Fundamentos de la programaci´on y la inform´atica Grados en ingenier´ıa aeroespacial, 2022-2023

[Escuela T´ecnica Superior de Ingenier´ıa de Telecomunicaci´on](http://www.urjc.es/universidad/facultades/escuela-tecnica-superior-de-ingenieria-de-las-telecomunicaciones)

[Universidad Rey Juan Carlo](https://www.urjc.es/)s

# Pr´actica 1.1. Uso b´asico de la shell

En este ejercicio probar´as las ´ordenes b´asicas 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 sesi´on en una m´aquina del laboratorio y

1. Observa el *prompt* y f´ıjate 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

F´ıjate en el uso de mayu´sculas y minu´sculas

1. Entra en cada uno de ellos y haz un listado de su contenido. Observa c´omo va cambiando el *prompt*
2. Crea los directorios ~/fpi/dir10/dir11, ~/fpi/dir10/dir12, ~/fpi/Dir20/dir21 y ~/fpi/dir30/dir31
3. Comprueba con tree que la estructura es la correcta
4. Borra el directorio ~/fpi/Dir20
5. Crea los directorios ~/fpi/dir20 y ~/fpi/dir20/dir21 (Observa que lo que cambia son las mayu´sculas. Aqu´ı recomendamos evitar mayu´sculas en los nombres de ficheros)
6. Comprueba con tree que la estructura es la correcta

# Pr´actica 1.2. Uso b´asico de nano

Redacta un pequen˜o documento llamado ~/fpi/ejemplo.txt enumerando tres o cuatro asigna- turas de las que te hayas matriculado este an˜o, describiendo de forma informal de qu´e tratan. Se trata solo de que pruebes el editor, no importa mucho lo que escribas.

Configura nano para que te indique la l´ınea y columna en la que est´a el cursor.

# Pr´actica 1.3. Hola Mundo

Ahora te ejercitar´as sobre la compilaci´on y ejecuci´on de ficheros, as´ı 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 pr´acticas. Esto incluye mayu´sculas y minu´sculas (usaremos siempre minu´scula). La recogida ser´a autom´atica, un error en una letra supondr´a un no presentado (aunque tendr´as un script para revisar que no hay errores en los nombres)

1. Comp´ılalo.
2. Ejecu´talo.
3. Escribe en tu cuenta un fichero con el nombre ~/fpi/practica01/nota\_fpi.pas cuyo contenido sea el fichero que encontrar´as en [https://gsyc.urjc.es/˜mortuno/nota fpi.pas](https://gsyc.urjc.es/~mortuno/nota_fpi.pas)

Si copias y pegas directamente con el rat´on en nano tal vez se pierda la tabulaci´on.

Para evitarlo, ejecuta en el terminal

wget https://gsyc.urjc.es/~mortuno/nota\_fpi.pas

La orden *wget* sirve para descargar una p´agina web: se le pasa como argumento la direcci´on web de un fichero, lo descarga y lo escribe en el directorio actual.

1. Comp´ılalo y ejecu´talo.
2. Ahora an˜ade al *holamundo* las directivas del compilador. Podr´ıas escribirlas a mano, pero mejor c´opialas desde el fichero nota\_fpi.pas. (La l´ınea con las directivas del compilador ser´a siempre la misma en todos los programas que haremos durante el curso)
3. Compila y ejecuta.

# Pr´actica 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, escr´ıbelo, y en vez de mantener el nombre original como te ofrece el editor por omisi´on, elige un nombre nuevo. As´ı habr´as copiado el fichero.

1. Modifica estos dos ficheros para que tengan algu´n 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.

Pru´ebalo con error, comprueba que falla. Comenta el error, comprueba que ya no falla nada m´as. Explica en un comentario d´onde est´a el error. Luego puedes dejar la versi´on con el error o la versi´on con el error inhabilitado, como prefieras.

# Pr´actica 1.5. Modificacio´n de un programa

1. Vuelve a copiar el programa anterior, esta vez con nombre ~/fpi/practica01/nota\_fpi\_04.pas

Modif´ıcalo 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.

# Pr´actica 1.6. Errores l´ogicos

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

Modif´ıcalos para que generen cada uno un error l´ogico distinto (y solo un error l´ogico). An˜ade un comentario en cada programa indicando d´onde est´a el error.

Fundamentos de la programaci´on y la inform´atica

Grados en ingenier´ıa aeroespacial, turno de tarde 2022-2023 [Escuela T´ecnica Superior de Ingenier´ıa de Telecomunicaci´on](http://www.urjc.es/universidad/facultades/escuela-tecnica-superior-de-ingenieria-de-las-telecomunicaciones)

[Universidad Rey Juan Carlo](https://www.urjc.es/)s

# Pr´actica 2.1. Casting

En este ejercicio probar´as el *casting* de tipos de datos

1. Crea el directorio

~/fpi/practica02

1. Configura tu editor para que muestre los caracteres invisibles. Observa c´omo representa los tabu- ladores y c´omo representa los espacios. Presta mucha atenci´on al sangrado de tu programa, en este ejercicio y en todos los que escribas durante el resto del curso.
2. Escribe un programa llamado ~/fpi/practica02/ahormados.pas

Ser´a parecido al programa *casting* del tema 2, pero no id´entico.

1. El programa debe hacer al menos 4 conversiones de tipos de dato correctas.
2. Escribe tambi´en un par de ejemplos incorrectos. Una vez que compruebes que dan error de compi- laci´on, *comenta* esas l´ıneas.

(Comentar una l´ınea es una expresi´on habitual en programaci´on. No significa que hagas un comen- tario explicando la l´ınea, sino que deshabilites la l´ınea convirti´endola en un comentario. No tendr´a efecto, pero seguir´a presente en el c´odigo para que el programador pueda verla)

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

# Pr´actica 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 c´entimos del IVA. Pero aqu´ı lo igno- raremos.

Observa que este programa ser´a 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, saldr´a 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(

# Pr´actica 2.3. Sustentaci´on

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

Emplea la notaci´on de esta [p´agina de wikipedia.](https://es.wikipedia.org/wiki/Sustentaci%C3%B3n)

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

Fundamentos de la programaci´on y la inform´atica

Grados en ingenier´ıa aeroespacial, turno de tarde, 2022-2023 [Escuela T´ecnica Superior de Ingenier´ıa de Telecomunicaci´on](http://www.urjc.es/universidad/facultades/escuela-tecnica-superior-de-ingenieria-de-las-telecomunicaciones)

[Universidad Rey Juan Carlo](https://www.urjc.es/)s

# Pr´actica 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.

# Pr´actica 3.2. Sustentaci´on (con funciones)

Escribe un programa llamado ~/fpi/practica03/sustentacion.pas que haga lo mismo que el pro- grama del ejercicio 2.5, con la misma f´ormula, pero empleando funciones.

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

# Pr´actica 3.3. Mu´ltiplos

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

Dados tres caracteres, C1, C2 y C3, que se corresponden los tres d´ıgitos de un nu´mero (en base 10), escriba si el nu´mero resultante es mu´ltiplo de 5 o no.

Por ejemplo, dados los caracteres ’0’ ’4’ ’7’, el programa ha de escribir FALSE, dado que 47 no es mu´ltiplo de 5.

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

Pautas:

Convierte los 3 caracteres en 3 d´ıgitos, los 3 d´ıgitos en un nu´mero y comprueba si el m´odulo de la divisi´on entera del nu´mero entre 5 es 0.

Para convertir un caracter en un d´ıgito, toma como punto de partida la funci´on *integer* que, a partir de un car´acter, devuelve el nu´mero entero correspondiente en la tabla ASCII. Por ejemplo del caracter 7 el c´odigo ASCII es 55. R´estale el c´odigo ASCII del car´acter 0 y tendr´as el nu´mero que necesitas. En este caso, 7.

Para convertir tres d´ıgitos en un nu´mero observa el siguiente ejemplo. 496 = 4 *×* 100 + 9 *×* 10 + 6

# Pr´actica 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 incluir´a una funci´on que calcule la velocidad de despegue de un avi´on gen´erico, dados los siguientes datos: densidad del aire, masa, coeficiente de sustentaci´on y superficie alar del avi´on. Recuerda que la velocidad de despegue es la velocidad cuando la sustentaci´on es igual al peso.

Las velocidades deber´an obtenerse tanto en m/s como en Km/h.

Utiliza la f´ormula de la sustentaci´on empleada en la pr´actica anterior, despejando la velocidad. Utiliza en la f´ormula la misma notaci´on que en la pr´actica 2.5.

Usa solamente funciones, constantes y expresiones. Ni variables ni sentencias if-then-else. Datos:

Densidad del aire: 1.225 kg por metro cu´bico. Airbus A380:

* Masa: 560 toneladas
* Superficie alar: 845 metros cuadrados
* Coeficiente de sustentaci´on: 2.4

Airbus A320:

* Masa: 78 toneladas
* Superficie alar: 122 metros cuadrados
* Coeficiente de sustentaci´on: 2.3

Fundamentos de la programaci´on y la inform´atica

Grados en ingenier´ıa aeroespacial, turno de tarde, 2022-2023 [Escuela T´ecnica Superior de Ingenier´ıa de Telecomunicaci´on](http://www.urjc.es/universidad/facultades/escuela-tecnica-superior-de-ingenieria-de-las-telecomunicaciones)

[Universidad Rey Juan Carlo](https://www.urjc.es/)s

## Cambios en el documento

Pr´actica 4.4. Donde dec´ıa *la magnitud ser´a un nu´mero entero* ahora dice *la magnitud ser´a un nu´mero real*

# Pr´actica 4.1. Temperatura

En wikipedia podemos leer lo siguiente:

Hipotermia: Cuando la temperatura axilar es inferior a 36 -C.

Febr´ıcula: Cuando la temperatura axilar se encuentra entre 37.0 oC y 37.5 oC. Fiebre: Cuando la temperatura axilar se encuentra entre 37.5 oC y 41 oC. Hiperpirexia: Cuando la temperatura axilar es igual o mayor que 41 oC.

La medicina no es una ciencia exacta as´ı que posiblemente esta definici´on es v´alida. Pero en inform´ati- ca es una ambiguedad inaceptable, porque si la temperatura es exactamente 37.5 -C no sabemos si es febr´ıcula, fiebre o ambas. Y si la temperatura es exactamente 41 -C , tambi´en hay ambigu¨edad entre fiebre e hiperpirexia.

Escribe un programa llamado ~/fpi/practica04/temperatura.pas que contenga una funci´on que

1. Tenga como argumento una temperatura
2. Devuelva la cadena de texto *hipotermia*, *temperatura normal*, *febr´ıcula*, *fiebre* o *hiperpirexia*, segu´n corresponda. Aplica el criterio definido anteriormente, pero resuelve las ambigu¨edades (de la manera que creas conveniente)

Observa que si la temperatura es superior a la hipotermia pero inferior a la febr´ıcula, es normal. El cuerpo principal del programa invocar´a 5 veces a esta funci´on, con diferentes valores

# Pr´actica 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*.

# Pr´actica 4.3. Aeropuertos

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

Puedes consultarlos aqu´ı [https://en.wikipedia.org/wiki/IATA airport code](https://en.wikipedia.org/wiki/IATA_airport_code)

Elige 5 aeropuertos cualquiera y escribe un programa llamado ~/fpi/practica04/aeropuertos.pas

segu´n la siguiente especificaci´on

1. El programa tendr´a una funci´on que reciba como argumento una cadena de texto con un c´odigo IATA de aeropuerto, y que devuelva

Una cadena de texto con el nombre del aeropuerto, si es uno de los 5 que la funci´on reconoce La cadena .Aeropuerto desconocido.en otro caso

1. Esta funci´on estar´a basada en sentencias if encadenadas (else-if)
2. El programa tendr´a un cuerpo principal que invocar´a a la funci´on al menos dos o tres veces, para probarla.

# Pr´actica 4.4. Fase de vuelo

Consideremos que las fases posibles de vuelo de un avi´on comercial son las siguientes: Estacionamiento: velocidad nula.

Despegue: velocidad no nula, menor de 150 nudos y aceleraci´on positiva Ascenso inicial: velocidad entre 150 y 240 nudos y aceleraci´on positiva

Ascenso final: velocidad mayor de 240, menor o igual a 520 nudos y aceleraci´on positiva Crucero: velocidad mayor de 520 nudos (sin importar la aceleraci´on)

Descenso inicial: velocidad mayor de 300 nudos, menor o igual a 520 nudos, aceleraci´on negativa Descenso final: velocidad mayor o igual a 140 nudos, menor o igual a 300 nudos, aceleraci´on negativa Aterrizaje: velocidad no nula, menor de 140 nudos y aceleraci´on negativa

Escribe un programa llamado ~/fpi/practica04/fases\_vuelo.pas segu´n la siguiente especificaci´on

1. A partir de la velocidad y la aceleraci´on, indicar´a la fase de vuelo correspondiente.
2. La velocidad que reciba podr´a estar especificada en metros por segundo, en kil´ometros por hora o en nudos.

La magnitud ser´a un nu´mero real. (Las versiones previas de este enunciado dec´ıan *nu´mero entero*. Si ya lo ten´ıas como entero y funciona bien, puedes dejarlo)

La unidad estar´a expresada con una cadena de texto, que podr´a tomar los valores m/s, km/h o kn. (Exactamente as´ı, en minu´sculas). Si la unidad es incorrecta, el programa mostrar´a un mensaje de error y no la fase de vuelo.

1. Internamente, el programa trabajar´a en nudos. As´ı que en caso de que la unidad de entrada sea distinta, lo primero que har´a el programa es convertirlo (observa que esto es el preproceso).
2. La magnitud y las unidades de la aceleraci´on no son relevantes, as´ı que estar´a expresada con un booleano que valdr´a *TRUE* si es positiva y *FALSE* si es negativa o nula. En otras palabras: habr´a una constante booleana con un nombre similar a *acelaracion positiva*
3. El programa no leer´a nada del teclado: usa constantes para los valores de entrada.

Fundamentos de la programaci´on y la inform´atica

Grados en ingenier´ıa aeroespacial, turno de tarde, 2022-2023 [Escuela T´ecnica Superior de Ingenier´ıa de Telecomunicaci´on](http://www.urjc.es/universidad/facultades/escuela-tecnica-superior-de-ingenieria-de-las-telecomunicaciones)

[Universidad Rey Juan Carlo](https://www.urjc.es/)s

# Pr´actica 5.1. Precondiciones en el programa mu´ltiplos

Copia tu pr´actica ~/fpi/practica03/multiplos.pas en ~/fpi/practica05/multiplos.pas. En este u´ltimo fichero:

An˜ade funciones que comprueben las precondiciones. Usa tambi´en un procedimiento: si las pre- condiciones se cumplen, se invocar´a a la funci´on y se mostrar´a el resultado en pantalla. En caso contrario, el procedimiento mostrar´a un error y no invocar´a a tu funci´on principal.

# Pr´actica 5.2. Variables y procedimientos

Escribe un programa con el nombre ~/fpi/practica05/procedimientos.pas acorde con la siguiente especificaci´on. Recuerda sincronizar siempre con FreeFileSync tu cuenta del laboratorio con tu ordenador de casa. Los enunciados no lo dir´an expl´ıcitamente, pero debes hacerlo.

El programa resolver´a algu´n problema sencillo de f´ısica elemental, geometr´ıa, matem´aticas, etc. Esto es, un problema que consista en devolver un valor a partir de uno o dos par´ametros de entrada y aplicar directamente una *f´ormula* (una expresi´on matem´atica sencilla). Elige la f´ormula que prefieras de tus estudios actuales o de tus estudios de secundaria / bachillerato.

1. El valor o valores de entrada ser´an nu´meros reales leidos desde el teclado.
2. Una (o varias) funciones har´an el c´alculo principal.
3. Toda la escritura en pantalla la har´a un procedimiento (no el cuerpo del programa principal).
4. El cuerpo del programa principal har´a muy pocas cosas: solo llamar a las funciones y procedimientos que corresponda.

Tu programa solicitar´a valores num´ericos al usuario. Si a pesar de ello el usuario introduce otro tipo de datos, el programa generar´a un error de ejecuci´on. (Este problema lo corregiremos m´as adelante)

# Pr´actica 5.3. Variables y procedimientos, mal hecho

Copia el programa ~/fpi/practica05/procedimientos.pas en ~/fpi/practica05/procedimientos\_mal.pas.

Edita este u´ltimo fichero segu´n la siguiente especificaci´on. Ten cuidado de no modificar el fichero original. Esta pr´actica 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 disen˜o: usa las funciones o pro- cedimientos 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 c´odigo

fuente, qu´e est´a mal y explica brevemente por qu´e.

# Pr´actica 5.4. Filtrado de datos de entrada

Escribe un programa llamado ~/fpi/practica05/filtrado.pas. En principio ser´a muy similar al que hiciste en el ejercicio 5.2: pedir´a datos al usuario y calcular´a un valor matem´atico sencillo. El valor a devolver tiene que depender de uno y solo un par´ametro. Esto es, el programa leer´a un solo valor. (Usa una f´ormula distinta a la del ejercicio anterior). El programa:

Pedir´a al usuario que escriba un nu´mero (real o entero, segu´n corresponda).

Leer´a no un nu´mero, sino una cadena.

Intentar´a convertir la cadena en nu´mero. Si tiene ´exito, invocar´a a la funci´on que hace los c´alculos y mostrar´a el resultado. En otro caso, mostrar´a un error describiendo lo sucedido.

# Pr´actica 5.5. Procedimiento de filtrado de entrada

Este ejercicio har´a algo similar a lo del ejercicio anterior, pero con mejor disen˜o. En pr´acticas an- teriores, hab´ıas hecho subprogramas (funciones o procedimientos) sencillos que *funcionaban siempre*. Subprogramas que *nunca tienen problemas*. Por ejemplo una funci´on que sume dos reales, siempre podr´a devolver un valor. No hay motivo para que *no haya funcionado*

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

As´ı que este tipo de subprogramas, devolver´an t´ıpicamente dos (o m´as cosas)

Un booleano, que si vale *true* significa *todo bien, aqu´ı va lo que me has pedido*. Si vale *false* significa

*algo ha fallado, no puedo darte lo que quer´ıas* [1](#_bookmark0).

Otro valor (o tal vez m´as de uno), donde est´e la informaci´on que el programador quer´ıa. Por ejemplo un nu´mero real, o una cadena, o dos nu´meros reales, etc.

En caso de que el booleano sea cierto, este u´ltimo valor tendr´a sentido y podr´a usarse. Pero si el booleano es falso, el segundo valor no tiene nada, no debe tenerse en cuenta.

Ahora escribir´as 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 pro-

grama en ingl´es). Este procedimiento tendr´a:

El par´ametro de entrada pregunta o (question, en ingl´es). Contendr´a una cadena, con la pregunta que le haremos al usuario

El par´ametro de salida *ok*, que ser´a un booleano indicando si todo ha ido bien y el usuario realmente escribi´o un real. (Recuerda que para que un par´ametro sea de salida, debes pasarlo por referencia, no por copia)

El par´ametro de salida *valor* (o *value*) , que ser´a 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 aceleraci´on, invocariamos lla- madas como esta:

intenta\_leer\_real(’Introduce la masa’, ok1, masa) intenta\_leer\_real(’Introduce la aceleraci´on’, ok2, aceleracion)

Si tanto ok1 como ok2 valen TRUE, el programa calcular´a la fuerza correspondiente. En otro caso, mostrar´a un error.

Tu programa har´a esto mismo, pero busca un ejemplo distinto al de calcular la fuerza a partir de la masa y la aceleraci´on. Usa si quieres algu´n ejemplo de alguna de tus pr´acticas anteriores (si es que ten´ıan dos o m´as par´ametros, no solamente uno)

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

Fundamentos de la programaci´on y la inform´atica

Grados en ingenier´ıa aeroespacial. Turno de tarde. 2022-2023 [Escuela T´ecnica Superior de Ingenier´ıa de Telecomunicaci´on](http://www.urjc.es/universidad/facultades/escuela-tecnica-superior-de-ingenieria-de-las-telecomunicaciones)

[Universidad Rey Juan Carlo](https://www.urjc.es/)s

# Pr´actica 6.1. Uso b´asico de registros

Escribre 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 matr´ıcula, marca y modelo de un coche. O autor, t´ıtulo y editorial de un libro, etc.

Tenga una funci´on llamada *rellena ejemplo 01* que devolver´a un ejemplo cualquiera de registros de ese tipo. Y una funci´on *rellena ejemplo 02* similar a la anterior, con otro ejemplo. (No son funciones muy realistas, son demasiado sencillas. Tienen un prop´osito similar al de un *holamundo*).

Un procedimiento llamado *escribe registro* que escribir´a el registro en pantalla.

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

# Pr´actica 6.2. Subprogramas que devuelven registros

Escribre un programa llamado ~/fpi/practica06/registros02.pas que tenga una funci´on que devuelva dos valores. Ser´a similar al subprograma division\_entera de la pg 57 del tema 5, pero:

Ser´a una funci´on, no un procedimiento.

Devolver´a un registro, cuyos campos ser´an los valores calculados.

Usa cualquier *f´ormula* de cualquier otra asignatura. Un ejemplo ser´ıa, a partir de un radio, devolver la longitud de la circunferencia y la superficie del c´ırculo (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 programaci´on y la inform´atica

Grados en ingenier´ıa aeroespacial, turno de tarde, 2022-2023 [Escuela T´ecnica Superior de Ingenier´ıa de Telecomunicaci´on](http://www.urjc.es/universidad/facultades/escuela-tecnica-superior-de-ingenieria-de-las-telecomunicaciones) [Universidad Rey Juan Carlo](https://www.urjc.es/)s

# Pr´actica 7.1. Bucle while 1

Escribe un programa en un fichero llamado ~/fpi/practica07/mientras01.pas que, usando la sentencia *while*, vaya generando nu´meros reales aleatorios entre

0 (incluido) y 1 (excluido), y mostr´andolos en pantalla. Este proceso se repetir´a mientras el nu´mero obtenido sea estrictamente menor que una constante (local) llamada *Objetivo*. Dale por ejemplo el valor 0.9. (En otras palabras: se dentendr´a cuando el nu´mer obtenido sea mayor o igual que esta constante). Al finalizar, el programa indicar´a el nu´mero de nu´meros que ha sido necesario generar hasta conseguir un valor mayor o igual que la constante.

# Pr´actica 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 acumula- dos. El programa seguir´a lanzando y sumando mientras la suma de puntos sea inferior a la constante *Objetivo*. El nu´mero de caras del dado ser´a la constante *CarasDado*. Ambas constante ser´an locales al cuerpo del programa principal, dales el valor que prefieras.

Observa que si los puntos acumulados est´an por debajo del objetivo, pero cerca, tendr´as que lanzar un u´ltimo 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*.

# Pr´actica 7.3. Lectura de teclado

## Pre´ambulo

En la pra´ctica 5 escribiste programas que ped´ıan informaci´on al usuario. Si este no segu´ıa 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 repetir´a la petici´on de informaci´on, todas las veces que haga falta.

## Descripci´on del funcionamiento

Copia tu pr´actica ~/fpi/practica05/filtrado02.pas en un fichero con nombre ~/fpi/practica07/filtrado03.pas. Modif´ıcalo 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 tendr´a un procedimiento llamado lee\_real con un par´ametro de entrada pregunta y un par´ametro de salida llamado valor. Si prefieres el ingl´es, los nombres ser´an read\_real, question y value, respectivamente. Este procedimento llamar´a 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 ser´a la misma que el programa pasar´a a intenta\_leer\_real.

Por tanto, lo usar´as de manera simila a esta:

lee\_real("Introduce la masa", masa); lee\_real("Introduce la aceleraci´on", aceleracion);

Fundamentos de la programaci´on y la inform´atica

Grados en ingenier´ıa aeroespacial, turno de tarde, 2022-2023 [Escuela T´ecnica Superior de Ingenier´ıa de Telecomunicaci´on](http://www.urjc.es/universidad/facultades/escuela-tecnica-superior-de-ingenieria-de-las-telecomunicaciones)

[Universidad Rey Juan Carlo](https://www.urjc.es/)s

# Pr´actica 8.1. Uso b´asico de un Array

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

1. Genere N nu´meros reales aleatorios mayores o iguales que 0 y menores que Valor\_maximo, donde N ser´a una constante global y Valor\_maximo ser´a 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 m´aximo y el m´ınimo de los valores del array.
7. Muestre solamente los valores del array mayores o iguales que *K valor maximo*, donde *K* ser´a un nu´mero real mayor que 0 y menor o igual que 1.

*∗*

Ejemplo: si *K* = 0*,*9 y el m´aximo es 1000, se mostrar´an los valores mayores o iguales a 900.

1. *K* ser´a una constante local del cuerpo del programa principal. Si no se cumple la precondici´on de ser mayor que 0 y menor o igual a 1, el programa mostrar´a un mensaje y finalizar´a.

# Pr´actica 8.2. Bu´squeda de matriz aleatoria

Escribe un programa en el fichero ~/fpi/practica08/busca\_matriz.pas que:

Genere matrices de nu´meros enteros, donde cada elemento sea el resultado de tirar un dado. El nu´mero de caras del dado ser´a igual al nu´mero de elementos en la matriz (nu´mero de filas por nu´mero de columnas)

Calcule la esperanza matem´atica de ese dado ([https://es.wikipedia.org/wiki/Esperanza\_](https://es.wikipedia.org/wiki/Esperanza_matem%C3%A1tica) [matem%C3%A1tica](https://es.wikipedia.org/wiki/Esperanza_matem%C3%A1tica)).

Genere todas las matrices necesarias hasta conseguir una tal que la media aritm´etica de sus valores coincida con la esperanza matem´atica del dado empleado.

Una vez encontrada, el programa indicar´a cuantos intentos fueron necesarios. Por ejemplo, una ejecuci´on podr´ıa ser algo similar a esto:

Esperanza matem´atica de un dado de 6 caras: 3.50

Buscando una matriz de 2x3 cuyo valor promedio coincida con la esperaza matem´atica 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

# Pr´actica 8.3. Valores u´nicos

Escribe un programa en el fichero ~/fpi/practica08/unicos.pas que rellene una matriz con va- lores aleatorios de un dado, de forma que no haya ningu´n valor repetido. El programa comprobar´a la precondici´on de que el nu´mero de caras sea mayor o igual que el nu´mero de elementos en la matriz, y mostrar´a un error o har´a el c´alculo, segu´n corresponda.

Una forma muy sencilla de resolver el problema es:

Inicialmente rellenar la matriz con valores *vac´ıos* (En este caso, cero, que es un valor imposible para un dado).

Para cada posici´on, tirar un dado las veces necesarias hasta que salga un valor que no est´e en ninguna posici´on de la matriz.

Aclaraci´on adicional:

Empieza escribiendo una funci´on que indique si cierto nu´mero existe o no en la matriz. Pru´ebala por separado. Cuando parezca funcionar correctamente, escribe el procedimiento que, para cada posici´on, genere nu´meros hasta encontrar uno que sea nuevo en la matriz.

# Pr´actica OPTATIVA 8.4. Valores u´nicos, 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 m´as eficiente: Cada vez que lance un dado, en vez de comprobar si existe el valor en cualquier posici´on de la matriz, haz que sea buscado solamente en las posiciones que ya est´an rellenas. Observa que en este caso no hace falta iniciar la matriz a cero.

# Pr´actica 8.5. Sustituci´on de espacios

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

# Pr´actica 8.6. Filtrado de d´ıgitos

Escribe un programa llamado ~/fpi/practica08/digitos.pas que tenga una funci´on que reciba una cadena, y que devuelva otra cadena que contenga los d´ıgitos, y solo los d´ıgitos, de la cadena inicial.

Ejemplo: para la cadena de entrada *23jl 12* deber´a devolver *2312*. Si no hay ningu´n digito en la cadena inicial, no har´a nada especial, esto es, devolver´a una cadena vac´ıa.

# Pr´actica 8.7. Repeticio´n de letras

Escribe un programa llamado ~/fpi/practica08/repiteletras.pas que tenga una funci´on 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 *hhhooolllaaa mmmuuunnndddooo*

Fundamentos de la programaci´on y la inform´atica Grados en ingenier´ıa aeroespacial 2022-2023

[Escuela T´ecnica Superior de Ingenier´ıa de Telecomunicaci´on](https://www.urjc.es/etsit)

[Universidad Rey Juan Carlo](https://www.urjc.es/)s

# Pr´actica 9.1. Escritura de un fichero

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

Observa que la matriz la escribir´a en el fichero y solo en el fichero, no en pantalla. Lo u´nico que mostrar´a el programa en pantalla ser´a un mensaje similar a este: *Salida generada en el fichero NOM- BRE 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 sustituci´on debe hacerla el programa, no escribas en el c´odigo fuente el nombre del fichero directamente.

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

# Pr´actica 9.2. Lectura de un fichero

Usando el editor de texto, prepara un fichero llamado ~/fpi/practica09/datos.txt que contenga varias l´ıneas, y en cada l´ınea, un nu´mero real. Escribe un programa llamado ~/fpi/practica09/lectura.pas que lea el fichero anterior, indique cu´antas l´ıneas tiene y escriba en pantalla la suma de todos los valores.

Observa que aunque el fichero contiene nu´meros, cuando se leen se consideran cadenas. As´ı que tendr´as que convertir estas cadenas en nu´meros, usando el procedimiento *val*.

# Pr´actica 9.3. Escritura de un fichero (II)

Copia tu pr´actica ~/fpi/practica07/mientras02.pas en un fichero llamado ~/fpi/practica09/escritura.pas.

Modif´ıcalo para que toda la salida del programa se escriba no en pantalla sino en un fichero lla-

mado ~/fpi/practica09/escritura.txt. Lo u´nico que mostrar´a el programa en pantalla ser´a un

mensaje similar a este: *Salida generada en el fichero NOMBRE DEL FICHERO*. Reemplazando *NOM- BRE DE FICHERO* por el valor adecuado, esto es, por el contenido de la constante o la variable corres- pondiente, al igual que en la pra´ctica 9.1.

# Revisi´on de los nombres de los ficheros

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

© 2022 Miguel Angel Ortuño Pérez.

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

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