Ejercicios 2

1. Hacer un programa que pida cadenas por teclado hasta que se introduzca una cadena vacía. Para cada cadena se mostrará su tamaño.
2. Hacer un programa que pida una cadena. Luego pedirá números hasta que se introduzca uno negativo, mostrando en cada caso el carácter que ocupa la posición indicada por el número.
3. Arreglamos el ejercicio anterior, para que muestre un mensaje de error cada vez que el número introducido sea mayor que la longitud de la cadena.
4. Hacer un programa que pida una palabra por teclado y la deletree separada por guiones.

   Ejemplo:

   hola -> h - o - l – a

5. Pedir una cadena por teclado y mostrarla invertida. Ejemplo: hola -> aloh
6. Hacer un programa que pida una frase por teclado y diga cuantas vocales hay.
7. Leer una frase de máximo 80 caracteres y escribirla progresivamente. Es decir si la frase es: "ciervo", el programa debe imprimir

   c
   ci
   cie
   cier
   cierv
   ciervo
8. Pedir una frase y mostrar las palabras en una columna. Además, decir cuántas palabras hay. (Sólo habrá un espacio en blanco entre dos palabras)

   Ejemplo:

   Introduce una frase: al ir por el campo, cazando mariposas
   al
   ir
   por
   el
   campo,
   cazando
   mariposas
   En total hay 7 palabras
9. Pedir una frase por teclado y codificarla de la siguiente manera: Se pedirá un número por teclado entre 1 y 5, y a cada carácter de la frase se le sumará dicho número. Por ejemplo, si la frase inicial es PROGRAMACION, y el número de codificación es el 3,el resultado será SURJUDPDFLRQ
10. Crear un método que devuelve si una frase pasada como parámetro es palíndromo o no. En el programa principal se pedirá una frase y se mostrará un mensaje diciendo si es palíndromo o no, continuará pidiendo frases hasta que se pulse enter.

    Se valorará positivamente utilizar la ejecución más óptima. (recorrer la frase el número de veces estrictamente necesario)

11. Igual al anterior, pero los espacios en blanco no se tendrán en cuenta. Ejemplo: se van sus naves --> es palíndromo.
12. Hacer un programa que pida dos nombres y te diga si son iguales. Si son distintos los muestra ordenados. Finalmente muestra los dos nombres en mayúsculas y en minúsculas.
13. Pedir 5 nombres (igual para 500) y decir cuál es el primero de la lista.
14. Inicializar un array con los días de la semana. A continuación se deben pedir números del 1 al 7 y mostrar por pantalla el día de la semana con el que se corresponde el número. El programa finaliza cuando se introduzca un número fuera de ese rango.
15. a) Pedir la nota de 5 alumnos (Igual para 500) y luego mostrarlas.
    b) Después modificar el programa para que al final muestre todas las notas y calcule la nota media.
    c) Modificar el anterior programa, para que después de pedir las 5 notas, muestre la nota media, y las notas que sean superior a la media.
    d) Finalmente, igual que la opción c), pero que muestre la mejor y la peor nota.
    e) Modificarlo de nuevo para que sume 1 punto a todos los alumnos que superen la nota media.
16. Hacer un programa que pida una fecha (día, mes y año) mientras no sea una fecha correcta. En caso de que no sea correcta deberá indicarlo y pedir una nueva fecha. Cuando se introduzca una fecha correcta se indicará y el programa terminará.
17. Diseñe un programa que actúe como calculadora de fracciones. Debe de leer el numerador y denominador de dos fracciones y la operación a realizar con ellas. Se deben ofrecer las cuatro operaciones aritméticas básicas (+, -, *, /). La fracción resultante siempre se debe mostrar simplificada.
18. Escriba un programa que sirva para examinar a un niño de las tablas de multiplicar. Para ello se generan de forma pseudoaleatoria diez preguntas que son planteadas al niño. Ante cada pregunta (por ejemplo “4x5=”) el niño contestará con un número. Si la respuesta es la correcta se le felicita. Si la respuesta es incorrecta se le informará al niño de su error y se le volverá a plantear la misma pregunta hasta que acierte. Después de concluir con la última pregunta se informará al niño sobre cuántas preguntas acertó a la primera.

19. Se debe escribir un programa que saque por pantalla el calendario de un mes cualquiera de cualquier año a partir de 1900, o bien un año completo, o bien indique para una fecha determinada su día de la semana. Para ello se deberá ofrecer un menú con esas cuatro opciones más la opción de salir. El programa debe diseñarse de modo que las entradas desde el teclado sean robustas, solicitando una nueva introducción en caso de que el dato introducido sea erróneo o inadecuado. Dato importante: el 1 de enero de 1900 fue lunes. El programa deberá presentar los meses en un formato similar al siguiente:

    ENERO

    ---

    1 2 3 4 5 6 7

8 9 10 11 12 13 14

15 16 17 18 19 20 21

22 23 24 25 26 27 28

29 30 31

20. Realizar un método para comprobar si un password es aceptable, el password se le pasará como parámetro. Un password es aceptable si su tamaño es mayor que 2 y menor que 10. Además no puede contener ningún espacio en blanco, y uno de los caracteres tiene que ser obligatoriamente uno de estos: #@$%&.

    El método devolverá un byte con la siguiente codificación

    0.- si es correcto

    1.- si es menor de dos caracteres.

    2.- si es mayor de 10 caracteres.

    3.- si contiene espacios en blanco.

    4.- si no contiene uno de los caracteres especiales.

21. Hacer un método al que se le pasará un DNI como parámetro y devolverá un booleano indicando si es correcto o no. Para ser correcto un DNI debe tener una longitud de 9 y el último carácter tiene que cumplir lo siguiente:
    Se divide la parte numérica del DNI para 23, y el resto de esa división se corresponde con la posición de la letra en la siguiente cadena:
    TRWAGMYFPDXBNJZSQVHLCKE
22. Hacer un programa que tenga un array con 10 números enteros entre el 1 y el 99, generados aleatoriamente.
    Pedir números por teclado hasta que se introduzca uno de los 10 números del array.
23. Hacer un programa en Java que pida una frase por teclado y luego te diga cuantas veces aparecen cada una de las vocales. Ej:
    Introduce el texto:
    Hola HOlA
    a e i o u
    2 0 0 2 0
    Nota: utilizar un Array para los contadores de vocales.
24. Hacer un método en Java que reciba como parámetros dos arrays de 10 números enteros y devuelva un array con la suma de cada uno los elementos de los dos arrays. Ejemplo:

a1	5	7	8	4	3	9	7	5	7	4
a2	5	3	9	3	2	6	5	3	8	9
resultado	10	10	17	7	5	15	12	8	15	13

Para probarlo, se crearán los arrays con números aleatorios entre 1 y 10.

25. Vamos a simular una carrera de caballos.Inicialmente se pedirá por teclado cuántos caballos van a correr.Después se irán haciendo sucesivos turnos en los que cada caballo avanzará un número aleatorio entre 0 y 5.Un caballo gana cuando alcanza la puntuación de 50.Al final se mostrará el dorsal del/de los caballos ganadores.
26. Buscar y Reemplazar: Haz un programa en java que pida un texto y luego una palabra a buscar y otra para reemplazarla. Finalmente mostrar el texto con la palabra cambiada. No se puede usar replace ni replaceAll y hay que utilizar indexOf y substring.
27. Hacer un programa en Java con un array bi-dimensional en el que tendremos 10 palabras en español y su traducción en inglés. Fila 0 castellano, Fila 1 inglés.
    El programa pedirá una palabra por teclado y mostrará su traducción en inglés, hasta que se introduzca una palabra vacía (enter). También indicará si la palabra no está en el diccionario.
    Utilizaremos el siguiente método (obligatoriamente)
    int buscaPalabra(String palabra, String [][] array)
    Devolverá:
    la posición de la palabra buscada en el array
    o
    '-1' si la palabra no está.
    nota: podemos inicializar el array de la siguiente manera:
    String[][] diccionario = {
    {"casa", "house"},
    {"perro", "dog"},
    {"gato", "cat"}
    };
28. Juego de Euromillón: Crear un programa en Java para comprobar los boletos del juego del Euromillón. Para ello crearemos primero los números premiados de la siguiente manera:
    Generar 5 números (rango entre 1 y 50 incluidos)y 2 estrellas (Entre el 1 y el 12, incluidos).Se mostrarán por pantalla, tener en cuenta que los números no se puede repetir.
    Posteriormente pedir por teclado los números que se juegan, cada boleto pedirá 5 números y 2 estrellas. (Comprobar que los números introducidos están en el rango permitido)
    Finalmente mostrar el número de aciertos: números + estrellas.
    Cada vez que se compruebe un boleto, se pedirá si quiere continuar, ("si", "no").
29. Realizar un programa en Java que genere un array de 50 números entre el 0 y el 99. Posteriormente ordenar dicho array, para ello buscaremos el menor número para cada posición, empezando desde la posición cero. Finalmente se mostrará el array original y el array ordenado.
30. Pedir por teclado un número entre 0 y 99 y mostrarlo con letras.
    Ej: Introduce un número entre 0 y 99:17
    17 se escribe como diecisiete.
31. Hacer un conversor de números arábigos a romanos.
32. Hacer un conversor de números romanos a arábigos.
33. Juego del ahorcado:

Se tendrá un array con varias palabras (todas las que se deseen), posteriormente se generará un número al azar, que no supere el tamaño del array, y se seleccionará la palabra que ocupe esa posición en el array.
Al jugador se le mostrarán tantos guiones como letras tenga dicha palabra y empezarán a pedirle letras al usuario. Las letras se piden como String, y se comprueba que sólo tengan 1 letra, luego se comprobará si dicha letra está en la palabra y se mostrarán de nuevo los guiones, con las letras acertadas en su sitio:
Ejemplo:*"potro"*
fallos:0- - - - -
Introduce una letra: *o*
fallos:0- o - - o
Introduce una letra: *m*
fallos:1- o - - o
El juego finalizará de dos formas:
1.Se acierta la palabra.
2.Se han cometido 7 fallos
En ese momento se le preguntará al usuario si quiere volver a jugar (0=SI)
Opcional
Comprobar que las palabras elegidas al azar no se repiten. Cuando se hayan jugado todas las palabras, se mostrará un mensaje de reinicio de juego.

34. Dado el siguiente array (podéis generarlo aleatoriamente)

```
float notas[ ][ ] = {
{1145f, 6.75f, 5.95f, 0f},
{7458f, 2.68f, 3.6f, 0f},
{6689f, 10.00f, 9.75f, 0f},
{9745f, 7.25f, 6.95f, 0f}
};
```

Los datos se mostrarían de la siguiente manera:
ALUMNO: 1 Expediente 1145 Notas:6,755,950,00 Nota media:4,23
ALUMNO: 2 Expediente 7458 Notas:2,683,600,00 Nota media:2,09
ALUMNO: 3 Expediente 6689 Notas:10,00 9,750,00 Nota media:6,58
ALUMNO: 4 Expediente 9745 Notas:7,256,950,00 Nota media:4,73
Se pide:
1.Pedir por teclado las notas de la última evaluación (Actualmente a 0)
2.Mostrar los datos del mejor alumno y del peor.
3.Mostrar la nota media del curso completo (igual que la tabla superior)
4.Mostrar los datos de todos los alumnos que superen la nota media.

LOS EJERCICIOS A PARTIR DE AQUÍ SON TOTALMENTE OPCIONALES Y PUEDEN INCLUIR CONCEPTOS COMPLEJOS QUE EL ALUMNO NO TIENE POR QUÉ ENTENDER.
LA IDEA ES PONER A SU DISPOSICIÓN UN MAYOR CONJUNTO DE EJERCICIOS PARA QUE LAS PERSONAS QUE HAN TERMINADO LOS ANTERIORES PUEDAN PRACTICAR Y PROFUNDIZAR.
HAY TAMBIÉN ALGUNOS EJERCICIOS CUYA RESOLUCIÓN NO IMPLICA UNA MAYOR DIFICULTAD.

35. Escribe un programa que, dado un valor para la variable x, y una serie de coeficientes (a, b, c, d…), evalúe la función polinómica correspondiente (f(x)= … + ax3 + bx2 + cx + d en el ejemplo anterior). Tanto el valor de x como el de los coeficientes se suministran de forma interactiva. El número de coeficientes puede ser variable.
36. Escribe un programa que resuelva ecuaciones de segundo grado, pidiendo interactivamente los coeficientes. Se deben dar soluciones complejas si no hay solución en el dominio de los reales.
37. Escribe un programa que transforme un número complejo representado en coordenadas cartesianas a polares y viceversa. Un número complejo en coordenadas cartesianas es a+bi. Para pasarlo a coordenadas polares: (sqrt(a2+b2), arctan(b/a)) y así queda expresado en forma (m,O). Para volverlo a pasar a cartesianas: a=m cos O, b=m sin O.
38. Escribe un programa que multiplique dos números complejos leídos de teclado. Se debe de dar la posibilidad al usuario de introducirlos en cualquiera de las dos representaciones según quiera, y el resultado debe ser mostrado en la representación que él elija.
39. Escribe un programa que factorice polinomios de coeficientes enteros usando la regla de Ruffini.
40. Escriba un programa que calcule el número mínimo de monedas que hay que dar para devolver una cantidad. Se presupone que usamos euros y disponemos de monedas de 1, 2, 5, 20, 50, 100 y 200 céntimos de euro.
41. La técnica de compresión de datos RLE (Run Length Encoding) se utiliza para reducir el tamaño de los ficheros de datos. Se basa en la sustitución de una secuencia de datos repetidos, por un dato especial (marcador) seguido del número de repeticiones y del dato que se repite. Evidentemente, dicha sustitución se realiza sólo en los casos en los que resulte favorable. Utilizando el carácter '\' como marcador, un ejemplo sería partir de la secuencia "aaaaaaabbccccddde\fg.", de la que obtendríamos "\7abb\4cddde\1\fg.". Obsérvese que sólo se sustituye a y c porque se ahorra espacio, mientras que en el resto de los casos o no se gana nada (d) o incluso se pierde. Un caso especial se produce cuando hay caracteres marcadores, que se sustituyen siempre aunque no estén repetidos. Leyendo carácter a carácter, escribe un programa que lea de teclado una secuencia de caracteres terminada en '.' y visualice la secuencia comprimida según la técnica explicada. En una segunda opción, debe permitir también realizar el paso contrario, de secuencia comprimida a descomprimida.
42. Escriba un programa que lea una secuencia de caracteres (carácter a carácter), terminada con un punto, en las que se expresan las fechas de nacimiento y defunción de una persona (sólo el mes y el año), y escriba su edad en el momento del fallecimiento. La secuencia puede contener otra información además de las fechas, pero verifica las siguientes restricciones:

    a. No aparece ninguna otra información numérica.
    b. En cada fecha, el mes se representa con uno o dos dígitos, y el año con cuatro.
    c. Dentro de una fecha, el mes y el año pueden aparecer en cualquier orden.
    d. Las dos fechas pueden aparecer en cualquier orden, pero los datos de una fecha (mes y año) aparecen antes que los de la otra.

    Ejemplos de ejecución:

    ubuntu$ edad

Mortadelo muere en un accidente de tráfico en febrero de 1956 (mes=2); había nacido en octubre (mes 10) de 1905.

Edad de fallecimiento: 50 años y 4 meses.

ubuntu$ edad

Filemón, nacido en 1959 el mes 10; fallecido el 6/1992.

Edad de fallecimiento: 32 años y 8 meses.

ubuntu$ edad

Nerón, muerto en un incendio el 10/1991, había nacido el 1 (enero) de 1891.

Edad de fallecimiento: 100 años y 9 meses

43. Se dice que una sucesión de enteros (a1, a2, …, an) forman una montaña si existe un h entre 1 y n tal que la sucesión crece hasta ah y luego decrece. Una sucesión es un valle si ocurre justo al revés (decrece hasta un ah y luego crece). Se define la longitud de una montaña (o valle) como el número de intervalos que hay en la sucesión y la altura como el mayor (o menor si se trata de un valle) de los números que la forman. Ejemplo: la sucesión 1, 3, 6, 21, 15 es una montaña de longitud 4 y altura 21. Suponiendo secuencias de números naturales no vacías y terminadas en un entero negativo, se pide un programa con dos opciones, la primera que visualice el número de montañas que hay en una secuencia de montañas, la longitud de la montaña más larga, y la altura de la montaña más alta, y la segunda, que visualice la longitud de la montaña o valle más largo, indicando si se trata de valle o montaña.
44. Modifique el programa anterior para tratar secuencias de enteros terminadas por un punto y entre los que podría haber comas, u otros separadores.
45. Se desea escribir un programa para convertir números entre bases. Recordar que en bases superiores a 10, cuando me quedo sin dígitos numéricos, se suelen utilizar letras mayúsculas por orden y empezando por la A. El programa empezará solicitando dos bases. A continuación mostrará las siguientes opciones (supongamos para el ejemplo que hemos introducido las bases 6 y 15):

    0. Salir

    1. Cambiar base 6 por otra

    2. Cambiar base 15 por otra

    3. Convertir número en base 6 a número en base 15

    4. Convertir número en base 15 a número en base 6

46. Escribe un programa que implemente una calculadora romana, dotada de las operaciones básicas (suma, resta, multiplicación y división entera). Los números

se introducirán como número romano y el resultado también se mostrará como número romano. La calculadora memorizará cada resultado y podrá ser utilizado para la operación siguiente: si se introduce una A, significará "último resultado obtenido". Para terminar la ejecución bastará con introducir un punto. Ejemplo de ejecución:

ubuntu$ calromana

> III + II

= V

> III*III

= IX

> A*V

= XLV

> .

ubuntu$

47. Escribe un programa que solicite al usuario dos matrices y las sume.
48. Escribe un programa que solicite al usuario una matriz y calcule su traspuesta.
49. Escribe un programa que calcule el producto de dos matrices introducidas por teclado.
50. Escribe un programa que calcule el determinante de una matriz 3x3 introducida por teclado. El determinante de una matriz se calcula así: a11a22a33 + a21a32a13 + a12a23a31 - a13a22a31 - a23a32a11 - a21a12a33, siendo aij el elemento de la fila i y columna j.
51. La matriz adjunta de una matriz es aquella que se obtiene de sustituir cada elemento (a_{ij}) por su adjunto (A_{ij}). El adjunto de un elemento es su menor complemantario (M_{ij}) con signo positivo o negativo según sea par o impar la suma de su número de fila y su número de columna. Es decir $A_{ij}=(-1)^{(i+j)}M_{ij}$. El menor complementario de un elemento de una matriz cuadrada es el determinante de la matriz que obtenemos al suprimir su fila y su columna. Escribe un programa que calcule la matriz adjunta de una matriz 3x3 introducida por teclado.
52. La matriz inversa de una matriz A es otra matriz que representamos por A^{-1} y que verifica $A\ A^{-1} = A^{-1}A = I$. Siendo I la matriz identidad (con unos en su diagonal y ceros en el resto de elementos). La matriz inversa se puede calcular como $A^{-1}=(1/|A|)((Adj(A))^T)$, es decir, 1 entre su determinante por la matriz adjunta de A traspuesta. Solamente tienen inversa las matrices cuadradas cuyo determinante es distinto de cero. Escribe un programa que calcule la matriz inversa de una matriz 3x3 introducida por teclado. El programa debe de comprobar que se cumple la definición de matriz inversa obteniendo la matriz identidad una vez calculada.
53. Escribe un programa que acepte una hora en formato militar y la transforme al formato habitual (horas, minutos y AM/PM) y viceversa. Para resolver el problema, usar registros en la definición de los tipos de datos.

54. El juego de la vida pretende simular el comportamiento de seres absolutamente simples (unicelulares) sobre las celdas de un tablero rectangular. Cada celda, excepto las de la periferia, tiene ocho vecinas y en cada una de ellas puede vivir un solo bicho. Sea numvecinas(i,j) el número de celdas contiguas a la celda (i,j) que están ocupadas.

    Cada cierto tiempo se produce una renovación generacional siguiendo las siguientes reglas:

    - El bicho que vive en la celda (i,j) sobrevive sólo si numvecinas(i,j) es 2 ó 3. En caso contrario, muere por aburrimiento o escasez de recursos.
    - Un bichito nace en la celda vacía (i,j) si y sólo si numvecinas(i,j) es exactamente 3.

    Escribe un programa que implemente este juego de la vida. Inicialmente se le preguntará al usuario por las dimensiones del tablero (máximo 20x20) y se ofrecerá al usuario dos opciones: generación de una configuración de bichos en el tablero aleatoriamente, o generación de un tablero en blanco.

    Una vez hecha esta inicialización se mostrará el resultado por pantalla, y se le ofrecerán al usuario las siguientes opciones:

    - Generar un nuevo tablero aleatoriamente.
    - Generar un nuevo tablero en blanco.
    - Introducir un nuevo bicho en una celda vacía dando sus coordenadas.
    - Eliminar un bicho en una celda donde haya bicho, dejándola vacía.
    - Simular un determinado número de generaciones.
    - Salir.

    Tras cada operación, se deberá mostrar el tablero resultante. Al generar un nuevo tablero, se le dará la opción al usuario de elegir unas nuevas dimensiones.

    Pista 1: Puesto que los cambios se tienen que realizar a la vez en todo el tablero, para realizar los cálculos anteriores se utilizarán dos tableros: uno que contenga la generación actual y otro la siguiente.

    Pista 2: Se recomienda implementar la función booleana posValida(i,j) que devuelva verdad sí y sólo si la posición (i,j) está dentro del tablero.

55. Desarrolla un juego de “hundir la flota” para jugar contra el ordenador. El ordenador colocará inicialmente sus barcos en su tablero sin mostrarlos por pantalla para que el usuario no conozca su ubicación. Le pedirá al usuario que introduzca información sobre la ubicación de sus barcos (los del usuario). Ambos jugarán con el mismo número de barcos y de las mismas dimensiones. Y se procederá a jugar por turnos, repitiendo disparo si se acierta y perdiendo el turno si el proyectil cae en una casilla en la que haya “agua”. Tras cada jugada, se mostrará información al usuario sobre la situación de su tablero, y sobre la situación que conoce del tablero del ordenador (es decir, sólo los disparos que ha hecho, sobre qué posición y sus aciertos y errores). La estrategia a utilizar por el ordenador para ganar, se deja en manos de cada desarrollador y se puede hacer tan sofisticada como se desee. No está permitido que el ordenador utilice

información sobre la situación de los barcos del adversario, se supone que el juego es limpio.

56. Desarrolla un juego de la vida 3D. Ahora las casillas no son cuadradas sino cúbicas. Cada casilla puede tener hasta 26 vecinas. Se debe pedir por pantalla tanto las dimensiones del tablero (3 valores) como el número mínimo y máximo de vecinas que debe tener cada celda para sobrevivir en el caso de que esté ocupada y el número mínimo y máximo de vecinas que debe tener una casilla vacía para que se genere vida en el turno siguiente.
57. Desarrolla el juego del tic-tac-toe para 2 jugadores.
58. Desarrolla el juego del conecta-4 para 2 jugadores.
59. Desarrolla el juego del 3 en raya 3D. Hay que colocar fichas hasta rellenar el tablero. Quien más 3 en raya tenga gana. Cada partida la empieza un jugador, teniendo que jugar hasta que el número de partidas ganadas por un jugador supere en 2 a las ganadas por el otro.