Fundamentos de la programación y la informática
Examen práctico. 21 de Diciembre de 2018
Grado en ingeniería aeroespacial en aeronavegación
Grado en ingeniería aeroespacial en vehículos aeroespaciales
Universidad Rey Juan Carlos

Preparativos

- Ejecuta el script `prepara_fpi`

  Esto creará 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 tamaño *lado*, construido con los caracteres almohadilla y espacio, según la siguiente especificación:

1. El programa pedirá al usuario un número entero entre 1 y 40, ambos inclusive. Si la entrada no es un número entero o está fuera de este rango, mostrará un error y repetirá la petición, todas las veces necesarias hasta que la entrega sea correcta.

   En el fichero tienes procedimiento que puedes aprovechar, aunque tendrás que modificarlo.

2. El programa tendrá un procedimiento llamado *escribe_cuadrado*, con un parámetro entero llamado *lado*.

3. Este procedimiento comprobará (de nuevo) que *lado* sea un número mayor o igual que 1, menor o igual que 40. Si el valor está fuera de este rango, mostrará un mensaje de error y finalizará la ejecución del programa.

4. Si *lado* vale 1, el programa escribirá

   ```
   #
   ```

5. Si *lado* vale 2, el programa escribirá

   ```
   # #
   # #
   ```

6. Si *lado* vale 3, el programa escribirá

   ```
   # # #
   #   #
   # # #
   ```

7. Para el resto de valores de *lado*, el programa se comportará de forma semejante.

Sugerencia: haz que tu programa considere los valores 1 y 2 como casos particulares, en ese caso escribirá 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.

Solución

```
{£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 número 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 están compuestas de lado-2 líneas
    // Cada línea 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(' . Debería 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 podría generalizarse también: una línea horizontal, cero líneas verticales y otra horizontal. Donde cada línea horizontal es un punto de tinta, cero espacios y otro punto. Pero la solución escrita aquí es más fácil de escribir y de entender.

Ejercicio 2 (3.5 puntos)

En la práctica 8.4, fichero `~/fpi/practica08/ordena.pas`, ordenaste un array sencillo donde cada elemento era un nombre y una puntuación. En este ejercicio, crearás un nuevo array solo con los *mejores* jugadores, que serán aquelos cuya puntuación sea mayor o igual a una puntuación mínima.

Copia tu programa `ordena.pas` en `~/fpi.diciembre.18/mejores.pas` y modifícalo para que siga la siguiente especificación:

1. Tendrá una función *solo_mejores* que
    - Recibirá como parámetros el array de registros y un entero *puntuacion_minima*.
    - Devolverá un array que contendrá aquellos registros de jugadores con puntuación mayor o igual a *puntuacion_minima*.
    - Si *puntuación_minima* está fuera del rango permitido (de 0 a 15), la función mostrará un error y el programa finalizará.

- El array devuelto tendrá la misma estructura que el original, esto es, será del mismo tipo y marcará la última posición con una entrada especial de nombre ZZZ.

2. El cuerpo principal del programa tendrá una variable local que usarás para dar valor a *puntuacion_minima*. No leas ningún valor desde el teclado.

3. El cuerpo principal del programa mostrará el array original sin ordenar, el array original ordenado y el array con los mejores jugadores (desordenado).

4. En tu programa, la función *solo_mejores* usará el array original sin ordenar. Pero deberá poder funcionar con cualquier array de este tipo, por ejemplo el array ordenado.

Solución

No publicamos una solución completa porque sería tanto como publicar una implementación de la práctica 8.4. Pero la función a escribir debería 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 función recibe el array *viejo* (*jugadores*) con todos los jugadores y la puntuación mínima 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 posición 1 hasta la posición de ZZZ (exclusive), por tanto usamos un bucle *for*

- Usamos *j* como índice para el array *nuevo*. No sabemos a priori cuántos elementos tendrá. Lo iniciamos a 1, y cada vez que un elemento (un jugador) tenga una puntuación mayor o igual a la requerida, lo copiamos del array nuevo al viejo e incrementamos *j*.

- Finalmente, añadimos ZZZ tras el último elemento del array nuevo, para marcar el fin.

Alternativas: como cualquier programa, se podría hacer de muchas formas distintas, similares, peores o mejores. Comentamos aquí alguna de las alternativas principales.

- En este ejemplo usamos una función *posicion_ultimo* solamente para saber la posición de último elemento, previo a ZZZ. Para un programador principiante posiblemente esto es lo más claro, permite usar un bucle sencillo. Un programador con más experiencia probablemente preferirá recorrer la variable *i* en un bucle *while*, que se vaya incrementando mientras no se alcance la posición ZZZ del array *viejo*.

- Si no fuera porque el enunciado pide una función, también podríamos haber usado un procedimiento que reciba los dos arrays, pasando el nuevo por referencia.

Ejercicio 3 (3.5 puntos)

En la práctica 9.2, fichero `~/fpi/practica09/aeropuerto_cercano.pas`, escribiste un programa que indicaba el aeropuerto más cercano a otro aeropuerto dado. Ahora lo modificarás para devolver todos los aeropuertos que estén a una distancia menor o igual que cierto valor.

Copia tu programa `~/fpi/practica09/aeropuerto_cercano.pas` en `~/fpi.diciembre.18/cercanos.pas` y modifícalo para que siga la siguiente especificación:

1. Al igual que en la versión anterior del programa, se le pedirá al usuario un código de aeropuerto, se le notificará que puede acabar pulsando *f*, se mostrará el código y el nombre del aeropuerto. En caso de error en el código, se mostrará un error.

2. A diferencia de la versión anterior, cuando el programa disponga de un código correcto, ya no se le mostrará al usuario el aeropuerto más cercano, sino que se le pedirá que introduza una distancia, en kilómetros. Si el valor introducido no es un número real o no es un número positivo, el programa lo notificará y volverá a pedir un valor numérico real, todas las veces que sea necesario hasta recibir un número válido. Consideramos *positivo* todo número estrictamente mayor que 0.

3. El programa escribirá en pantalla todos los aeropuertos (código y nombre) que estén a una distancia menor o igual que la indicada por el usuario.

4. En todo lo demás, este programa será igual que `aeropuerto_cercano.pas`. Esto es, leerá el mismo fichero, con la misma estructura, lo meterá en un array, trabajará sobre el array, etc. Cualquier cosa que no especifique este enunciado, se entiende que es igual que la práctica 9.2.

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

Solución

No publicamos una solución completa porque sería tanto como publicar una implementación de la práctica 9.2. Pero la función a escribir debería ser similar a esta:

```
procedure lee_distancia(var distancia: real);
var
    linea: string;
    num_real: real;
    codigo: integer;
begin
    repeat
        write('Introduce la distancia máxima, 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* podría usar una variable booleana (llamada por ejemplo *sal*) de forma que cuando valga *TRUE*, el bucle finalice. La versión escrita aquí es más compacta.
- El procedimiento *escribe_cercanos* podría recibir el código de aeropuerto inicial y sus coordenadas. Pero la versión escrita aquí es mejor, basta con pasar el índice de ese aeropuerto en el array para que el propio procedimiento obtenga estos datos.
- El procedimiento *escribe_cercanos* recibe como argumento el número de aeropuertos. También podría averiguarse desde el mismo procedimiento, bien llamando a una función que lo busque o bien incluyendo la búsqueda 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 búsqueda del último elemento, y por otro, la búsqueda de los elementos que cumplan la condición.
- La sentencia *write* y la sentencia *writeln* podrían mezclarse en una única sentencia *writeln*. Pero para el programador principiante es preferible separarlo, de esta forma los posibles errores se localizan más fácilmente.

Fundamentos de la programación y la informática
Examen de teoría. 21 de Diciembre de 2018
Grado en ingeniería aeroespacial en aeronavegación
Grado en ingeniería aeroespacial en vehículos aeroespaciales
Universidad Rey Juan Carlos

Preparativos

- Ejecuta el script `prepara_fpi`
- Esto creará 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 expresión lógica equivalente, más legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresión lógica equivalente, más legible, de nombre *s2*.
3. Escribe una expresión *n1* que sea la negación de *s1*.
4. Escribe una expresión *n2* que sea la negación de *s2*.

Solución

```
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 compilación, errores de ejecución, errores lógicos y defectos en la claridad del código. El siguiente programa está muy mal escrito. Señala 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 compilación, ejecución o lógico.

```
 1 {$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
 2 program errores-1;
 3
 4 var
 5     character: char;
 6     marks_table: array[1..100] of real;
 7     i: integer;
 8
 9 function valornumerico(carácter: char): integer;
10 begin
11     result := ord(character)-ord('0');
12 end;
13
14 procedure esmultiplode3(number: integer; result:boolean);
15 begin
16     result := ((number mod 3) = 0);
17 end;
18
19 function notapractnumerica(notapract: real; notaejerc: real): real;
20 begin
21     case notapract of
22     5.0:
23         notaejerc := result;
24     otherwise
25         result := 0.0;
26     end;
27 end;
28
29 function media( number1, number2: real): real;
30 begin
31     result := number1 + number2 / 2.0;
32 end;
33
34 begin
35     // Escribe True si la media de 5.0 y 7.0 es multiplo de 3
36     writeln(esmultiplode3(trunc(media(5.0, 7.0))));
37
38     // Rellena la tabla y escribe sus elementos en orden descendente
39     for i = 100 downto 0 do
40         marks_table[i] := notapractnumerica(5.0, 6.0 + real(i div 10));
41         write(marks_table[i], ' ');
42     writeln;
43 end.
```

Solución

1. Defecto. General: Mezcla de dos idiomas, español e inglés. Debería estar todo en inglés, o excepcionalmente, todo en español.

2. Error. Línea 2: El Identificador *Errores-1*, no es válido porque usa el carácter '-', que no está permitido.

3. Defecto. Línea 4: Variables globales. Estas variables deberían declararse después de los

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

4. Defecto. Línea 6: El *número mágico* 100 deberia ser declarado como constante (como el resto de números del programa).

5. Error. Línea 9: El identificador *carácter* no es válido porque usa el carácter 'á', que no está permitido.

6. Error. Línea 11: Uso de la variable global *character* dentro de un *procedure*.

7. Error. Línea 14: El segundo parámetro, *result*, debe pasarse por referencia (*var result:boolean*).

8. Error. Línea 21: *Case* está usando una variable real para seleccionar una rama u otra. Esto no está permitido, solo pueden usarse tipos discretos.

9. Error. Línea 23: La asignación debe ser al revés, *result := notaejerc ;*

10. Error. Línea 31: La expresión para calcular la media es incorrecta, debería ser: *(number1 + number2) / 2.0*.

11. Error. Línea 36: La llamada a *esmultiplode3()* debe hacerse con 2 argumentos, falta el segundo.

12. Error. Línea 36: El argumento de un subprograma, en este caso *writeln*, no puede ser un procedimiento, porque un procedimiento no devuelve ningún valor.

13. Error. Línea 39: El operador de asignación es `:=`, no `=`. Por tanto, la asignación debería ser *i := 100*, no *i = 100*.

14. Error. Línea 39: El bucle for debería ser *100 downto 1*, no *100 downto 0*.

15. Error. Líneas 39-42: Falta el *begin-end* del *for*. Tal y como está escrito el programa, la sentencia *write* está fuera del bucle, a pesar de que la tabulación indica lo contrario.

Fundamentos de la programación y la informática
Examen práctico. 11 de junio de 2019
Grado en ingeniería aeroespacial en aeronavegación
Grado en ingeniería aeroespacial en vehículos aeroespaciales
Universidad Rey Juan Carlos

Preparativos

- Ejecuta el script `~mortuno/prepara`

  Esto creará 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 tamaño variable, al que llamaremos *ventana*. El tamaño dependerá de un parámetro al que llamaremos *hueco*. La *ventana* de *hueco* 1 será la siguiente:

```
# # # # #
#   #   #
# # # # #
#   #   #
# # # # #
```

La *ventana* de *hueco* 2 será:

```
# # # # # # #
#     #     #
#     #     #
# # # # # # #
#     #     #
#     #     #
# # # # # # #
```

La de *hueco* 3 será

```
# # # # # # # # #
#       #       #
#       #       #
#       #       #
# # # # # # # # #
#       #       #
#       #       #
#       #       #
# # # # # # # # #
```

Y así sucesivamente, hasta un hueco máximo de 20. El programa pedirá al usuario que escriba por teclado el valor para *hueco*, un número entero entre 1 y 20. Si el usuario se equivoca, la pregunta se repetirá todas las veces necesarias.

Con esto ya tienes información suficiente para resolver el ejercicio, pero para que resulte más sencillo, aquí tienes instrucciones adicionales

1. La figura está formada una sucesión de líneas 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.
2. La figura puede descomponerse como:
   - Una barra horizontal.
   - Unos *barrotes*.
   - Otra barra horizontal.
   - Unos barrotes iguales a los anteriores.
   - Otra barra horizontal.
3. Los *barrotes* están formados por un número de líneas igual a *hueco*, donde cada línea está formada por
   - Tinta.
   - Tantos elementos *papel* como el valor de *hueco*.
   - Tinta.
   - Tantos elementos *papel* como el valor de *hueco*.
   - Tinta.

Solución

```
{£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 número 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 están compuestas de hueco-2 líneas
    // Cada línea 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 práctica 9.2 para que el programa no pida al usuario un código de aeropuerto, sino que elija un aeropuerto al azar, entre todos los disponibles. Una vez elegido, el programa se comportará de la misma forma que la versión anterior, esto es, mostrará el aeropuerto más cercano.

El programa deberá llamarse `~/fpi.junio.19/aleatorio.pas`. Asegúrate de que el nombre sea exactamente ese, de lo contrario, será como no haberlo escrito.

Ejercicio 3 (3.5 puntos)

En el fichero `~/fpi.junio.19/ejercicio_tabla.pas` encontrarás un programa que genera un array de valores aleatorios. Modifícalo para que calcule y muestre:

- El valor máximo.
- El valor mínimo.
- La media aritmética de los valores.

Puedes organizar tu código como creas conveniente, pero ten en cuenta que se valorará la calidad del diseño. Lo único que no puedes hacer es modificar los tres subprogramas que ya están escritos (*dado*, *escribe_valores* e *inicia_valores*).

Fundamentos de la programación y la informática
Examen de teoría. 11 de Junio de 2019
Grado en ingeniería aeroespacial en aeronavegación
Grado en ingeniería aeroespacial en vehículos aeroespaciales
Universidad Rey Juan Carlos

Preparativos

- Ejecuta el script `prepara`
- Esto creará 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 expresión lógica equivalente, más legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresión lógica equivalente, más legible, de nombre *s2*.
3. Escribe una expresión *n1* que sea la negación de *s1*.
4. Escribe una expresión *n2* que sea la negación de *s2*.

Solución

```
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 compilación, errores de ejecución, errores lógicos y defectos en la claridad del código. El siguiente programa está 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 compilación, ejecución o lógico.

- Obviamente, el número que aparece al principio de cada línea no forma parte del programa, es el número de línea. Para cada error que encuentres, indica en qué linea está.

  Por ejemplo, un error encontrado en el código podría describirse de la siguiente forma: *Línea 23: Error: El identificador 'inéxistente' no es válido porque contiene el carácter 'é', que no está permitido.*

```
 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 código y altitud de
 4 aeropuerto, luego suma todas las altitudes.}
 5
 6 const
 7     Num_aeropuertos = 2;
 8 type
 9     tipo_aeropuerto = record
10         código : 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;
28
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;
```

solucion

Línea 9: Defecto. Según el criterio que seguimos un tipo debería estar escrito en NotaciónCamello, es decir sin barra baja ni ningún otro elemento que separe las palabras y con la primera letra de cada palabra en mayúscula.

Línea 10: Error. El carácter "ó"no es válido para identificadores, habría que quitarle la tilde.

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

Línea 19: Error. El operador de asignación es :=, no =. Por tanto, la asignación debería ser i := 1, no i = 1.

Línea 19: Error. El bucle debería ser de 1 a Num_aeropuertos para recorrer el array en su totalidad.

Línea 21: Error. Sobra el ";.ª"l final de la línea ya que interrumpe el if then else.

Línea 23: Error. No es posible sumar "suma"que es un integer con .ªeropuertos[i].codigo"que es un string.

Líneas 22-25: Falta el begin-end del else. Tal y como está escrito el programa, la sentencia writeln está fuera del bucle, a pesar de que la tabulación indica lo contrario.

Línea 24: Error. Es una función que tiene efectos laterales.

Linea 26: Error. Debería poner simplemente result:= suma;", ya que es como se fija el resultado de una función. Además 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 ningún campo result.

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

Línea 40: Error. El argumento de un subprograma, en este caso writeln, no puede ser un procedimiento, porque un procedimiento no devuelve ningún valor.

Línea 41: Error. Sobra un paréntesis de apertura, debería ser writeln(suma_altitudes(tipo_aeropuerto));.

Línea 42: Error. El programa debe terminar con un "."no un ";".

Fundamentos de la programación y la informática
Examen práctico. 19 de diciembre de 2019
Grado en ingeniería aeroespacial en vehículos aeroespaciales
Universidad Rey Juan Carlos

Preparativos

- Ejecuta el script `~mortuno/prepara`

  Esto creará 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 tamaño variable y densidad variable, según la siguiente especificación

- Primero pedirá al usuario que indique el tamaño del cuadrado, un entero entre 5 y 25, ambos inclusive. Repetirá la petición hasta que el usuario haga lo indicado.
- Luego pedirá al usuario que indique la densidad del cuadrado, un entero entre 1 y 10, ambos inclusive. Repetirá la petición hasta que el usuario haga lo indicado.
- El programa escribirá en pantalla una figura a la que llamaremos *cuadrado de densidad variable*. Estará formado por N filas de N puntos, donde N es el tamaño. Cada punto podrá ser o bien un *punto relleno* o bien un *punto vacío*.
- Cuando la densidad sea 10, todos los puntos estarán rellenos. Cuando la densidad sea alta pero no 10 (por ejemplo 8), habrá muchos puntos rellenos y algunos vacíos. Cuando la densidad sea baja (por ejemplo 1) habrá muchos puntos vacíos y pocos puntos llenos. No permitiremos que la densidad sea nula porque entonces todos los puntos estarían vacíos, sería un cuadrado vacío 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 carácter X y el carácter espacio.
  3. En otro caso, escribiremos un *punto vacío*, 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 serán menores o iguales que 2, por tanto solo se rellenarán el 20 % de los puntos.

Aquí tienes un ejemplo de ejecución, tu programa tendrá que hacer algo similar esto:

```
koji@mazinger:~/fpi.practico.dic.19 ./densidad_cuadrado
Indica el tamaño del cuadrado
Introduce un número entero entre 5 y 25
7
Indica la densidad
Introduce un número 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 tamaño del cuadrado
Introduce un número entero entre 5 y 25
12
Indica la densidad
Introduce un número 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
```

Solución

http://ortuno.es/densidad_cuadrado.pas

Ejercicio 2 (6 puntos)

Este ejercicio está basado en tu prática 9.2. Copia tu fichero `~/fpi/practica09/busca_codigo.pas` en `~/fpi.practico.dic.19/distancia.pas` y haz que cumpla la siguiente especificación. Todo lo que este enunciado no diga, se entiende que ha de ser igual que en la práctica que hiciste en el laboratorio. [1]

1. Si todo ha ido bien y el programa ha encontrado el aeropuerto cuyo código ha introducido el usuario, no solo mostrará sus datos, sino que indicará cuál es el aeropuerto más cercano, a qué distancia se cuentra, cuál es el más lejano y a qué distancia se encuentra.

2. Naturalmente, para el punto anterior tu programa tendrá 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 demás del mundo, buscar el máximo y buscar el mínimo.

3. El aeropuerto más cercano a un aeropuerto es el mismo aeropuerto, que está 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 práctica 8.2 donde calculaste la distancia entre un punto del plano y el origen de coordenadas, para luego buscar el máximo y el mínimo. Pero observa que hay dos diferencias importantes:

   - En la práctica calculabas la distancia entre el punto y el origen de coordenadas. En este ejercicio calcularás 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 Pitágoras, pero en otro caso es necesario tener en cuenta la curvatura de la tierra. Esto lo resuelve la *fórmula del semiverseno*. En el fichero https://gsyc.urjc.es/~mortuno/haversine.pas encontrarás una función que implementa esta fórmula (*haversine* es la palabra inglesa para *semiverseno*). Compílalo, pruébalo y copia y pega en tu ejercicio todo lo que necesites.

[1] Nota para alumnos repetidores: este ejercicio se parece mucho a una práctica del año pasado, pero la estructura del array y el interface de usuario cambia un poco. Si algún ejercicio se corresponde exactamente con la especificación del año pasado y no con la de este, su nota será 0.

Solución

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

```
Escribe el codigo IATA del aeropuerto a buscar (3 caracteres)
AAAA
Código incorrecto. Por favor escribe 3 caracteres


Escribe el codigo IATA del aeropuerto a buscar (3 caracteres)
ZSY
Código no encontrado


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

Adolfo Suárez Madrid-Barajas Airport. Codigo IATA: MAD. Coordenadas: 40.472 -3.563

Aeropuerto más próximo:
Torrejón Airport. Codigo IATA: TOJ. Coordenadas: 40.497 -3.446. Distancia: 10.29km

Aeropuerto más lejano:
Palmerston North Airport. Codigo IATA: PMR. Coordenadas: -40.321 175.617. Distancia: 19965.99km
```

Fundamentos de la programación y la informática
Examen de teoría. 19 de diciembre de 2019
Grado en ingeniería aeroespacial en vehículos aeroespaciales
Universidad Rey Juan Carlos

Preparativos

- Ejecuta el script `~mortuno/prepara`
- Esto creará en tu ordenador el fichero `~/fpi.teoria.dic.19/teoria.TULOGIN.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 and c) or not (b or d or a) );

e2 := not  (x > 15) and not (y = 0);
```

1. A partir de *e1*, escribe una expresión lógica equivalente, más legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresión lógica equivalente, más legible, de nombre *s2*.
3. Escribe una expresión *n1* que sea la negación de *s1*.
4. Escribe una expresión *n2* que sea la negación de *s2*.

Solución

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, serán reemplazadas por un 10.

Así que escribe el programa mostrado a continuación 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 compilación, errores de ejecución, errores lógicos y defectos en la claridad del código. 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 compilación, ejecución o lógico.
- Obviamente, el número que aparece al principio de cada línea no forma parte del programa, es el número de línea. Para cada error que encuentres, indica en qué linea está.

  Por ejemplo, un error encontrado en el código podría describirse de la siguiente forma: *Línea 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;
03 uses crt;
04 const
05     tamanio_array : 8;
06 type
07     TipoTabla : array[1..tamanio_array] of integer;
08 var
09     tabla_bruto, tabla_filtrada = TipoTabla;
10
11
12 function tira_dado(caras_dado: integer):integer;
13 begin
14     result :=  random( caras_dado ) + 1 ;
15 end;
16
17
18 procedure rellena_tabla(var tabla:TipoTabla; valor_maximo:integer);
19 var
20     i : integer;
21 begin
22     for i := 1 to tamanio_array do
23         tabla[i] := tira_dado(valor_maximo);
24 end;
25
26
27 procedure filtra_tabla(tabla_entrada, tabla_salida:TipoTabla; limite_Saturación:integer);
28 var
29     i : integer;
30 begin
31     for i = 1 to tamanio_array do begin
32         if tabla_entrada[i] < limite_Saturación then
33             tabla_salida[i] := limite_Saturación;
34         else
35             tabla_salida[i] := tabla_entrada[i]
36         end;
37 end;
38
39
40 procedure muestra_tabla(var tabla: TipoTabla);
41 var
42     i : integer;
43 begin
44     for i = 1 to tamanio_array do
45         writeln(tabla[i]);
46 end;
47
48
49 const
50     ValorMaximo = 15;
51     LimiteSaturación = 10;
```

```
52 begin
53     rellena_tabla(tabla_bruto, ValorMaximo);
54     tabla_filtrada := filtra_tabla(tabla_bruto,  LimiteSaturación);
55     writeln('Tabla en bruto:');
56     muestra_tabla(tabla_bruto);
57     writeln('Tabla filtrada:');
58     muestra_tabla(tabla_filtrada);
59 end;
```

Solución:

(En la página siguiente)

Código dado en el examen (mal)

```
programa filtra_tabla;
uses crt;
const
    tamanio_array : 8;
type
    TipoTabla : array[1..tamanio_array] of integer;
var
    tabla_bruto, tabla_filtrada = TipoTabla;


function tira_dado(caras_dado: integer):integer;
begin
    result := random( caras_dado ) + 1 ;
end;


procedure rellena_tabla(var tabla:TipoTabla; valor_maximo:integer);
var
    i : integer;
begin
    for i := 1 to tamanio_array do
        tabla[i] := tira_dado(valor_maximo);
end;


procedure filtra_tabla(tabla_entrada, tabla_salida:TipoTabla; limite_Saturación:integer);
var
    i : integer;
begin
    for i = 1 to tamanio_array do begin
        if tabla entrada[i] < limite Saturación then
            tabla_salida[i] := limite_Saturación;
        else
            tabla_salida[i] := tabla_entrada[i]
        end;
end;


procedure muestra_tabla(var tabla: TipoTabla);
var
    i : integer;
begin
    for i = 1 to tamanio_array do
        writeln(tabla[i]);
end;




const
    ValorMaximo = 15;
    LimiteSaturación = 10;

begin



    rellena tabla(tabla bruto, ValorMaximo);
    tabla filtrada := filtra tabla(tabla_bruto, LimiteSaturación);
    writeln('Tabla en bruto:');
    muestra tabla(tabla bruto);
    writeln('Tabla filtrada:');
    muestra_tabla(tabla_filtrada);
end;
```

Solución (bien)

```
program filtra_tabla;
uses crt;
const
    TamanioArray = 8;
type
    TipoTabla = array[1..TamanioArray] of integer;




function tira_dado(caras_dado: integer):integer;
begin
    result := random( caras_dado ) + 1 ;
end;


procedure rellena_tabla(var tabla:TipoTabla; valor_maximo:integer);
var
    i : integer;
begin
    for i := 1 to TamanioArray do
        tabla[i] := tira_dado(valor_maximo)-1;
end;

procedure filtra_tabla(var tabla_entrada, tabla_salida:TipoTabla; limite_saturacion:integer);

var
    i : integer;
begin
    for i := 1 to TamanioArray do begin
        if tabla entrada[i] > limite saturacion then
            tabla_salida[i] := limite_saturacion
        else
            tabla_salida[i] := tabla_entrada[i]
        end;
end;


procedure muestra_tabla(var tabla: TipoTabla);
var
    i : integer;
begin
    for i := 1 to TamanioArray do
        writeln(tabla[i]);
end;


var
    tabla_bruto, tabla_filtrada : TipoTabla;
const
    ValorMaximo = 16; // Generaremos valores entre 1 y 16, luego restaremos 1
                      // para que vaya de 0 a 15
    LimiteSaturacion = 10;
begin
    delay(1000);
    randomize;

    rellena tabla(tabla bruto, ValorMaximo);
    filtra tabla(tabla bruto, tabla_filtrada, LimiteSaturacion);
    writeln('Tabla en bruto:');
    muestra tabla(tabla bruto);
    writeln('Tabla filtrada:');
    muestra_tabla(tabla_filtrada);
end.
```

Fundamentos de la programación y la informática
Examen de ejercicios de laboratorios
1 de Febrero de 2021
Grados en ingeniería aeroespacial. Turno de tarde
Universidad Rey Juan Carlos

Ejercicio 1 (5 puntos)

En tu cuenta del laboratorio encontrarás los siguiente ficheros [1].

- `~/fpi2021feb/todos_mayores.pas`
- `~/fpi2021feb/limita_matriz.TULOGIN.pas`.

El programa `todos_mayores.pas` deberías conocerlo. Está 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**. Está aquí por si necesitas copiarlo de nuevo o consultarlo.

El fichero `limita_matriz.TULOGIN.pas` de momento es una copia idéntica del programa fichero anterior. **Modifica el fichero limita_matriz.TULOGIN.pas** para que cumpla la siguiente especificación:

1. El programa generará una matriz de números aleatorios y la mostrará por pantalla, exactamente igual que en la versión anterior.
2. El programa generará una nueva matriz, donde los valores mayores que k serán reemplazados por k.
3. El resultado será 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
```

4. 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 tendrá 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 posición
     ```

   - Debes hacer algo similar a esto:

     ```
     escribe_matriz(matriz);
     [bla bla blá]
     escribe_matriz(matriz);
     ```

     O bien

     ```
     escribe_matriz(matriz);
     [bla bla blá]
     escribe_matriz(matriz_modificada);
     ```

[1] TULOGIN será tu nombre de usuario en el laboratorio

Solución

```
{$mode objfpc}{$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 encontrarás el fichero

- `~/fpi2021feb/dni.TULOGIN.pas`

Que contiene el siguiente trozo de código

```
{£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;
```

Complétalo 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* será un entero aleatorio entre 1 (incluido) y 99999999 (100 millones menos una unidad)
- El campo *genero* será o bien el carácter M o bien el carácter F (mayúsculas). Elegirá uno u otro tirando un dado de dos caras.

Ejemplo de ejecución:

```
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.

Solución

```
{£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 programación y la informática
Examen de entrega de prácticas
1 de Febrero de 2021
Grados en ingeniería aeroespacial. Turno de tarde
Universidad Rey Juan Carlos

Ejercicio único

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 especificación:

1. Tendrá una constante global llamada `MaxRepes`. Ponle por ejemplo el valor 3.
2. El programa generará una matriz donde los valores podrán estar repetidos, pero como mucho cada valor aparecerá un total de `MaxRepes` veces.
3. Comprobará que se cumple la precondición de que $CarasDado * MaxRepes <= Filas * Columnas$

Observaciones

- En tu programa original deberías tener una función que te indicaba si un número estaba o no en la matriz. Reemplázala por una que indique cuántas veces aparece el número.
- Supongamos que MaxRepes vale 3. Para rellenar una posición, tendrás que generar números hasta obtener uno que, como mucho, haya aparecido hasta el momento 2 veces. Una vez que añadas a la matriz el número generado, serán 3.

Ejemplo de ejecución:

```
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 ejecución:

```
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 programación y la informática
Test final
1 de Febrero de 2021
Grados en ingeniería 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 deberás resolver el ejercicio [1].

Ejercicio único (10 puntos)

Cualquiera que tenga nociones de álgebra sabe que $\frac{(a+b-b)*b}{b} = a$. Sin embargo, el código a continuación parece demostrar lo contrario. Obviamente el programa tiene que tener algún problema. Indica:

1. Por qué parece demostrar lo contrario.
2. Qué error o errores hay en el código.
3. Por qué se producen.
4. Cómo se deberían 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 ejecución:

```
FALSE
TRUE
```

[1] TULOGIN será 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 fórmula indicada en el enunciado, compara el valor tras estas operaciones con el valor original, y aunque deberían ser siempre iguales, en un caso no lo es. Concretamente cuando *pizca* vale una diezmillonésima.

   Esto se debe a los errores de conversión propios de la representación de números reales en *coma flotante* (ver tema 2, pg 43-46)

   Para solucionarlo, nunca deberíamos comparar que dos números reales sean iguales. Deberíamos buscar que la diferencia entre los dos números sea menor que cierto valor, despreciable en nuestro ámbito (ver tema 8, pg 25-26)[2].

2. Las variables x y *pizca* son globales, porque están declaradas antes que los subprogramas. Deberíamos declaralas inmediatamente antes del cuerpo del programa principal.

[2]Para reducir estos errores podríamos usar tipos de datos reales con mayor precisión, pero nunca los podremos eliminar por completo, siempre tendremos que buscar una diferencia lo bastante pequeña

Fundamentos de la programación y la informática
Examen de ejercicios de laboratorios
8 de Julio de 2021
Grados en ingeniería aeroespacial. Turno de tarde
Universidad Rey Juan Carlos

Ejercicio 1 (5 puntos)

En tu cuenta del laboratorio encontrarás el siguiente fichero [1].

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

Escribe en él un programa que tenga una función 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`
Devolverá la cadena
`juan[punto]perez[arroba]urjc[punto]es`

Ejercicio 2 (5 puntos)

En tu cuenta del laboratorio encontrarás un programa que genera una matriz aleatoria y la escribe en pantalla, está en el fichero

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

Añade los subprogramas necesarios para generar un programa que genere otra matriz de las mismas dimensiones, pero con todos sus elementos iguales. Este valor será 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 deberá *generar* otra matriz, esto es, otro elemento de tipo `TipoMatriz`. No basta con mostrar un texto en pantalla.

[1] TULOGIN será tu nombre de usuario en el laboratorio

Fundamentos de la programación y la informática
Examen de entrega de prácticas
8 de Julio de 2021
Grados en ingeniería aeroespacial. Turno de tarde
Universidad Rey Juan Carlos

Ejercicio 1 (5 puntos)

Ahora modificarás tu práctica 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 especificación:

1. El código 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 código en el cuerpo del programa principal, entonces deberás modificarlo). Este procedimiento puede llamar a otros subprogramas. O no, ambos enfoques son válidos.
2. Este procedimiento debe devolver, en un parámetro por referencia, el valor entero positivo escrito por el usuario.
3. Este procedimiento debe devolver, en otro parámetro por referencia, el número de veces que el usuario se ha equivocado.

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

Ejercicio 2 (5 puntos)

Este ejercicio es una modificación de tu práctica 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 usuario no escriba reales sino enteros. Además, en vez de mostrar el máximo y el mínimo de los valores introducidos, el programa deberá escribir:

- El número de valores pares y el número de valores impares.
- El máximo de los valores pares y el máximo de los valores impares.
- El mínimo de los valores pares y el mínimo de los valores impares.

Fundamentos de la programación y la informática
Test parcial
8 de Julio de 2021
Grados en ingeniería 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 deberás resolver el ejercicio [1].

Ejercicio único (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 expresión lógica equivalente, más legible, de nombre s1
2. A partir de e2 escribe una expresión lógica equivalente, más legible, de nombre s2
3. Escribe una expresión n1 que sea la negación de s1
4. Escribe una expresión n2 que sea la negación de s2

[1] TULOGIN será tu nombre de usuario en el laboratorio

Fundamentos de la programación y la informática
Test final
8 de Julio de 2021
Grados en ingeniería 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 deberás contestar las siguientes preguntas [1].

Ejercicio 1 (4 puntos)

Explica brevemente la siguiente afirmación: *una función no debería tener efectos laterales.*

Ejercicio 2 (3 puntos)

¿Para qué sirven los registros?

Ejercicio 3 (3 puntos)

¿Por qué decimos que los punteros son *peligrosos*?

[1] TULOGIN será tu nombre de usuario en el laboratorio

Fundamentos de la programación y la informática
Examen Parcial. 23 de Noviembre de 2021
Grados en ingeniería 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 deberás resolver el ejercicio 1 [1]..
  `~/fpi.nov.21/precio.TULOGIN.txt`, donde deberás resolver el ejercicio 2.
  `~/fpi.nov.21/ejemplo`, un ejecutable que puedes usar como referencia para ver cómo debería 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 expresión lógica equivalente, intentando que sea más clara, de nombre s1
2. A partir de e2, escribe una expresión lógica equivalente, más legible, de nombre s2
3. Escribe una expresión n1 que sea la negación de s1
4. Escribe una expresión n2 que sea la negación de s2

Solución

```
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 añade 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 indicará el tipo del IVA escribiendo *g*, *r* o *s*, en mínúscula. Los detalles puedes verlos en esta ejemplo de ejecución:

[1] TULOGIN será 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 número en las dos primeras preguntas o no escribe ni *g* ni *r* ni *s* en la tercera pregunta), el programa mostrará un error. Aquí 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 práctica 5.5.
- Escribe un procedimiento *intenta_leer_tipo_iva*, similar a *intenta_leer_real*, que le haga la pregunta al usuario, que devuelva el carácter que ha escrito el usuario y también un booleano *TRUE* si la respuesta es correcta (*g*, *r* o *s*, en minúscula) o FALSE, en otro caso.
- Escribe una función que a partir del carácter *g*, *r* o *s*, devuelva un real con el tipo del impuesto (21, 10 o 4).
- Calcula así 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;
```

Solución

```
{$mode objfpc}{$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
    else
        writeln('Has escrito algo mal');
end.
```

Fundamentos de la programación y la informática
Examen de entrega de prácticas
19 de Enero de 2022
Grados en ingeniería aeroespacial. Turno de tarde
Universidad Rey Juan Carlos

Ejercicio 1 (5 puntos)

En este ejercicio modificarás tu práctica 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 especificación:

1. El procedimiento *lee_real* hará lo mismo que hasta ahora y devolverá las mismas cosas. Pero, además, devolverá también un entero indicando cuántas veces se ha equivocado el usuario, esto es, cuántas veces escribió algo distinto de un número real. Si lo escribió bien a la primera, este valor será cero.
2. Tienes libertad para diseñar los retoques de este procedimiento.
3. En alguna parte del programa, debes escribir en pantalla el número de veces que el usuario se ha equivocado. Tienes libertad para decidir cómo y donde.
4. Evidentemente, si estas *libertades* las implementas de forma contradictoria con los principios de buen diseño que hemos visto durante el curso, la nota tendrá la penalización correspondiente.

Si tu programa no compila, su nota será nula.

Ejercicio 2 (5 puntos)

Ahora modificarás tu práctica 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

- Además de mostrar la misma información que la versión anterior, indique al final de su ejecución (y solo al final de su ejecución) cuál ha sido la media más baja y la media más alta de todas las obtenidas.

Si tu programa no compila, su nota será nula.

Fundamentos de la programación y la informática
Examen de ejercicios de laboratorios
19 de Enero de 2022
Grados en ingeniería 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].

Ejercicio 1 (4 puntos)

Edita el fichero `~/fpi.enero.22/cadenas.TULOGIN.pas` para escribir un programa en Pascal que tenga una función 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`
Devolverá la cadena
`[hashtag]examen-fpi[at]urjc`
Si el programa no compila, su nota será nula.

Ejercicio 2 (6 puntos)

En el fichero `~/fpi.enero.22/diagonales.TULOGIN.pas` encontrarás un programa que genera una matriz aleatoria y la escribe en pantalla. Añade los subprogramas necesarios para generar un programa que

- Compruebe la precondición de que la matriz es cuadrada (tiene el mismo número de filas que de columnas)
- Si esta precondición se cumple, mostrará 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 precondición no se cumple, mostrará un mensaje de error.

Observaciones

- Se denomina *diagonal principal* de una matriz cuadrada al conjunto de elementos cuyo primer índice es igual al segundo índice. Por ejemplo para matrices cuadradas de tres filas son los elementos $a_{1,1}a_{2,2}a_{3,3}$

[1] TULOGIN será tu nombre de usuario en el laboratorio

- Se denomina *diagonal secundaria* de una matriz cuadrada al conjunto de elementos cuyos índices i,j cumplen la condición de sumar n+1, siendo n el número 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 (índice i) y calcular la columna (índice j) a partir de la expresión $n - i + 1$

- Obviamente, este programa tiene que seguir funcionando si modificamos el valor de las constantes *Filas* y *Columnas*, no es válido que escribas un programa que solo funciona si el número de filas es exactamente 3.

- Si el programa no compila, su nota será nula.

Solución

```
{£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 programación y la informática
Test parcial
19 de Enero de 2022
Grados en ingeniería 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 deberás resolver el ejercicio 2 [1].

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 expresión lógica equivalente, más legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresión lógica equivalente, más legible, de nombre *s2*.
3. Escribe una expresión *n1* que sea la negación de *s1*.
4. Escribe una expresión *n2* que sea la negación de *s2*.

Solución

```
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 qué líneas 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.
```

[1] TULOGIN será tu nombre de usuario en el laboratorio

Solución

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

```
program pts_eur;

// Normalmente los números deben definirse como constantes, no
// usar 'números mágicos'
const PtsEur = 166.386;

// Este subprograma se invoca como función, así que debe ser una
// función, no un procedimiento (o si se usa como procedimiento,
// invocarlo correctamente)
function pts_eur(pts:real): real;
begin
    result := pts / PtsEur;
end;

// Si llamamos al parámetro 'eur', no podemos escribir la expresión
// con el identificador 'euros'
procedure pts_eur(pts:real;  var eur: real );
begin
    eur := pts / PtsEur;
end;

// Las variables deben ser locales
var
    valor_ejemplo : real = 1000;
    valor_convertido : real;
// Las variables deben usarse con el mismo nombre con el que se
// han definido

begin
    // pts ->  eur
    // El operador de asignación es :=, no =
    valor_convertido := pts_eur(valor_ejemplo) ;

    // Las sentencias acaban en punto y coma
    pts_eur(valor_ejemplo, valor_convertido);
    write(valor_ejemplo:0:2, ' pts son ');
    writeln(valor_convertido:0:2, ' euros');
end.
```

Fundamentos de la programación y la informática
Test final
19 de Enero de 2022

Grados en ingeniería 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 deberás resolver el ejercicio [1].

Dados los párrafos siguientes, indica para cada uno de ellos si lo que dice es cierto o es falso. Justifica brevemente tu respuesta. Si hay más de un error, señálalos todos. Ambas cosas (justificar las respuestas y señalar todos los errores) son imprescindibles: si de cada párrafo lo único que dices es *cierto* o *falso*, tu respuesta no tendrá casi ningún valor.

Párrafo 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 número de clientes, ya sean 2, 2000 o 20000 (siempre que el ordenador sea lo bastante potente)

Si usamos un lenguajes de programación *tradicional* como Pascal o C, esto no es demasiado complicado, porque siempre podemos usar *memoria estática*, esto es, punteros.

Lenguajes más modernos como Python o Java permiten usar *memoria dinámica*, esto es, arrays. También sirven aunque a priori no conozcamos el número de clientes, pero son más complicados de programar.

Párrafo 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 llenará de *datos basura* y posiblemente se perderá 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 dinámica, no memoria estática.
   - Los lenguajes más modernos se ocupan automática de la gestión de la memoria dinámica. No es necesario que lo gestione el programador, por lo que resulta más sencillo.

[1] TULOGIN será tu nombre de usuario en el laboratorio

2. Falso. El programador debería cerrar los ficheros, pero si se olvida, no suele resultar fatal porque al acabar la ejecución del programa, el sistema operativo se ocupa de cerrarlos [2].

[2]El 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 podrán acceder al fichero. Otro problema puede darse si hay muchas instancias del mismo programa abriendo un fichero en modo lectura, sin cerrarlo. Podrían superar el número máximo de veces que se puede abrir un fichero. No es demasiado frecuente porque este número suele ser alto, pero puede suceder.

Fundamentos de la programación y la informática
Examen de ejercicios de laboratorio
15 de Junio de 2022

Grados en ingeniería 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].

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 especificación:

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 será una constante.
3. A la cara le corresponde un valor de un punto, a la cruz, ninguno.
4. El resultado de cada tirada se mostrará en una línea de la pantalla, esto es, las monedas que han salido en esa tirada y la puntuación de esa tirada.
5. El juego seguirá mientras no se alcance una *puntuación_objetivo*, por ejemplo 4 puntos. Este valor será una constante. Cuando la puntuación de una tirada sea mayor o igual que el objetivo, el juego se dentendrá. Observa que hablamos siempre de la puntuación en una *tirada*, (p.e. 5 monedas). No de la suma de todas las tiradas.

Ejemplo de ejecución:

```
cxcxx 2
xcxxc 2
ccxcx 3
xxxcc 2
xxcxx 1
ccccc 5
```

[1] TULOGIN será tu nombre de usuario en el laboratorio

Solución

```
{$mode objfpc}{$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 número mágico 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;
         writeln(' ',puntuacion);
    end
end.
```

Ejercicio 2 (5 puntos)

Escribe un programa en Pascal en el fichero `~/fpi.junio.22/mezcla.TULOGIN.pas`, que contenga y use una función que:

1. Reciba dos cadenas de entrada.
2. Si la longitud de ambas cadenas de entrada no es igual, devolverá la cadena *ERROR*.
3. Si la longitud es igual, devolverá la cadena resultante de *mezclar* ambas cadenas de la siguiente forma:
    - Primer carácter de la primera cadena.
    - Primer carácter de la segunda cadena.
    - Segundo carácter de la primera cadena.
    - Segundo carácter 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 programación y la informática
Examen de entrega de ejercicios
15 de Junio de 2022

Grados en ingeniería 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].
- En caso de que alguno de los programas que entregues hoy no compile, la nota de ese ejercicio será prácticamente nula.

Ejercicio 1 (6 puntos)

En tu práctica 8.3 escribiste un programa llamado *unicos.pas* que ahora encontrarás en el fichero `~/fpi.junio.22/crecientes.TULOGIN.pas`. Modifícalo para que cumpla la siguiente especificación:

1. El dado tenga un número de caras bastante alto, p.e. 100.
2. Los valores que se van añadiendo a la matriz no solo sean únicos, sino además, estrictamente crecientes. Esto es:
   a) Cada número no solamente no podrá repetirse, sino que deberá ser mayor que el máximo que haya salido hasta el momento
   b) Cuando salga un número igual al máximo (p.e. 100) , ya será imposible generar valores mayores. En ese caso, el programa mostrará un mensaje de error y dejará de generar valores.

Sugerencia:

- Vete controlando el máximo valor aparecido en la matriz
- Cuando tu programa empieza a buscar números hasta encontrar uno nuevo, cambia la condición, de forma que no solo sea un valor no repetido sino, además, mayor que el máximo.

Ejemplo de ejecución:

```
2     4     23
31    46    59
72    82    89
```

[1] TULOGIN será tu nombre de usuario en el laboratorio

Ejercicio 2 (4 puntos)

En tu práctica 7.3 escribiste un programa llamado *filtrado03.pas*. que ahora encontrarás en el fichero `~/fpi.junio.22/filtrado04.TULOGIN.pas`. Modifícalo para que escriba toda la información relevante de su ejecución no solo en pantalla, sino en el fichero *resultado.txt*

- Entendemos como *relevantes* todos los mensajes que aparecen en pantalla, excepto aquellos dirigidos al usuario para preguntarle información. P.e. *introduce el radio*, *introduce la superficie alar*. Estos mensajes **no** son relevante.
- Cuando el usuario escriba un valor (correcto o incorrecto), esto será relevante. Así que el programa escribirá mensajes como

  El usuario escribe 23

  El usuario escribe j41

  La masa son 2320 Kg

Fundamentos de la programación y la informática
Test de junio (correspondiente al test parcial y test final de enero)
15 de Junio de 2022

Grados en ingeniería 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 deberás resolver el ejercicio [1].

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 expresión lógica equivalente, más legible, de nombre *s1*.
2. A partir de *e2*, escribe una expresión lógica equivalente, más legible, de nombre *s2*.
3. Escribe una expresión *n1* que sea la negación de *s1*.
4. Escribe una expresión *n2* que sea la negación 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 párrafos siguientes, indica para cada uno de ellos si lo que dice es cierto o es falso. Justifica brevemente tu respuesta. Si hay más de un error, señálalos todos. Ambas cosas (justificar las respuestas y señalar todos los errores) son imprescindibles: si de cada párrafo lo único que dices es *cierto* o *falso*, tu respuesta no tendrá casi ningún valor.

[1] TULOGIN será tu nombre de usuario en el laboratorio

Párrafo 1

Es conveniente que el programador abra y cierre todos los ficheros que utilice. Pero si lo olvida, el sistema operativo se encargará. Esto resultará menos eficiente y puede dar problemas en algún caso particular, pero no será un error fatal.

Párrafo 2

Los ordenadores representan internamente los número 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 prácticamente cualquier programador en cualquier lenguaje haga cálculos con todos los decimales que necesite. Será más o menos cómodo o más o menos *bonito* para las personas, pero los cálculos siempre serán exactos.

Respuesta

1. Falso. Abrir es imprescindible, de lo contrario el programa dará un error. Cerrar es conveniente, si el programador olvida cerrar, el sistema operativo lo cerrará. Pero solamente el programador puede abrir.

2. Es cierto que todos los ordenadores representan internamente los número reales (y todos los demás datos) en binario. Es cierto que un voltaje superior a cierto umbral es un uno y un cero en otro caso [2]. Pero es falso que todos los cálculos serán siempre exactos con todos los decimales necesarios. Como vimos en el tema 2, apartado *representación de los números reales*, la conversión de un número desde decimal hasta binario provoca errores a partir de cierto número de decimales. El programador puede elegir tipos de datos con mayor precisión, de entre los que le ofrezca el lenguaje. Pero los lenguajes de programación convencionales siempre tienen errores a partir de cierto número de decimales.

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