columnas] of real;

function genera\_real(cota\_superior:real):real; begin

result := random() \* cota\_superior; end;

procedure inicia\_matriz(var matriz:TipoMatriz; cota\_superior: real); var

i,j : integer; begin

for i := 1 to Filas do

for j:= 1 to Columnas do

matriz[i,j] := genera\_real(cota\_superior);

end;

procedure escribe\_matriz(matriz:TipoMatriz); var i,j: integer;

begin

for i := 1 to Filas do begin for j:= 1 to Columnas do

write(matriz[i,j]:6:2); writeln;

end; end;

procedure cambia\_mayores(var matriz: TipoMatriz;k: real); var

i,j: integer; begin

for i:= 1 to Filas do

for j:= 1 to Columnas do

if matriz[i,j] > k then matriz[i,j] := k;

end;

var

matriz: TipoMatriz;

const

CotaSuperior = 10 ; K = 7;

begin

randomize(); delay(800);

inicia\_matriz(matriz, CotaSuperior); writeln('Matriz original:'); escribe\_matriz(matriz);

writeln('K: ',k); writeln('Matriz limitada:'); cambia\_mayores(matriz,k); escribe\_matriz(matriz);

end.

## Ejercicio 2 (5 puntos)

En tu cuenta del laboratorio encontrar´as el fichero

~/fpi2021feb/dni.TULOGIN.pas

Que contiene el siguiente trozo de c´odigo

*{£mode objfpc}{£H-}{£R+}{£T+}{£Q+}{£V+}{£D+}{£X-}{£warnings on}*

program dni; uses crt; type

TipoPersona = Record dni : integer; genero : char;

end;

function tira\_dado(caras\_dado:integer):integer; begin

result := random(caras\_dado) + 1; end;

Compl´etalo para que genere de forma aleatoria unos cuantos (los que quieras) registros de tipo *TipoPersona* y los muestre en pantalla. No es necesario que los almacene en un vector, basta con que los genere y los muestre.

El campo *dni* ser´a un entero aleatorio entre 1 (incluido) y 99999999 (100 millones menos una unidad)

El campo *genero* ser´a o bien el car´acter M o bien el car´acter F (mayu´sculas). Elegir´a uno u otro tirando un dado de dos caras.

Ejemplo de ejecuci´on:

|  |  |
| --- | --- |
| 85763386 | M |
| 16664532 | M |
| 33731778 | F |
| 24172391 | F |
| 83404647 | M |
| 57830100 | F |

Observa que no basta con que tu programa genere una salida similar a la del ejemplo: es necesario que realmente utilice registros del tipo indicado.

### Solucio´n

*{£mode objfpc}{£H-}{£R+}{£T+}{£Q+}{£V+}{£D+}{£X-}{£warnings on}*

program dni; uses crt; type

TipoPersona = Record dni : integer; genero : char;

end;

function tira\_dado(caras\_dado:integer):integer; begin

result := random(caras\_dado) + 1; end;

procedure genera\_persona(var persona: TipoPersona); const

CarasDado = 99999999; begin

persona.dni := tira\_dado(CarasDado);

if tira\_dado(2) = 1 then persona.genero := 'M'

else

persona.genero := 'F';

end;

procedure imprime\_persona(persona: TipoPersona); begin

write(persona.dni:8); writeln(persona.genero:8);

end;

var

i : integer;

persona : TipoPersona;

const

NumeroEjemplos = 6; begin

delay(800); randomize();

for i := 1 to NumeroEjemplos do begin genera\_persona(persona); imprime\_persona(persona);

end; end.

### Fundamentos de la programacio´n y la inform´atica Examen de entrega de pr´acticas

**1 de Febrero de 2021**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

### Ejercicio u´nico

Entra en tu cuenta del laboratorio y haz una copia de tu fichero ~/fpi/practica08/unicos.pas

que se llame ~/fpi/practica08/repes.pas

Esto puedes hacerlo o bien con el editor nano o bien con los siguientes comandos:

cd

cd fpi

cd practica08

cp unicos.pas repes.pas

**No toques el fichero unicos.pas. Modifica el fichero repes.pas** para que cumpla la siguiente especificaci´on:

1. Tendr´a una constante global llamada MaxRepes. Ponle por ejemplo el valor 3.
2. El programa generar´a una matriz donde los valores podr´an estar repetidos, pero como mucho cada valor aparecer´a un total de MaxRepes veces.
3. Comprobar´a que se cumple la precondici´on de que *CarasDado ∗ MaxRepes <*= *F ilas ∗ Columnas*

Observaciones

En tu programa original deber´ıas tener una funci´on que te indicaba si un nu´mero estaba o no en la matriz. Reempl´azala por una que indique cu´antas veces aparece el nu´mero.

Supongamos que MaxRepes vale 3. Para rellenar una posici´on, tendr´as que generar nu´meros hasta obtener uno que, como mucho, haya aparecido hasta el momento 2 veces. Una vez que an˜adas a la matriz el nu´mero generado, ser´an 3.

Ejemplo de ejecuci´on:

Caras dado: 10 MaxRepes: 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 5 | 8 | 7 |
| 2 | 1 | 1 | 7 | 6 |
| 1 | 2 | 6 | 7 | 9 |
| 5 | 10 | 6 | 3 | 4 |

Ejemplo de otra ejecuci´on:

Caras dado: 10 MaxRepes: 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | 6 | 8 | 7 | 9 |
| 8 | 6 | 6 | 10 | 1 |
| 9 | 5 | 1 | 7 | 1 |
| 10 | 3 | 10 | 7 | 9 |

### Fundamentos de la programacio´n y la inform´atica Test final

**1 de Febrero de 2021**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

Ejecuta en un terminal la orden

~mortuno/prepara

Comprueba que esto ha dejado en tu cuenta el fichero

~/fpi2021feb/test.TULOGIN.txt, donde deber´as resolver el ejercicio [1](#_bookmark2).

### Ejercicio u´nico (10 puntos)

Cualquiera que tenga nociones de ´algebra sabe que (*a*+*b−b*)*∗b* = *a*. Sin embargo, el c´odigo a continua- ci´on parece demostrar lo contrario. Obviamente el programa tiene que tener algu´n problema. Indica:

*b*

1. Por qu´e parece demostrar lo contrario.
2. Qu´e error o errores hay en el c´odigo.
3. Por qu´e se producen.
4. C´omo se deber´ıan solucionar.

1 *{£mode objfpc}{£H-}{£R+}{£T+}{£Q+}{£V+}{£D+}{£X-}{£warnings on}*

2 program errores\_p2;

3

4 var

5 x,pizca : real;

6

7 function f(x, pizca:real): boolean;

8 var

9 y : real;

10 begin

11 y := x;

12 y := y \* pizca;

13

14 y := y + pizca;

15 y := y - pizca;

16

17 y := y / pizca;

18

19 result := (x = y);

20 end;

21

22 begin

23 x := 2.345678901;

24 pizca := 0.0000001;

25 writeln(f(x,pizca));

26

27 pizca := 0.001;

28 writeln(f(x,pizca));

29 end.

Resultado de la ejecuci´on:

FALSE TRUE

1TULOGIN sera´

tu nombre de usuario en el laboratorio

### Respuesta

1. Parece demostrar lo contrario porque hace una serie de operaciones con *x* que resultan equivalentes a la f´ormula indicada en el enunciado, compara el valor tras estas operaciones con el valor original, y aunque deber´ıan ser siempre iguales, en un caso no lo es. Concretamente cuando *pizca* vale una diezmillon´esima.

Esto se debe a los errores de conversi´on propios de la representaci´on de nu´meros reales en *coma flotante* (ver tema 2, pg 43-46)

Para solucionarlo, nunca deber´ıamos comparar que dos nu´meros reales sean iguales. Deber´ıamos buscar que la diferencia entre los dos nu´meros sea menor que cierto valor, despreciable en nuestro

´ambito (ver tema 8, pg 25-26)[2](#_bookmark3).

1. Las variables *x* y *pizca* son globales, porque est´an declaradas antes que los subprogramas. De- ber´ıamos declaralas inmediatamente antes del cuerpo del programa principal.

2Para reducir estos errores podr´ıamos usar tipos de datos reales con mayor precisi´on, pero nunca los podremos eliminar por completo, siempre tendremos que buscar una diferencia lo bastante pequen˜a

### Fundamentos de la programacio´n y la inform´atica Examen de ejercicios de laboratorios

**8 de Julio de 2021**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

## Ejercicio 1 (5 puntos)

En tu cuenta del laboratorio encontrar´as el siguiente fichero [1](#_bookmark4).

~/fpi.julio.21/cadenas.TULOGIN.pas

Escribe en ´el un programa que tenga una funci´on que reciba una cadena, y que devuelva otra cadena, construida a partir de la original, reemplazando

El caracter @ por la cadena [arroba]

El caracter . por la cadena [punto]

Ejemplo: Si recibe la cadena [juan.perez@urjc.es](mailto:juan.perez@urjc.es) Devolver´a la cadena juan[punto]perez[arroba]urjc[punto]es

## Ejercicio 2 (5 puntos)

En tu cuenta del laboratorio encontrar´as un programa que genera una matriz aleatoria y la escribe en pantalla, est´a en el fichero

~/fpi.julio.21/media.TULOGIN.pas

An˜ade los subprogramas necesarios para generar un programa que genere otra matriz de las mismas dimensiones, pero con todos sus elementos iguales. Este valor ser´a la media de los elementos de la matriz original. Ejemplo:

Matriz original:

82.25 47.65 39.50 22.03

1.56 12.77 15.43 7.81

16.65 58.13 4.11 35.94

Matriz constante:

28.65 28.65 28.65 28.65

28.65 28.65 28.65 28.65

28.65 28.65 28.65 28.65

Observa que el programa deber´a *generar* otra matriz, esto es, otro elemento de tipo TipoMatriz. No basta con mostrar un texto en pantalla.

1TULOGIN sera´

tu nombre de usuario en el laboratorio

### Fundamentos de la programacio´n y la inform´atica Examen de entrega de pr´acticas

**8 de Julio de 2021**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

### Ejercicio 1 (5 puntos)

Ahora modificar´as tu pr´actica 7.3. Haz una copia de tu fichero

~/fpi/practica07/hasta01.pas

que se llame

~/fpi/practica07/examen01.pas

Esto puedes hacerlo o bien con el editor nano o bien con los siguientes comandos:

cd

cd fpi

cd practica07

cp hasta01.pas examen01.pas

**No toques el fichero hasta01.pas. Modifica el fichero examen01.pas** para que cumpla la siguiente especificaci´on:

1. El c´odigo que pide al usuario un entero positivo, debe ser un procedimiento (Tal vez esto ya lo cumples. O tal vez no, esto es, puede que tengas el c´odigo en el cuerpo del programa principal, entonces deber´as modificarlo). Este procedimiento puede llamar a otros subprogramas. O no, ambos enfoques son v´alidos.
2. Este procedimiento debe devolver, en un par´ametro por referencia, el valor entero positivo escrito por el usuario.
3. Este procedimiento debe devolver, en otro par´ametro por referencia, el nu´mero de veces que el usuario se ha equivocado.

En el cuerpo del programa principal, llama al procedimiento y escribe en pantalla los dos par´ametros devueltos por el procedimiento.

### Ejercicio 2 (5 puntos)

Este ejercicio es una modificaci´on de tu pr´actica 7.8. Haz una copia de tu fichero

~/fpi/practica07/maximo\_minimo.pas

que se llame

~/fpi/practica07/examen02.pas

Esto puedes hacerlo o bien con el editor nano o bien con los siguientes comandos:

cd

cd fpi

cd practica07

cp maximo\_minimo.pas examen02.pas

**No toques el fichero maximo minimo.pas. Modifica el fichero examen02.pas** para que el usua- rio no escriba reales sino enteros. Adem´as, en vez de mostrar el m´aximo y el m´ınimo de los valores introducidos, el programa deber´a escribir:

El nu´mero de valores pares y el nu´mero de valores impares.

El m´aximo de los valores pares y el m´aximo de los valores impares. El m´ınimo de los valores pares y el m´ınimo de los valores impares.

### Fundamentos de la programacio´n y la inform´atica Test parcial

**8 de Julio de 2021**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

Ejecuta en un terminal la orden

~mortuno/prepara

Comprueba que esto ha dejado en tu cuenta el fichero

~/fpi.julio.21/parcial.TULOGIN.txt, donde deber´as resolver el ejercicio 1.

## Ejercicio u´nico (10 puntos)

Sean las expresiones

e1 := ( not p and not (p and q) ) and not r e2 := not (not (a < 0) and not (b < 10 ))

1. A partir de e1 escribe una expresi´on l´ogica equivalente, m´as legible, de nombre s1
2. A partir de e2 escribe una expresi´on l´ogica equivalente, m´as legible, de nombre s2
3. Escribe una expresi´on n1 que sea la negaci´on de s1
4. Escribe una expresi´on n2 que sea la negaci´on de s2

### Fundamentos de la programacio´n y la inform´atica Test final

**8 de Julio de 2021**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

Ejecuta en un terminal la orden

~mortuno/prepara

Comprueba que esto ha dejado en tu cuenta el fichero

~/fpi.julio.21/test.TULOGIN.txt, donde deber´as contestar las siguientes preguntas 1.

## Ejercicio 1 (4 puntos)

Explica brevemente la siguiente afirmaci´on: *una funci´on no deber´ıa tener efectos laterales*.

## Ejercicio 2 (3 puntos)

¿Para qu´e sirven los registros?

## Ejercicio 3 (3 puntos)

¿Por qu´e decimos que los punteros son *peligrosos*?

## Fundamentos de la programacio´n y la inform´atica Examen Parcial. 23 de Noviembre de 2021

Grados en ingenier´ıa aeroespacial. Turno de tarde

Universidad Rey Juan Carlos

Ejecuta en un terminal la orden

~mortuno/prepara

Comprueba que esto ha dejado en tu cuenta los ficheros

~/fpi.nov.21/logica.TULOGIN.txt, donde deber´as resolver el ejercicio 1 [1](#_bookmark5)..

~/fpi.nov.21/precio.TULOGIN.txt, donde deber´as resolver el ejercicio 2.

~/fpi.nov.21/ejemplo, un ejecutable que puedes usar como referencia para ver c´omo deber´ıa comportarse tu ejercicio 2.

# Ejercicio 1 (2 puntos)

Sean las expresiones

e1 := not (not a or not b) and not (b or c) and (not d and not e) ; e2 := not (x <= 2) or not (y > 12);

1. A partir de e1, aplica De Morgan y escribe una expresi´on l´ogica equivalente, intentando que sea m´as clara, de nombre s1
2. A partir de e2, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre s2
3. Escribe una expresi´on n1 que sea la negaci´on de s1
4. Escribe una expresi´on n2 que sea la negaci´on de s2

## Solucio´n

s1 := (a and b) and (not b and not c) and (not d and not e) s2 := (x > 2) or ( y <=12 )

n1 := (not a or not b) or (b or c) or (d or e) n2 := (x <= 2) and (y > 12)

# Ejercicio 2 (8 puntos)

Supongamos un comerciante que calcula el precio de un producto de la siguiente forma: a partir del precio de coste, le an˜ade su margen comercial (expresado como porcentaje) y le suma el IVA. Este impuesto tiene 3 tipos: general (21 %), reducido (10 %) y superreducido (4 %).

Escribe un programa en Pascal le pregunte todo esto al usuario y calcule el precio final. El usuario indicar´a el tipo del IVA escribiendo *g*, *r* o *s*, en m´ınu´scula. Los detalles puedes verlos en esta ejemplo de ejecuci´on:

1TULOGIN ser´a tu nombre de usuario en el laboratorio

¿Precio de coste?

100

¿Margen comercial? (en porcentaje) 15

¿Tipo de iva? g: general r: reducido s: superreducido r

Precio final: 126.50

Si el usuario hace algo distinto de lo pedido (no escribe un nu´mero en las dos primeras preguntas o no escribe ni *g* ni *r* ni *s* en la tercera pregunta), el programa mostrar´a un error. Aqu´ı puedes ver un ejemplo:

¿Precio de coste?

100

¿Margen comercial? (en porcentaje) 15

¿Tipo de iva? g: general r: reducido s: superreducido x

Has escrito algo mal

## Sugerencias

Reutiliza el procedimiento *intenta leer real* que escribiste para tu pr´actica 5.5.

Escribe un procedimiento *intenta leer tipo iva*, similar a *intenta leer real*, que le haga la pregunta al usuario, que devuelva el car´acter que ha escrito el usuario y tambi´en un booleano *TRUE* si la respuesta es correcta (*g*, *r* o *s*, en minu´scula) o FALSE, en otro caso.

Escribe una funci´on que a partir del car´acter *g*, *r* o *s*, devuelva un real con el tipo del impuesto (21, 10 o 4).

Calcula as´ı el precio final:

function calcula\_precio(coste, margen, tipo\_iva: real):real; var

precio\_sin\_iva, precio\_final : real; begin

precio\_sin\_iva := coste \* (1 + 0.01 \* margen); precio\_final := precio\_sin\_iva \* (1 + 0.01 \* tipo\_iva); result := precio\_final;

end;

## Solucio´n

{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}

program precio;

function dime\_iva(char\_iva: char):real; begin

if char\_iva = 'g' then *// general*

result := 21.0

else if char\_iva = 'r' then *//reducido*

result := 10.0

else if char\_iva = 's' then *///superreducido*

result := 4.0 else begin

writeln('Error, tipo no reconocido:', char\_iva); halt;

end;

end;

procedure intenta\_leer\_real(pregunta:string; var ok: boolean; var valor:real); var

codigo: integer; cadena: string;

begin

writeln(pregunta); readln(cadena);

val(cadena, valor, codigo); if codigo = 0 then

ok := True else

ok := False;

end;

procedure intenta\_leer\_tipo\_iva(pregunta:string; var ok: boolean; var char\_iva:char); begin

writeln(pregunta); readln(char\_iva);

ok := (char\_iva = 'g') or (char\_iva = 'r') or (char\_iva = 's')

end;

function calcula\_precio(coste, margen, tipo\_iva: real):real; var

precio\_sin\_iva, precio\_final : real; begin

precio\_sin\_iva := coste \* (1 + 0.01 \* margen); precio\_final := precio\_sin\_iva \* (1 + 0.01 \* tipo\_iva); result := precio\_final;

end;

var

char\_iva : char; ok1,ok2, ok3: boolean; coste, margen : real; pregunta : string;

begin

intenta\_leer\_real('¿Precio de coste?', ok1, coste); intenta\_leer\_real('¿Margen comercial? (en porcentaje)', ok2, margen); pregunta := '¿Tipo de iva? g: general r: reducido s: superreducido';

intenta\_leer\_tipo\_iva(pregunta, ok3, char\_iva); if (ok1 and ok2 and ok3) then begin

write('Precio final: ');

writeln(calcula\_precio(coste, margen, dime\_iva(char\_iva)):0:2);

end.

end else

writeln('Has escrito algo mal');

### Fundamentos de la programacio´n y la inform´atica Examen de entrega de pr´acticas

**19 de Enero de 2022**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

### Ejercicio 1 (5 puntos)

En este ejercicio modificar´as tu pr´actica 7.3. Haz una copia de tu fichero

~/fpi/practica07/filtrado03.pas

que se llame

~/fpi/practica07/filtrado04.pas

Esto puedes hacerlo o bien con el editor nano o bien con los siguientes comandos:

cd

cd fpi

cd practica07

cp filtrado03.pas filtrado04.pas

**No toques el fichero filtrado03.pas. Modifica el fichero filtrado04.pas** para que cumpla la siguiente especificaci´on:

1. El procedimiento *lee real* har´a lo mismo que hasta ahora y devolver´a las mismas cosas. Pero, adem´as, devolver´a tambi´en un entero indicando cu´antas veces se ha equivocado el usuario, esto es, cu´antas veces escribi´o algo distinto de un nu´mero real. Si lo escribi´o bien a la primera, este valor ser´a cero.
2. Tienes libertad para disen˜ar los retoques de este procedimiento.
3. En alguna parte del programa, debes escribir en pantalla el nu´mero de veces que el usuario se ha equivocado. Tienes libertad para decidir c´omo y donde.
4. Evidentemente, si estas *libertades* las implementas de forma contradictoria con los principios de buen disen˜o que hemos visto durante el curso, la nota tendr´a la penalizaci´on correspondiente.

Si tu programa no compila, su nota ser´a nula.

### Ejercicio 2 (5 puntos)

Ahora modificar´as tu pr´actica 8.2. Haz una copia de tu fichero

~/fpi/practica08/busca\_matriz.pas

que se llame

~/fpi/practica08/busca\_matriz02.pas

Esto puedes hacerlo o bien con el editor nano o bien con los siguientes comandos:

cd

cd fpi

cd practica08

cp busca\_matriz.pas busca\_matriz02.pas

#### No toques el fichero busca matriz.pas. Modifica el fichero busca matriz02.pas para que

Adem´as de mostrar la misma informaci´on que la versi´on anterior, indique al final de su ejecuci´on (y solo al final de su ejecuci´on) cu´al ha sido la media m´as baja y la media m´as alta de todas las obtenidas.

Si tu programa no compila, su nota ser´a nula.

### Fundamentos de la programacio´n y la inform´atica Examen de ejercicios de laboratorios

**19 de Enero de 2022**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

Ejecuta en un terminal la orden

~mortuno/prepara

Comprueba que esto ha dejado en tu cuenta los ficheros

~/fpi.enero.22/cadenas.TULOGIN.pas

~/fpi.enero.22/diagonales.TULOGIN.pas

[1](#_bookmark6).

## Ejercicio 1 (4 puntos)

Edita el fichero ~/fpi.enero.22/cadenas.TULOGIN.pas para escribir un programa en Pascal que tenga una funci´on que reciba una cadena, y que devuelva otra cadena, construida a partir de la original, pero reemplazando

El caracter # por la cadena [hashtag]

El caracter @ por la cadena [at]

Ejemplo: Si recibe la cadena #examen-fpi@urjc

Devolver´a la cadena

[hashtag]examen-fpi[at]urjc

Si el programa no compila, su nota ser´a nula.

## Ejercicio 2 (6 puntos)

En el fichero ~/fpi.enero.22/diagonales.TULOGIN.pas encontrar´as un programa que genera una matriz aleatoria y la escribe en pantalla. An˜ade los subprogramas necesarios para generar un programa que

Compruebe la precondici´on de que la matriz es cuadrada (tiene el mismo nu´mero de filas que de columnas)

Si esta precondici´on se cumple, mostrar´a en pantalla

* El sumatorio de la diagonal principal
* El sumatorio de la diagonal secundaria Ejemplo:

4 2 3

2 3 1

1 4 2

Sumatorio diagonal principal:9 Sumatorio diagonal secundaria:7

Si la precondici´on no se cumple, mostrar´a un mensaje de error.

Observaciones

Se denomina *diagonal principal* de una matriz cuadrada al conjunto de elementos cuyo primer

´ındice es igual al segundo ´ındice. Por ejemplo para matrices cuadradas de tres filas son los elementos

*a*1*,*1*a*2*,*2*a*3*,*3

1TULOGIN sera´

tu nombre de usuario en el laboratorio

Se denomina *diagonal secundaria* de una matriz cuadrada al conjunto de elementos cuyos ´ındices i,j cumplen la condici´on de sumar n+1, siendo n el nu´mero de filas.

Por ejemplo para matrices cuadradas de tres filas son los elementos *a*1*,*3*a*2*,*2*a*3*,*1

En otras palabras: los elementos i,j donde j vale *n − i* + 1

En otras palabras: debes recorrer todos las filas (´ındice i) y calcular la columna (´ındice j) a partir de la expresi´on *n − i* + 1

Obviamente, este programa tiene que seguir funcionando si modificamos el valor de las constantes *Filas* y *Columnas*, no es v´alido que escribas un programa que solo funciona si el nu´mero de filas es exactamente 3.

Si el programa no compila, su nota ser´a nula.

### Solucio´n

*{£mode objfpc}{£H-}{£R+}{£T+}{£Q+}{£V+}{£D+}{£X-}{£warnings on}*

program diagonales;

uses crt; *// Necesario para delay*

const

Filas = 3;

Columnas = 3; type

TipoMatriz = array[1..Filas, 1..Columnas] of integer;

function tira\_dado(caras\_dado:integer):integer; begin

result := random(caras\_dado) + 1; end;

procedure inicia\_matriz(var matriz:TipoMatriz); var

i,j : integer; const

CarasDado = 6; begin

for i := 1 to Filas do

for j:= 1 to Columnas do

matriz[i,j] := tira\_dado(CarasDado);

end;

procedure escribe\_matriz(matriz:TipoMatriz); var i,j: integer;

begin

for i := 1 to Filas do begin for j:= 1 to Columnas do

write(matriz[i,j], ' '); writeln;

end; end;

function suma\_diag\_prin(var matriz:TipoMatriz):integer; var i, sumatorio: integer;

begin

sumatorio := 0;

for i:= 1 to Filas do begin

sumatorio := sumatorio + matriz[i,i]; end;

result := sumatorio; end;

function suma\_diag\_sec(var matriz:TipoMatriz):integer; var i, sumatorio: integer;

begin

sumatorio := 0;

for i:= 1 to Filas do begin

sumatorio := sumatorio + matriz[i, filas - i +1 ]; end;

result := sumatorio; end;

var

matriz : TipoMatriz;

begin

randomize(); delay(800); inicia\_matriz(matriz); escribe\_matriz(matriz);

writeln('Sumatorio diagonal principal:', suma\_diag\_prin(matriz)); writeln('Sumatorio diagonal secundaria:', suma\_diag\_sec(matriz));

end.

### Fundamentos de la programacio´n y la inform´atica Test parcial

**19 de Enero de 2022**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

Ejecuta en un terminal la orden

~mortuno/prepara

Comprueba que esto ha dejado en tu cuenta el fichero

~/fpi.enero.22/parcial.TULOGIN.txt, donde deber´as resolver el ejercicio 2 [1](#_bookmark7).

## Ejercicio 1 (3 puntos)

Sean *a*, *b*, *c*, *d* variables booleanas. Sean *x* e *y* variables reales. Sean las expresiones

e1 := not ( (a and not b) and not (c or d) ); e2 := not (x > 3) or (y <= 5);

1. A partir de *e1*, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre *s2*.
3. Escribe una expresi´on *n1* que sea la negaci´on de *s1*.
4. Escribe una expresi´on *n2* que sea la negaci´on de *s2*.

### Solucio´n

s1:= not a or b or c or d s2:= (x<=3) or (y<= 5)

n1:= a and not b and not c and not d n2:= (x>3) and (y>5)

## Ejercicio 2 (7 puntos)

Indica qu´e l´ıneas del siguiente programa contienen errores. Describe brevemente cada error, aunque no es necesario que lo clasifiques.

{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}

1 program pts\_eur;

2

3 var

4 val\_ejemplo : real = 1000;

5 val\_convertido : real;

6

7 procedure pts\_eur(pts:real; var eur: real );

8 begin

9 euros := pts / 166.386;

10 end;

11

12 begin

13 valor\_convertido = pts\_eur(valor\_ejemplo)

14 write(valor\_ejemplo:0:2, ' pts son ')

15 writeln(valor\_convertido:0:2, ' euros')

16 end.

1TULOGIN sera´

tu nombre de usuario en el laboratorio

### Solucio´n

{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}

1 program pts\_eur;

2

3 *// Normalmente los n´umeros deben definirse como constantes, no*

4 *// usar 'n´umeros m´agicos'*

5 const PtsEur = 166.386;

6

7 *// Este subprograma se invoca como funci´on, as´ı que debe ser una*

8 *// funci´on, no un procedimiento (o si se usa como procedimiento,*

9 *// invocarlo correctamente)*

10 function pts\_eur(pts:real): real;

11 begin

12 result := pts / PtsEur;

13 end;

14

15 *// Si llamamos al par´ametro 'eur', no podemos escribir la expresi´on*

16 *// con el identificador 'euros'*

17 procedure pts\_eur(pts:real; var eur: real );

18 begin

19 eur := pts / PtsEur;

20 end;

21

22 *// Las variables deben ser locales*

23 var

24 valor\_ejemplo : real = 1000;

25 valor\_convertido : real;

26 *// Las variables deben usarse con el mismo nombre con el que se*

27 *// han definido*

28

29 begin

30 *// pts -> eur*

31 *// El operador de asignaci´on es :=, no =*

32 valor\_convertido := pts\_eur(valor\_ejemplo) ;

33

34 *// Las sentencias acaban en punto y coma*

35 pts\_eur(valor\_ejemplo, valor\_convertido);

36 write(valor\_ejemplo:0:2, ' pts son ');

37 writeln(valor\_convertido:0:2, ' euros');

38 end.

## Fundamentos de la programacio´n y la inform´atica Test final

**19 de Enero de 2022**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

Ejecuta en un terminal la orden

~mortuno/prepara

Comprueba que esto ha dejado en tu cuenta el fichero

~/fpi.enero.22/test.TULOGIN.txt, donde deber´as resolver el ejercicio [1](#_bookmark8).

Dados los p´arrafos siguientes, indica para cada uno de ellos si lo que dice es cierto o es falso. Justifica brevemente tu respuesta. Si hay m´as de un error, sen˜´alalos todos. Ambas cosas (justificar las respuestas y sen˜alar todos los errores) son imprescindibles: si de cada p´arrafo lo u´nico que dices es *cierto* o *falso*, tu respuesta no tendr´a casi ningu´n valor.

## P´arrafo 1

Supongamos que:

1. Queremos escribir un programa que almacene y gestione los datos de los clientes de una empresa.
2. El programa debe funcionar para cualquier nu´mero de clientes, ya sean 2, 2000 o 20000 (siempre que el ordenador sea lo bastante potente)

Si usamos un lenguajes de programaci´on *tradicional* como Pascal o C, esto no es demasiado complicado, porque siempre podemos usar *memoria est´atica*, esto es, punteros.

Lenguajes m´as modernos como Python o Java permiten usar *memoria din´amica*, esto es, arrays. Tambi´en sirven aunque a priori no conozcamos el nu´mero de clientes, pero son m´as com- plicados de programar.

## P´arrafo 2

En Pascal, si un programador olvida cerrar un fichero el resultado es similar al de una persona que olvida cerrar un grifo: el disco se llenar´a de *datos basura* y posiblemente se perder´a todo.

## Soluciones

1. Falso, por los siguiente motivos:

En lenguajes tradicionales, resolver el problema indicado tiene cierta dificultad, porque hay que usar punteros.

Usar punteros es usar memoria din´amica, no memoria est´atica.

Los lenguajes m´as modernos se ocupan autom´atica de la gesti´on de la memoria din´ami- ca. No es necesario que lo gestione el programador, por lo que resulta m´as sencillo.

1TULOGIN ser´a tu nombre de usuario en el laboratorio

1. Falso. El programador deber´ıa cerrar los ficheros, pero si se olvida, no suele resultar fatal porque al acabar la ejecuci´on del programa, el sistema operativo se ocupa de cerrarlos [2](#_bookmark9).

2El inconveniente puede ser, por ejemplo, que si un programador abre un fichero en modo escritura, olvida cerrarlo y el programa tarda en acabar, durante ese tiempo otros programas no podr´an acceder al fichero. Otro problema puede darse si hay muchas instancias del mismo programa abriendo un fichero en modo lectura, sin cerrarlo. Podr´ıan superar el nu´mero m´aximo de veces que se puede abrir un fichero. No es demasiado frecuente porque este nu´mero suele ser alto, pero puede suceder.

## Fundamentos de la programacio´n y la inform´atica Examen de ejercicios de laboratorio

**15 de Junio de 2022**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

Ejecuta en un terminal la orden

~mortuno/prepara

Comprueba que esto ha dejado en tu cuenta los ficheros

~/fpi.junio.22/monedas.TULOGIN.pas ~/fpi.junio.22/mezcla.TULOGIN.pas [1](#_bookmark10).

# Ejercicio 1 (5 puntos)

Escribe un programa en Pascal en el fichero ~/fpi.junio.22/monedas.TULOGIN.pas, que simule un juego de azar basado en las tiradas de unas monedas, con la siguiente especificaci´on:

1. Representaremos la cara con la letra *c* y la cruz con la letra *x*.
2. En cada tirada se lanzan *numero monedas*, por ejemplo 5. Este valor ser´a una constante.
3. A la cara le corresponde un valor de un punto, a la cruz, ninguno.
4. El resultado de cada tirada se mostrar´a en una l´ınea de la pantalla, esto es, las monedas que han salido en esa tirada y la puntuaci´on de esa tirada.
5. El juego seguir´a mientras no se alcance una *puntuaci´on objetivo*, por ejemplo 4 puntos. Este valor ser´a una constante. Cuando la puntuaci´on de una tirada sea mayor o igual que el objetivo, el juego se dentendr´a. Observa que hablamos siempre de la puntuaci´on en una *tirada*, (p.e. 5 monedas). No de la suma de todas las tiradas.

Ejemplo de ejecuci´on:

|  |  |
| --- | --- |
| cxcxx | 2 |
| xcxxc | 2 |
| ccxcx | 3 |
| xxxcc | 2 |
| xxcxx | 1 |
| ccccc | 5 |

1TULOGIN ser´a tu nombre de usuario en el laboratorio

## Solucio´n

{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}

program monedas; uses crt;

function dado(caras\_dado:integer):integer; begin

result := random( caras\_dado ) + 1 ;

end;

function tira\_moneda(): string; var

valor : integer; begin

valor := dado(2);

*// Permitimos este n´umero m´agico porque todas*

*// las monedas tienen 2 caras*

if valor = 1 then result := 'c'

else

result := 'x';

end;

var

sigue : boolean = True; moneda : string; puntuacion: integer = 0; i : integer;

const

Numero\_tiradas = 5;

Puntuacion\_objetivo = 4; begin

delay(600); randomize;

while sigue do begin puntuacion := 0;

for i := 1 to Numero\_tiradas do begin moneda := tira\_moneda(); write(moneda);

if moneda = 'c' then

puntuacion := puntuacion + 1;

if puntuacion >= Puntuacion\_objetivo then sigue := False;

end.

end

end;

writeln(' ',puntuacion);

# Ejercicio 2 (5 puntos)

Escribe un programa en Pascal en el fichero ~/fpi.junio.22/mezcla.TULOGIN.pas, que con- tenga y use una funci´on que:

1. Reciba dos cadenas de entrada.
2. Si la longitud de ambas cadenas de entrada no es igual, devolver´a la cadena *ERROR*.
3. Si la longitud es igual, devolver´a la cadena resultante de *mezclar* ambas cadenas de la siguiente forma:

Primer car´acter de la primera cadena. Primer car´acter de la segunda cadena. Segundo car´acter de la primera cadena. Segundo car´acter de la segunda cadena. etc.

Ejemplo:

Cadenas de entrada

*abcdef 012345*

Cadena de salida

*a0b1c2d3e4f5*

{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}

program mezcla;

function mezcla\_cadenas(s1, s2:string):string; var

salida : string; i: integer;

begin

if length(s1) <> length(s2) then result := 'ERROR'

else begin

salida := '';

for i := 1 to length(s1) do

salida := salida + s1[i] + s2[i]; result := salida;

end;

end;

var

s1,s2: string;

begin

s1 := 'abcdef';

s2 := '012345';

writeln(mezcla\_cadenas(s1,s2));

end.

## Fundamentos de la programacio´n y la inform´atica Examen de entrega de ejercicios

**15 de Junio de 2022**

Grados en ingenier´ıa aeroespacial. Turno de tarde Universidad Rey Juan Carlos

Ejecuta en un terminal la orden

~mortuno/prepara

Comprueba que esto ha dejado en tu cuenta los ficheros

~/fpi.junio.22/crecientes.TULOGIN.pas ~/fpi.junio.22/filtrado04.TULOGIN.pas [1](#_bookmark11).

En caso de que alguno de los programas que entregues hoy no compile, la nota de ese ejercicio ser´a pr´acti- camente nula.

# Ejercicio 1 (6 puntos)

En tu pr´actica 8.3 escribiste un programa llamado *unicos.pas* que ahora encontrar´as en el fichero ~/fpi.junio.22/crecientes.TULOGIN.pas. Modif´ıcalo para que cumpla la siguiente es- pecificacion:

1. El dado tenga un nu´mero de caras bastante alto, p.e. 100.
2. Los valores que se van an˜adiendo a la matriz no solo sean u´nicos, sino adem´as, estrictamente crecientes. Esto es:
   1. Cada nu´mero no solamente no podr´a repetirse, sino que deber´a ser mayor que el m´axi- mo que haya salido hasta el momento
   2. Cuando salga un nu´mero igual al m´aximo (p.e. 100) , ya ser´a imposible generar valores mayores. En ese caso, el programa mostrar´a un mensaje de error y dejar´a de generar valores.

Sugerencia:

Vete controlando el m´aximo valor aparecido en la matriz

Cuando tu programa empieza a buscar nu´meros hasta encontrar uno nuevo, cambia la condici´on, de forma que no solo sea un valor no repetido sino, adem´as, mayor que el m´aximo.

Ejemplo de ejecuci´on:

2 4 23

31 46 59

72 82 89

1TULOGIN ser´a tu nombre de usuario en el laboratorio

# Ejercicio 2 (4 puntos)

En tu pr´actica 7.3 escribiste un programa llamado *filtrado03.pas*. que ahora encontrar´as en el fichero ~/fpi.junio.22/filtrado04.TULOGIN.pas. Modif´ıcalo para que escriba toda la infor- maci´on relevante de su ejecuci´on no solo en pantalla, sino en el fichero *resultado.txt*

Entendemos como *relevantes* todos los mensajes que aparecen en pantalla, excepto aque- llos dirigidos al usuario para preguntarle informaci´on. P.e. *introduce el radio*, *introduce la superficie alar*. Estos mensajes **no** son relevante.

Cuando el usuario escriba un valor (correcto o incorrecto), esto ser´a relevante. As´ı que el programa escribir´a mensajes como

*El usuario escribe 23 El usuario escribe j41 La masa son 2320 Kg*

## Fundamentos de la programacio´n y la inform´atica

**Test de junio (correspondiente al test parcial y test final de enero) 15 de Junio de 2022**

Grados en ingenier´ıa aeroespacial. Turno de tarde

Universidad Rey Juan Carlos

Ejecuta en un terminal la orden

~mortuno/prepara

Comprueba que esto ha dejado en tu cuenta el fichero

~/fpi.junio.22/test.TULOGIN.txt, donde deber´as resolver el ejercicio [1](#_bookmark12).

# Ejercicio 1 (3 puntos)

Sean *a*, *b*, *c*, *d* variables booleanas. Sean *x* e *y* variables reales. Sean las expresiones

e1 := not (not p or q) and not (r and not s) ; e2 := not (x >=10 ) or (y > 4);

1. A partir de *e1*, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresi´on l´ogica equivalente, m´as legible, de nombre *s2*.
3. Escribe una expresi´on *n1* que sea la negaci´on de *s1*.
4. Escribe una expresi´on *n2* que sea la negaci´on de *s2*.

## Respuesta

s1:= (p and not q) and (not r or s);

s2:= (x<10) or (y>4);

n1:= (not p or q) or (r and not s);

n2:= (x>=10) and (y<=4);

# Ejercicio 2 (7 puntos)

Dados los p´arrafos siguientes, indica para cada uno de ellos si lo que dice es cierto o es falso. Justifica brevemente tu respuesta. Si hay m´as de un error, sen˜´alalos todos. Ambas cosas (justificar las respuestas y sen˜alar todos los errores) son imprescindibles: si de cada p´arrafo lo u´nico que dices es *cierto* o *falso*, tu respuesta no tendr´a casi ningu´n valor.

1TULOGIN ser´a tu nombre de usuario en el laboratorio

## P´arrafo 1

Es conveniente que el programador abra y cierre todos los ficheros que utilice. Pero si lo olvida, el sistema operativo se encargar´a. Esto resultar´a menos eficiente y puede dar problemas en algu´n caso particular, pero no ser´a un error fatal.

## P´arrafo 2

Los ordenadores representan internamente los nu´mero reales en binario. Un voltaje superior a cierto umbral representa un uno, un voltaje que no cumpla esto representa un cero. No hay valores intermedios, por tanto esto es muy exacto y permite que pr´acticamente cualquier programador en cualquier lenguaje haga c´alculos con todos los decimales que necesite. Ser´a m´as o menos c´omodo o m´as o menos *bonito* para las personas, pero los c´alculos siempre ser´an exactos.

## Respuesta

1. Falso. Abrir es imprescindible, de lo contrario el programa dar´a un error. Cerrar es conve- niente, si el programador olvida cerrar, el sistema operativo lo cerrar´a. Pero solamente el programador puede abrir.
2. Es cierto que todos los ordenadores representan internamente los nu´mero reales (y todos los dem´as datos) en binario. Es cierto que un voltaje superior a cierto umbral es un uno y un cero en otro caso [2](#_bookmark13). Pero es falso que todos los c´alculos ser´an siempre exactos con todos los decimales necesarios. Como vimos en el tema 2, apartado *representaci´on de los nu´meros reales*, la conversi´on de un nu´mero desde decimal hasta binario provoca errores a partir de cierto nu´mero de decimales. El programador puede elegir tipos de datos con mayor precisi´on, de entre los que le ofrezca el lenguaje. Pero los lenguajes de programaci´on convencionales siempre tienen errores a partir de cierto nu´mero de decimales.

2A veces el uno es el voltaje inferior al umbral, pero esto es irrelevante

© 2022 Miguel Angel Ortuño Pérez.

Algunos derechos reservados. Este documento se distribuye bajo la licencia *Atribución-CompartirIgual 4.0 Internacional* de Creative Commons disponible en

<https://creativecommons.org/licenses/by-sa/4.0/deed.es>