Fundamentos de la programación y la informática
Grados en ingeniería aeroespacial, 2022-2023
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Práctica 1.1. Uso básico de la shell

En este ejercicio probarás las órdenes básicas de la shell en el laboratorio. Usando un cliente de ssh adecuado para tu sistema operativo (SmarTTY, Terminal o cualquier otro si lo prefieres), abre una sesión en una máquina del laboratorio y

1. Observa el *prompt* y fíjate en sus componentes
2. Crea el directorio `fpi`
3. Prueba `tree` para ver la estructura actual
4. Crea los directorios `~/fpi/dir10`, `~/fpi/Dir20` y `~/fpi/dir30`
   Fíjate en el uso de mayúsculas y minúsculas
5. Entra en cada uno de ellos y haz un listado de su contenido. Observa cómo va cambiando el *prompt*
6. Crea los directorios `~/fpi/dir10/dir11`, `~/fpi/dir10/dir12`, `~/fpi/Dir20/dir21` y `~/fpi/dir30/dir31`
7. Comprueba con `tree` que la estructura es la correcta
8. Borra el directorio `~/fpi/Dir20`
9. Crea los directorios `~/fpi/dir20` y `~/fpi/dir20/dir21` (Observa que lo que cambia son las mayúsculas. Aquí recomendamos evitar mayúsculas en los nombres de ficheros)
10. Comprueba con `tree` que la estructura es la correcta

Práctica 1.2. Uso básico de nano

- Redacta un pequeño documento llamado `~/fpi/ejemplo.txt` enumerando tres o cuatro asignaturas de las que te hayas matriculado este año, describiendo de forma informal de qué tratan. Se trata solo de que pruebes el editor, no importa mucho lo que escribas.
- Configura nano para que te indique la línea y columna en la que está el cursor.

Práctica 1.3. Hola Mundo

Ahora te ejercitarás sobre la compilación y ejecución de ficheros, así como copiar y pegar texto entre ellos.

1. Escribe un *holamundo* como este

```
program holamundo;
begin
    writeln('Hola, mundo');
end.
```

(sin directivas del compilador) en el fichero `~/fpi/practica01/holamundo.pas`

Es importante que respetes al pie de la letra el nombre de todos los ficheros especificados en los guiones de prácticas. Esto incluye mayúsculas y minúsculas (usaremos siempre minúscula). La recogida será automática, un error en una letra supondrá un no presentado (aunque tendrás un script para revisar que no hay errores en los nombres)

2. Compílalo.
3. Ejecútalo.
4. Escribe en tu cuenta un fichero con el nombre `~/fpi/practica01/nota_fpi.pas`

   cuyo contenido sea el fichero que encontrarás en

   https://gsyc.urjc.es/~mortuno/nota_fpi.pas

   Si copias y pegas directamente con el ratón en nano tal vez se pierda la tabulación.

   Para evitarlo, ejecuta en el terminal

   `wget https://gsyc.urjc.es/~mortuno/nota_fpi.pas`

   La orden *wget* sirve para descargar una página web: se le pasa como argumento la dirección web de un fichero, lo descarga y lo escribe en el directorio actual.
5. Compílalo y ejecútalo.
6. Ahora añade al *holamundo* las directivas del compilador. Podrías escribirlas a mano, pero mejor cópialas desde el fichero `nota_fpi.pas`. (La línea con las directivas del compilador será siempre la misma en todos los programas que haremos durante el curso)
7. Compila y ejecuta.

Práctica 1.4. Errores de sintaxis

1. Haz una copia de `~/fpi/practica01/nota_fpi.pas` con el nombre `~/fpi/practica01/nota_fpi_02.pas`

   y otra con el nombre

   `~/fpi/practica01/nota_fpi_03.pas`

   Observa que puedes hacer esto con nano: abre el fichero, escríbelo, y en vez de mantener el nombre original como te ofrece el editor por omisión, elige un nombre nuevo. Así habrás copiado el fichero.
2. Modifica estos dos ficheros para que tengan algún error de sintaxis. (excepto la ausencia de punto y coma o el punto final, que son los ejemplos vistos en clase). Comprueba los errores generados.

   Pruébalo con error, comprueba que falla. Comenta el error, comprueba que ya no falla nada más. Explica en un comentario dónde está el error. Luego puedes dejar la versión con el error o la versión con el error inhabilitado, como prefieras.

Práctica 1.5. Modificación de un programa

1. Vuelve a copiar el programa anterior, esta vez con nombre `~/fpi/practica01/nota_fpi_04.pas`

   Modifícalo para que el compensable sea el 4.5. Compila y ejecuta. Cambia las notas de jperez 2 o 3 veces, para comprobar que todo funciona como debe.

Práctica 1.6. Errores lógicos

1. Haz otras dos copias del programa `nota_fpi.pas` con nombres `~/fpi/practica01/nota_fpi_05.pas` y `~/fpi/practica01/nota_fpi_06.pas`

   Modifícalos para que generen cada uno un error lógico distinto (y solo un error lógico). Añade un comentario en cada programa indicando dónde está el error.

Fundamentos de la programación y la informática
Grados en ingeniería aeroespacial, turno de tarde 2022-2023
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Práctica 2.1. Casting

En este ejercicio probarás el *casting* de tipos de datos

1. Crea el directorio

   `~/fpi/practica02`

2. Configura tu editor para que muestre los caracteres invisibles. Observa cómo representa los tabuladores y cómo representa los espacios. Presta mucha atención al sangrado de tu programa, en este ejercicio y en todos los que escribas durante el resto del curso.

3. Escribe un programa llamado `~/fpi/practica02/ahormados.pas`

   Será parecido al programa *casting* del tema 2, pero no idéntico.

4. El programa debe hacer al menos 4 conversiones de tipos de dato correctas.

5. Escribe también un par de ejemplos incorrectos. Una vez que compruebes que dan error de compilación, *comenta* esas líneas.

   (Comentar una línea es una expresión habitual en programación. No significa que hagas un comentario explicando la línea, sino que deshabilites la línea convirtiéndola en un comentario. No tendrá efecto, pero seguirá presente en el código para que el programador pueda verla)

6. Usa solamente constantes y expresiones. Ni funciones ni variables ni sentencias if-then-else.

Práctica 2.2. IVA

Escribe un programa llamado `~/fpi/practica02/iva.pas` que, a partir de un precio sin IVA, muestre el importe del IVA y el precio final con IVA.

Y que a partir de una constante con un precio con IVA, muestre el importe del IVA y el precio sin IVA.

- En todos los casos entenderemos siempre que se trata del IVA general (el 21 %).
- En contabilidad hay ciertas normas sobre el redondeo de los céntimos del IVA. Pero aquí lo ignoraremos.
- Observa que este programa será similar a los ejemplos *descuento02.pas* y *area_triangulo.pas* del tema 2 de las transparencias.
- Usa solamente constantes y expresiones. Ni funciones ni variables ni sentencias if-then-else.

Ejemplo:
Al ejecutar tu programa, saldrá algo similar a esto:

```
Precio sin IVA: 200.00€  IVA: 42.00€   Precio con IVA: 242.00€

Precio con IVA: 1000€   IVA: 173.55€  Precio sin IVA: 826.44€
```

Práctica 2.3. Sustentación

Escribe un programa llamado `~/fpi/practica02/sustentacion.pas` que calcule la fuerza de sustentación de un avión en Newtons a partir de la densidad del aire, la velocidad del avión en m/s, la superficie alar y el coeficiente de sustentación. Solo puedes usar constantes y expresiones (no funciones ni sentencias if-then-else). Invéntante los valores de entrada.

- Emplea la notación de esta página de wikipedia.
- Usa solamente constantes y expresiones. Ni funciones ni variables ni sentencias if-then-else.

Fundamentos de la programación y la informática
Grados en ingeniería aeroespacial, turno de tarde, 2022-2023
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Práctica 3.1. IVA (con funciones)

Escribe un programa llamado `~/fpi/practica03/iva.pas` que haga lo mismo que el programa del ejercicio 2.2, pero empleando funciones.

- Usa solamente funciones, constantes y expresiones. No uses variables ni sentencias if-then-else.

Práctica 3.2. Sustentación (con funciones)

Escribe un programa llamado `~/fpi/practica03/sustentacion.pas` que haga lo mismo que el programa del ejercicio 2.5, con la misma fórmula, pero empleando funciones.

- Usa solamente funciones, constantes y expresiones. No uses variables ni sentencias if-then-else.

Práctica 3.3. Múltiplos

Escribe un programa en Pascal llamado `~/fpi/practica03/multiplos.pas` que:

- Dados tres caracteres, C1, C2 y C3, que se corresponden los tres dígitos de un número (en base 10), escriba si el número resultante es múltiplo de 5 o no.

  Por ejemplo, dados los caracteres '0' '4' '7', el programa ha de escribir FALSE, dado que 47 no es múltiplo de 5.

- Recuerda dividir el problema en sub-problemas y resolver cada uno de ellos con una función.
- Usa solamente funciones, constantes y expresiones. No uses variables ni sentencias if-then-else.

Pautas:

- Convierte los 3 caracteres en 3 dígitos, los 3 dígitos en un número y comprueba si el módulo de la división entera del número entre 5 es 0.
- Para convertir un caracter en un dígito, toma como punto de partida la función *integer* que, a partir de un carácter, devuelve el número entero correspondiente en la tabla ASCII. Por ejemplo del caracter `7` el código ASCII es 55. Réstale el código ASCII del carácter `0` y tendrás el número que necesitas. En este caso, `7`.
- Para convertir tres dígitos en un número observa el siguiente ejemplo.

  $496 = 4 \times 100 + 9 \times 10 + 6$

Práctica 3.4. Velocidad de despegue

Escribe un programa en Pascal llamado `~/fpi/practica03/velocidad_despegue.pas` que calcule las velocidades de despegue de los aviones Airbus A380 y Airbus A320,

- Para ello, el programa incluirá una función que calcule la velocidad de despegue de un avión genérico, dados los siguientes datos: densidad del aire, masa, coeficiente de sustentación y superficie alar del avión. Recuerda que la velocidad de despegue es la velocidad cuando la sustentación es igual al peso.

- Las velocidades deberán obtenerse tanto en m/s como en Km/h.
- Utiliza la fórmula de la sustentación empleada en la práctica anterior, despejando la velocidad. Utiliza en la fórmula la misma notación que en la práctica 2.5.
- Usa solamente funciones, constantes y expresiones. Ni variables ni sentencias if-then-else.

Datos:

- Densidad del aire: 1.225 kg por metro cúbico.
- Airbus A380:
  - Masa: 560 toneladas
  - Superficie alar: 845 metros cuadrados
  - Coeficiente de sustentación: 2.4
- Airbus A320:
  - Masa: 78 toneladas
  - Superficie alar: 122 metros cuadrados
  - Coeficiente de sustentación: 2.3

Fundamentos de la programación y la informática
Grados en ingeniería aeroespacial, turno de tarde, 2022-2023
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Cambios en el documento

- Práctica 4.4. Donde decía *la magnitud será un número entero* ahora dice *la magnitud será un número real*

Práctica 4.1. Temperatura

En wikipedia podemos leer lo siguiente:

- Hipotermia: Cuando la temperatura axilar es inferior a 36 ºC.
- Febrícula: Cuando la temperatura axilar se encuentra entre 37.0 °C y 37.5 °C.
- Fiebre: Cuando la temperatura axilar se encuentra entre 37.5 °C y 41 °C.
- Hiperpirexia: Cuando la temperatura axilar es igual o mayor que 41 °C.

La medicina no es una ciencia exacta así que posiblemente esta definición es válida. Pero en informática es una ambiguedad inaceptable, porque si la temperatura es exactamente 37.5 ºC no sabemos si es febrícula, fiebre o ambas. Y si la temperatura es exactamente 41 ºC , también hay ambigüedad entre fiebre e hiperpirexia.

Escribe un programa llamado `~/fpi/practica04/temperatura.pas` que contenga una función que

1. Tenga como argumento una temperatura
2. Devuelva la cadena de texto *hipotermia*, *temperatura normal*, *febrícula*, *fiebre* o *hiperpirexia*, según corresponda. Aplica el criterio definido anteriormente, pero resuelve las ambigüedades (de la manera que creas conveniente)

Observa que si la temperatura es superior a la hipotermia pero inferior a la febrícula, es normal.

El cuerpo principal del programa invocará 5 veces a esta función, con diferentes valores

Práctica 4.2. Case

Escribe un programa llamado `~/fpi/practica04/edad.pas` que sea equivalente al programa `case_en_funcion` de la pg 37 del tema 4, pero empleando sentencias *else if*.

Práctica 4.3. Aeropuertos

Como seguramente sabes, casi todos los aeropuertos del mundo tienen un código de tres letras mayúsculas denominado *código IATA* que se usa, por ejemplo, en las etiquetas del equipaje.

Puedes consultarlos aquí https://en.wikipedia.org/wiki/IATA_airport_code

Elige 5 aeropuertos cualquiera y escribe un programa llamado `~/fpi/practica04/aeropuertos.pas` según la siguiente especificación

1. El programa tendrá una función que reciba como argumento una cadena de texto con un código IATA de aeropuerto, y que devuelva
   - Una cadena de texto con el nombre del aeropuerto, si es uno de los 5 que la función reconoce
   - La cadena .ªeropuerto desconocido.ª otro caso

2. Esta función estará basada en sentencias if encadenadas (else-if)

3. El programa tendrá un cuerpo principal que invocará a la función al menos dos o tres veces, para probarla.

Práctica 4.4. Fase de vuelo

Consideremos que las fases posibles de vuelo de un avión comercial son las siguientes:

- Estacionamiento: velocidad nula.
- Despegue: velocidad no nula, menor de 150 nudos y aceleración positiva
- Ascenso inicial: velocidad entre 150 y 240 nudos y aceleración positiva
- Ascenso final: velocidad mayor de 240, menor o igual a 520 nudos y aceleración positiva
- Crucero: velocidad mayor de 520 nudos (sin importar la aceleración)
- Descenso inicial: velocidad mayor de 300 nudos, menor o igual a 520 nudos, aceleración negativa
- Descenso final: velocidad mayor o igual a 140 nudos, menor o igual a 300 nudos, aceleración negativa
- Aterrizaje: velocidad no nula, menor de 140 nudos y aceleración negativa

Escribe un programa llamado `~/fpi/practica04/fases_vuelo.pas` según la siguiente especificación

1. A partir de la velocidad y la aceleración, indicará la fase de vuelo correspondiente.
2. La velocidad que reciba podrá estar especificada en metros por segundo, en kilómetros por hora o en nudos.
    - La magnitud será un número real. (Las versiones previas de este enunciado decían *número entero*. Si ya lo tenías como entero y funciona bien, puedes dejarlo)
    - La unidad estará expresada con una cadena de texto, que podrá tomar los valores `m/s`, `km/h` o `kn`. (Exactamente así, en minúsculas). Si la unidad es incorrecta, el programa mostrará un mensaje de error y no la fase de vuelo.
3. Internamente, el programa trabajará en nudos. Así que en caso de que la unidad de entrada sea distinta, lo primero que hará el programa es convertirlo (observa que esto es el preproceso).
4. La magnitud y las unidades de la aceleración no son relevantes, así que estará expresada con un booleano que valdrá *TRUE* si es positiva y *FALSE* si es negativa o nula. En otras palabras: habrá una constante booleana con un nombre similar a *acelaracion_positiva*
5. El programa no leerá nada del teclado: usa constantes para los valores de entrada.

Fundamentos de la programación y la informática
Grados en ingeniería aeroespacial, turno de tarde, 2022-2023
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Práctica 5.1. Precondiciones en el programa múltiplos

Copia tu práctica `~/fpi/practica03/multiplos.pas` en `~/fpi/practica05/multiplos.pas`. En este último fichero:

- Añade funciones que comprueben las precondiciones. Usa también un procedimiento: si las precondiciones se cumplen, se invocará a la función y se mostrará el resultado en pantalla. En caso contrario, el procedimiento mostrará un error y no invocará a tu función principal.

Práctica 5.2. Variables y procedimientos

Escribe un programa con el nombre `~/fpi/practica05/procedimientos.pas` acorde con la siguiente especificación. Recuerda sincronizar siempre con FreeFileSync tu cuenta del laboratorio con tu ordenador de casa. Los enunciados no lo dirán explícitamente, pero debes hacerlo.

El programa resolverá algún problema sencillo de física elemental, geometría, matemáticas, etc. Esto es, un problema que consista en devolver un valor a partir de uno o dos parámetros de entrada y aplicar directamente una *fórmula* (una expresión matemática sencilla). Elige la fórmula que prefieras de tus estudios actuales o de tus estudios de secundaria / bachillerato.

1. El valor o valores de entrada serán números reales leidos desde el teclado.
2. Una (o varias) funciones harán el cálculo principal.
3. Toda la escritura en pantalla la hará un procedimiento (no el cuerpo del programa principal).
4. El cuerpo del programa principal hará muy pocas cosas: solo llamar a las funciones y procedimientos que corresponda.

Tu programa solicitará valores numéricos al usuario. Si a pesar de ello el usuario introduce otro tipo de datos, el programa generará un error de ejecución. (Este problema lo corregiremos más adelante)

Práctica 5.3. Variables y procedimientos, mal hecho

Copia el programa `~/fpi/practica05/procedimientos.pas` en `~/fpi/practica05/procedimientos_mal.pas`. Edita este último fichero según la siguiente especificación. Ten cuidado de no modificar el fichero original.

Esta práctica es peculiar, ahora vas a *estropear* el ejercicio anterior. Haz que siga compilando sin errores y devolviendo los mismos valores que antes, pero con un mal diseño: usa las funciones o procedimientos de forma incorrecta. Haz alguna otra cosa mal: usa mal las constantes, pon identificadres inadecuados... cualquier aspecto que hayamos tratado en la asignatura. Deja **muy claro** en el código fuente, qué está mal y explica brevemente por qué.

Práctica 5.4. Filtrado de datos de entrada

Escribe un programa llamado `~/fpi/practica05/filtrado.pas`. En principio será muy similar al que hiciste en el ejercicio 5.2: pedirá datos al usuario y calculará un valor matemático sencillo. El valor a devolver tiene que depender de uno y solo un parámetro. Esto es, el programa leerá un solo valor. (Usa una fórmula distinta a la del ejercicio anterior). El programa:

- Pedirá al usuario que escriba un número (real o entero, según corresponda).

- Leerá no un número, sino una cadena.
- Intentará convertir la cadena en número. Si tiene éxito, invocará a la función que hace los cálculos y mostrará el resultado. En otro caso, mostrará un error describiendo lo sucedido.

Práctica 5.5. Procedimiento de filtrado de entrada

Este ejercicio hará algo similar a lo del ejercicio anterior, pero con mejor diseño. En prácticas anteriores, habías hecho subprogramas (funciones o procedimientos) sencillos que *funcionaban siempre*. Subprogramas que *nunca tienen problemas*. Por ejemplo una función que sume dos reales, siempre podrá devolver un valor. No hay motivo para que *no haya funcionado*

Pero en programación es normal que haya subprogramas que no *sean capaces* de devolver lo pedido. Por ejemplo, un subprograma al que *le pidamos* que devuelva un número entero escrito por el usuario, pero no pueda porque el usuario haya tecleado un número real o incluso una cadena. O una función a la que le pidamos que nos devuelva un número real con la raiz cuadrada de un número, y no pueda porque lo que le pasamos sea un número negativo.

Así que este tipo de subprogramas, devolverán típicamente dos (o más cosas)

- Un booleano, que si vale *true* significa *todo bien, aquí va lo que me has pedido*. Si vale *false* significa *algo ha fallado, no puedo darte lo que querías* [1].
- Otro valor (o tal vez más de uno), donde esté la información que el programador quería. Por ejemplo un número real, o una cadena, o dos números reales, etc.

En caso de que el booleano sea cierto, este último valor tendrá sentido y podrá usarse. Pero si el booleano es falso, el segundo valor no tiene nada, no debe tenerse en cuenta.

Ahora escribirás un subprograma con esta estructura. Escribe un programa llamado `~/fpi/practica05/filtrado02.p` que tenga un procedimiento llamado `intenta_leer_real` (o `try_read_real` si prefieres escribir el programa en inglés). Este procedimiento tendrá:

- El parámetro de entrada `pregunta` o (`question`, en inglés). Contendrá una cadena, con la pregunta que le haremos al usuario
- El parámetro de salida *ok*, que será un booleano indicando si todo ha ido bien y el usuario realmente escribió un real. (Recuerda que para que un parámetro sea de salida, debes pasarlo por referencia, no por copia)
- El parámetro de salida *valor* (o *value*) , que será el real que ha introducido el usuario (si todo ha ido bien) o un valor indeterminado en otro caso.

De esta forma, si necesitamos que el usuario escriba una masa y una aceleración, invocariamos llamadas como esta:

```
intenta_leer_real('Introduce la masa', ok1, masa)
intenta_leer_real('Introduce la aceleración', ok2, aceleracion)
```

Si tanto ok1 como ok2 valen TRUE, el programa calculará la fuerza correspondiente. En otro caso, mostrará un error.

Tu programa hará esto mismo, pero busca un ejemplo distinto al de calcular la fuerza a partir de la masa y la aceleración. Usa si quieres algún ejemplo de alguna de tus prácticas anteriores (si es que tenían dos o más parámetros, no solamente uno)

[1] El procedimiento val hace lo mismo pero no con un booleano sino con un entero

Fundamentos de la programación y la informática
Grados en ingeniería aeroespacial. Turno de tarde. 2022-2023
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Práctica 6.1. Uso básico de registros

Escribe un programa llamado `~/fpi/practica06/registros.pas` que:

- Defina un tipo registro con varios campos. Por ejemplo nombre, apellidos y dni de una persona. O matrícula, marca y modelo de un coche. O autor, título y editorial de un libro, etc.
- Tenga una función llamada *rellena_ejemplo_01* que devolverá un ejemplo cualquiera de registros de ese tipo. Y una función *rellena_ejemplo_02* similar a la anterior, con otro ejemplo. (No son funciones muy realistas, son demasiado sencillas. Tienen un propósito similar al de un *holamundo*).
- Un procedimiento llamado *escribe_registro* que escribirá el registro en pantalla.

El cuerpo principal de tu programa simplemente llamará a estos procedimientos para rellenar y escribir en pantalla un par de registros.

Práctica 6.2. Subprogramas que devuelven registros

Escribe un programa llamado `~/fpi/practica06/registros02.pas` que tenga una función que devuelva dos valores. Será similar al subprograma `division_entera` de la pg 57 del tema 5, pero:

- Será una función, no un procedimiento.
- Devolverá un registro, cuyos campos serán los valores calculados.

Usa cualquier *fórmula* de cualquier otra asignatura. Un ejemplo sería, a partir de un radio, devolver la longitud de la circunferencia y la superficie del círculo (piensa otro, no uses este). O devolver algo en dos unidades distintas. Por ejemplo una distancia en millas y en metros.

Bucles
Fundamentos de la programación y la informática
Grados en ingeniería aeroespacial, turno de tarde, 2022-2023
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Práctica 7.1. Bucle while 1

Escribe un programa en un fichero llamado `~/fpi/practica07/mientras01.pas` que, usando la sentencia *while*, vaya generando números reales aleatorios entre 0 (incluido) y 1 (excluido), y mostrándolos en pantalla. Este proceso se repetirá mientras el número obtenido sea estrictamente menor que una constante (local) llamada *Objetivo*. Dale por ejemplo el valor 0.9. (En otras palabras: se dentendrá cuando el númer obtenido sea mayor o igual que esta constante). Al finalizar, el programa indicará el número de números que ha sido necesario generar hasta conseguir un valor mayor o igual que la constante.

Práctica 7.2. Bucle while 2

Escribe un programa en un fichero llamado `~/fpi/practica07/mientras02.pas` que, usando *while*, vaya *lanzando dados* y sumando el total de puntos acumulados. El programa seguirá lanzando y sumando mientras la suma de puntos sea inferior a la constante *Objetivo*. El número de caras del dado será la constante *CarasDado*. Ambas constante serán locales al cuerpo del programa principal, dales el valor que prefieras.

Observa que si los puntos acumulados están por debajo del objetivo, pero cerca, tendrás que lanzar un último dado. Que puede provocar que *te pases* y excedas el objetivo. Es normal, es lo que pide el enunciado. Piensa en el juego de *las siete y media*.

Práctica 7.3. Lectura de teclado

Preámbulo

En la praćtica 5 escribiste programas que pedían información al usuario. Si este no seguía las instrucciones correctamente, los programas mostraban un error y finalizaba. Ahora que conoces los bucles, puedes mejorarlo: si el usuario no sigue las instrucciones, el programa repetirá la petición de información, todas las veces que haga falta.

Descripción del funcionamiento

Copia tu práctica `~/fpi/practica05/filtrado02.pas` en un fichero con nombre `~/fpi/practica07/filtrado03.pas`. Modifícalo de forma que haga lo mismo que *filtrado02.pas*, con la diferencia de que si el usuario se equivoca no se limite a mostras un mensaje de error, sino que le siga preguntando, hasta que conteste correctamente.

Instrucciones adicionales

El programa tendrá un procedimiento llamado `lee_real` con un parámetro de entrada `pregunta` y un parámetro de salida llamado `valor`. Si prefieres el inglés, los nombres serán `read_real`, `question` y `value`, respectivamente. Este procedimento llamará a `intenta_leer_real`, las veces necesarias hasta que el usuario escriba un valor correcto. Observa que la pregunta que el programador incluya en el procedimiento `lee_real` será la misma que el programa pasará a `intenta_leer_real`.

Por tanto, lo usarás de manera simila a esta:

```
lee_real("Introduce la masa", masa);
lee_real("Introduce la aceleración", aceleracion);
```

Fundamentos de la programación y la informática
Grados en ingeniería aeroespacial, turno de tarde, 2022-2023
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Práctica 8.1. Uso básico de un Array

Escribe un programa en el fichero `~/fpi/practica08/tabla.pas` que:

1. Genere `N` números reales aleatorios mayores o iguales que 0 y menores que `Valor_maximo`, donde `N` será una constante global y `Valor_maximo` será una constantes local del cuerpo del programa principal. Dale a estas constantes el valor que quieras.
2. Guarde los resultados en un array.
3. Muestre todos los valores almacenados en el array.
4. Muestre todos los valores del array, en orden inverso.
5. Calcule (y muestre) la suma y la media de los valores del array.
6. Calcule (y muestre) el máximo y el mínimo de los valores del array.
7. Muestre solamente los valores del array mayores o iguales que $K * valor\_maximo$, donde K será un número real mayor que 0 y menor o igual que 1.

   Ejemplo: si $K = 0{,}9$ y el máximo es 1000, se mostrarán los valores mayores o iguales a 900.
8. K será una constante local del cuerpo del programa principal. Si no se cumple la precondición de ser mayor que 0 y menor o igual a 1, el programa mostrará un mensaje y finalizará.

Práctica 8.2. Búsqueda de matriz aleatoria

Escribe un programa en el fichero `~/fpi/practica08/busca_matriz.pas` que:

- Genere matrices de números enteros, donde cada elemento sea el resultado de tirar un dado. El número de caras del dado será igual al número de elementos en la matriz (número de filas por número de columnas)
- Calcule la esperanza matemática de ese dado (`https://es.wikipedia.org/wiki/Esperanza_matem%C3%A1tica`).
- Genere todas las matrices necesarias hasta conseguir una tal que la media aritmética de sus valores coincida con la esperanza matemática del dado empleado.
- Una vez encontrada, el programa indicará cuantos intentos fueron necesarios.

Por ejemplo, una ejecución podría ser algo similar a esto:

```
Esperanza matemática de un dado de 6 caras: 3.50
Buscando una matriz de 2x3 cuyo valor promedio coincida con la esperaza matemática
6 4 5
4 2 4
Media obtenida: 4.17
4 4 4
6 2 2
Media obtenida: 3.67
4 4 4
1 3 3
Media obtenida: 3.17
6 3 2
```

```
2 2 1
Media obtenida: 2.67
1 6 4
3 3 5
Media obtenida: 3.67
5 3 2
3 2 4
Media obtenida: 3.17
5 3 1
4 2 6
Media obtenida: 3.50
Conseguido en 7 intentos
```

Práctica 8.3. Valores únicos

Escribe un programa en el fichero `~/fpi/practica08/unicos.pas` que rellene una matriz con valores aleatorios de un dado, de forma que no haya ningún valor repetido. El programa comprobará la precondición de que el número de caras sea mayor o igual que el número de elementos en la matriz, y mostrará un error o hará el cálculo, según corresponda.

Una forma muy sencilla de resolver el problema es:

- Inicialmente rellenar la matriz con valores *vacíos* (En este caso, cero, que es un valor imposible para un dado).
- Para cada posición, tirar un dado las veces necesarias hasta que salga un valor que no esté en ninguna posición de la matriz.

Aclaración adicional:

- Empieza escribiendo una función que indique si cierto número existe o no en la matriz. Pruébala por separado. Cuando parezca funcionar correctamente, escribe el procedimiento que, para cada posición, genere números hasta encontrar uno que sea nuevo en la matriz.

Práctica OPTATIVA 8.4. Valores únicos, algoritmo mejorado

Si deseas mejorar tu nota, escribe un programa en el fichero `~/fpi/practica08/unicos2.pas` que resuelva el problema del programa anterior con un algoritmo más eficiente: Cada vez que lance un dado, en vez de comprobar si existe el valor en cualquier posición de la matriz, haz que sea buscado solamente en las posiciones que ya están rellenas. Observa que en este caso no hace falta iniciar la matriz a cero.

Práctica 8.5. Sustitución de espacios

Escribe un programa llamado `~/fpi/practica08/quitaespacios.pas` que tenga una función que reciba una cadena, y que devuelva esa misma cadena, pero reemplazando los espacios por barras bajas (_)

Práctica 8.6. Filtrado de dígitos

Escribe un programa llamado `~/fpi/practica08/digitos.pas` que tenga una función que reciba una cadena, y que devuelva otra cadena que contenga los dígitos, y solo los dígitos, de la cadena inicial.

Ejemplo: para la cadena de entrada *23jl 12* deberá devolver *2312*. Si no hay ningún digito en la cadena inicial, no hará nada especial, esto es, devolverá una cadena vacía.

Práctica 8.7. Repetición de letras

Escribe un programa llamado `~/fpi/practica08/repiteletras.pas` que tenga una función que

- Reciba una cadena y un numero entero n.
- Devuelta otra cadena, con el mismo contenido, pero repitiendo cada letra n veces.

Ejemplo, si recibe *hola mundo* y 3, debe devolver *hhhoooolllaaa mmmuuunnndddooo*

Fundamentos de la programación y la informática
Grados en ingeniería aeroespacial 2022-2023
Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

Práctica 9.1. Escritura de un fichero

Escribe un programa llamado `~/fpi/practica09/aleatorios.pas` que genere una matriz de números aleatorios y la escriba en un fichero `~/fpi/practica09/aleatorios.txt`. Los números pueden ser del tipo que quieras, en el rango que prefieras.

Observa que la matriz la escribirá en el fichero y solo en el fichero, no en pantalla. Lo único que mostrará el programa en pantalla será un mensaje similar a este: *Salida generada en el fichero NOMBRE_DEL_FICHERO*. Reemplazando *NOMBRE_DE_FICHERO* por el valor adecuado, esto es, por el contenido de la constante o la variable correspondiente. Observa que esta sustitución debe hacerla el programa, no escribas en el código fuente el nombre del fichero directamente.

Recuerda que en Linux respresentamos el directorio *home* con la virgulilla, pero esto no está soportado en Pascal.

Práctica 9.2. Lectura de un fichero

Usando el editor de texto, prepara un fichero llamado `~/fpi/practica09/datos.txt` que contenga varias líneas, y en cada línea, un número real. Escribe un programa llamado `~/fpi/practica09/lectura.pas` que lea el fichero anterior, indique cuántas líneas tiene y escriba en pantalla la suma de todos los valores.

Observa que aunque el fichero contiene números, cuando se leen se consideran cadenas. Así que tendrás que convertir estas cadenas en números, usando el procedimiento *val*.

Práctica 9.3. Escritura de un fichero (II)

Copia tu práctica `~/fpi/practica07/mientras02.pas` en un fichero llamado `~/fpi/practica09/escritura.pas`. Modifícalo para que toda la salida del programa se escriba no en pantalla sino en un fichero llamado `~/fpi/practica09/escritura.txt`. Lo único que mostrará el programa en pantalla será un mensaje similar a este: *Salida generada en el fichero NOMBRE_DEL_FICHERO*. Reemplazando *NOMBRE_DE_FICHERO* por el valor adecuado, esto es, por el contenido de la constante o la variable correspondiente, al igual que en la praćtica 9.1.

Revisión de los nombres de los ficheros

Ejecuta `~mortuno/revisa practicas fpi` para comprobar que los nombres de los programas son los correctos.